

Table of Contents

はじめに	1.1
1. 教材の準備	1.2
1.1 開発に必要なソフトウェア	1.2.1
1.2 アプリのダウンロード	1.2.2
1.3 教材のダウンロード	1.2.3
1.4 サンプルのダウンロード	1.2.4
1.5 Checkツール	1.2.5
2. DeviceWebAPIManagerの設定	1.3
2.1 DeviceWebAPI Managerの設定	1.3.1
2.2 DeviceWebAPI Managerの生存確認	1.3.2
2.3 DEMOアプリで動作確認	1.3.3
2.4 ATOMのインストール	1.3.4
2.5 setting.jsの設定	1.3.5
2.6 Developer Tool	1.3.6
3. Host基礎	1.4
3.1 バイブルーション	1.4.1
3.2 写真を撮影する	1.4.2
3.3 プレビューを開始	1.4.3
4. Robotを操作する	1.5
4.1 Robotカーへの接続	1.5.1
4.2 Robot操作の基礎	1.5.2
4.3 Robot操作のサンプル	1.5.3
5. FaBo基礎	1.6
5.1 FaBo Pluginの接続	1.6.1
5.2 LEDの操作(アドレスバー)	1.6.2
5.3 LEDの操作(JavaScript)	1.6.3
5.4 距離を取得	1.6.4

5.5 温度を取得	1.6.5
5.6 照度を取得	1.6.6
6. FaBo仮想デバイス	1.7
6.1 3軸加速度センサーの登録	1.7.1
6.2 3軸加速度センサーの取得	1.7.2
7. その他	1.8
7.1 認証ダイアログの表示/非表示設定	1.8.1
7.2 Firmataのインストール	1.8.2
7.3 USB Deviceの認識	1.8.3
7.4 デモサイトをスマフォにいれる	1.8.4
8. トラブルシューティング	1.9
8.1 トラブルシューティング	1.9.1

DeviceWebAPI Docs

本ドキュメントについて

DeviceWebAPIの学習用ドキュメントになっています。

対象レベル

プログラムを過去に書いたことがある

使用言語

HTML5/JavaScript

使用ハード

FaBo

使用ソフトウェア

- [Google Chrome](#)
- [Sublime text](#)

Powered by [FaBo](#)

1.1. 開発で必要なソフトウェア

研修で使用するソフトウェア

アプリ	URL
Google Chrome	https://www.google.co.jp/chrome/browser/desktop/
ATOM	https://atom.io/

1.2. アプリのダウンロード

研修で使用するアプリ

アプリ	必要なVersion	URL
DeviceWebAPI Manager	2.1.2	https://play.google.com/store/apps/details?id=org.deviceconnect.android.manager
FaBo Device Plugin	2.2.5	dConnectDeviceFaBo_v2_2_5.apk

1.3. ドキュメントのダウンロード

ドキュメントのダウンロード

項目	概要
最新版の教材	devicewebapi-docs.pdf

1.4. サンプルのダウンロード

項目	概要
サンプル	sample.zip

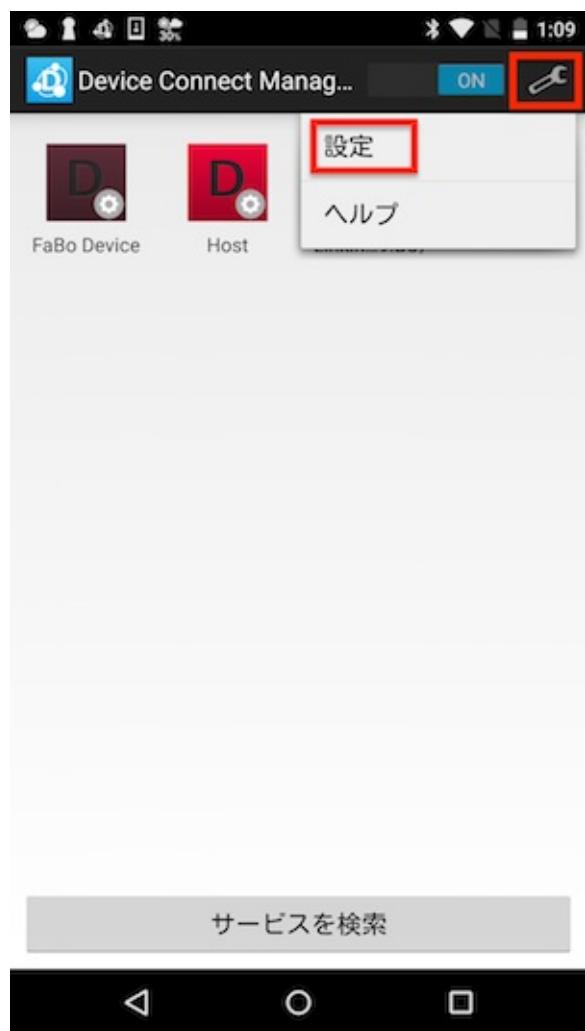
1.5 チェックツール

項目	概要
チェックツール	http://www.fabo.io/tool/

2.1 DeviceWebAPI Managerの設定

DeviceWebAPI Managerを起動します。

設定を選択します。



これから設定の変更の処理をするため、一旦、DeviceWebAPI Managerをオフにします。

2.1 DeviceWebAPI Managerの設定



2.1 DeviceWebAPI Managerの設定



赤線で囲まれたチェックボックスを以下のように変更します。

項目	処理
外部IPを許可	選択する
Local OAuth	選択をはずす
Originの有効化	選択をはずす

2.1 DeviceWebAPI Managerの設定



2.1 DeviceWebAPI Managerの設定



設定の変更が完了したら、DeviceWebAPI Managerを再びオンにします。

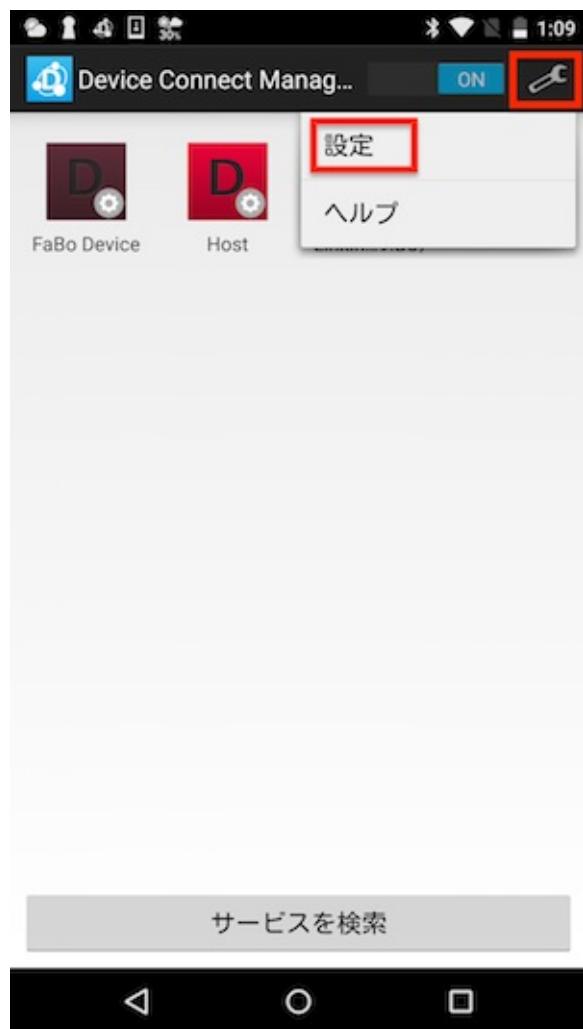
2.1 DeviceWebAPI Managerの設定



2.2 DeviceWebAPI Managerの生存確認

IPアドレスの確認

設定>設定の項目のHostとPort番号をメモしておきます。



2.2 DeviceWebAPI Managerの生存確認



DeviceWebAPI Managerの生存確認

DeviceWebAPI Managerの生存確認は、Checkツールを用いておこないます。

<http://www.fabo.io/tool/>

接続ができた場合

2.2 DeviceWebAPI Managerの生存確認



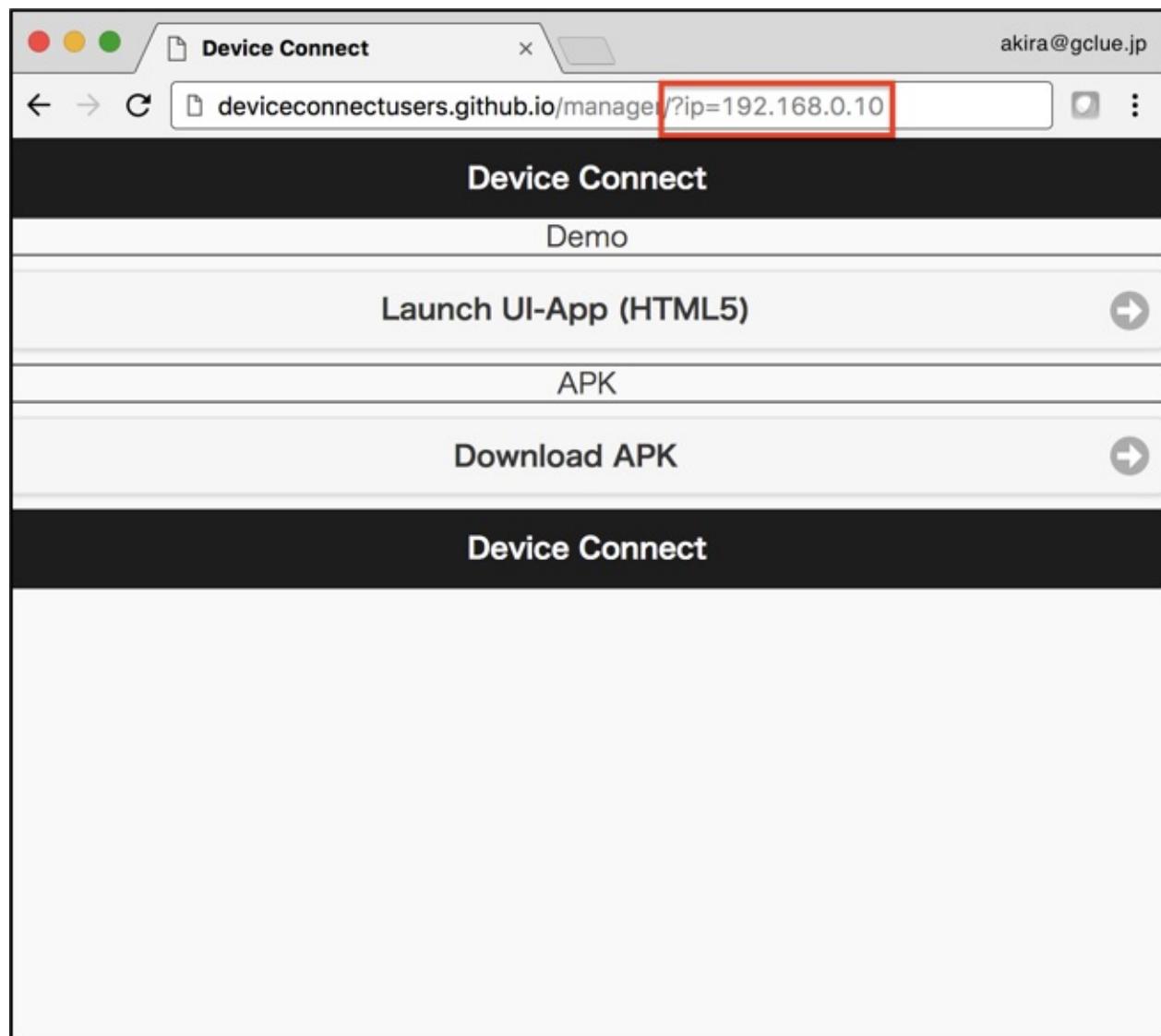
どこか設定が間違っている場合

```
{"result":1,"product":"Device Connect Manager","errorCode":18,"version":"glasses-release-20170526-1-geecce84b","errorMessage":"Origin is not specified."}
```

2.3 DEMOアプリで動作確認

<http://deviceconnectusers.github.io/manager/>にDEMOアプリがあります。

202. DeviceWebAPI Managerの生存確認でメモしたIPアドレスを、URLの一番最後に、`?ip=メモしたIPアドレス` の形式で追加し、アクセスします。



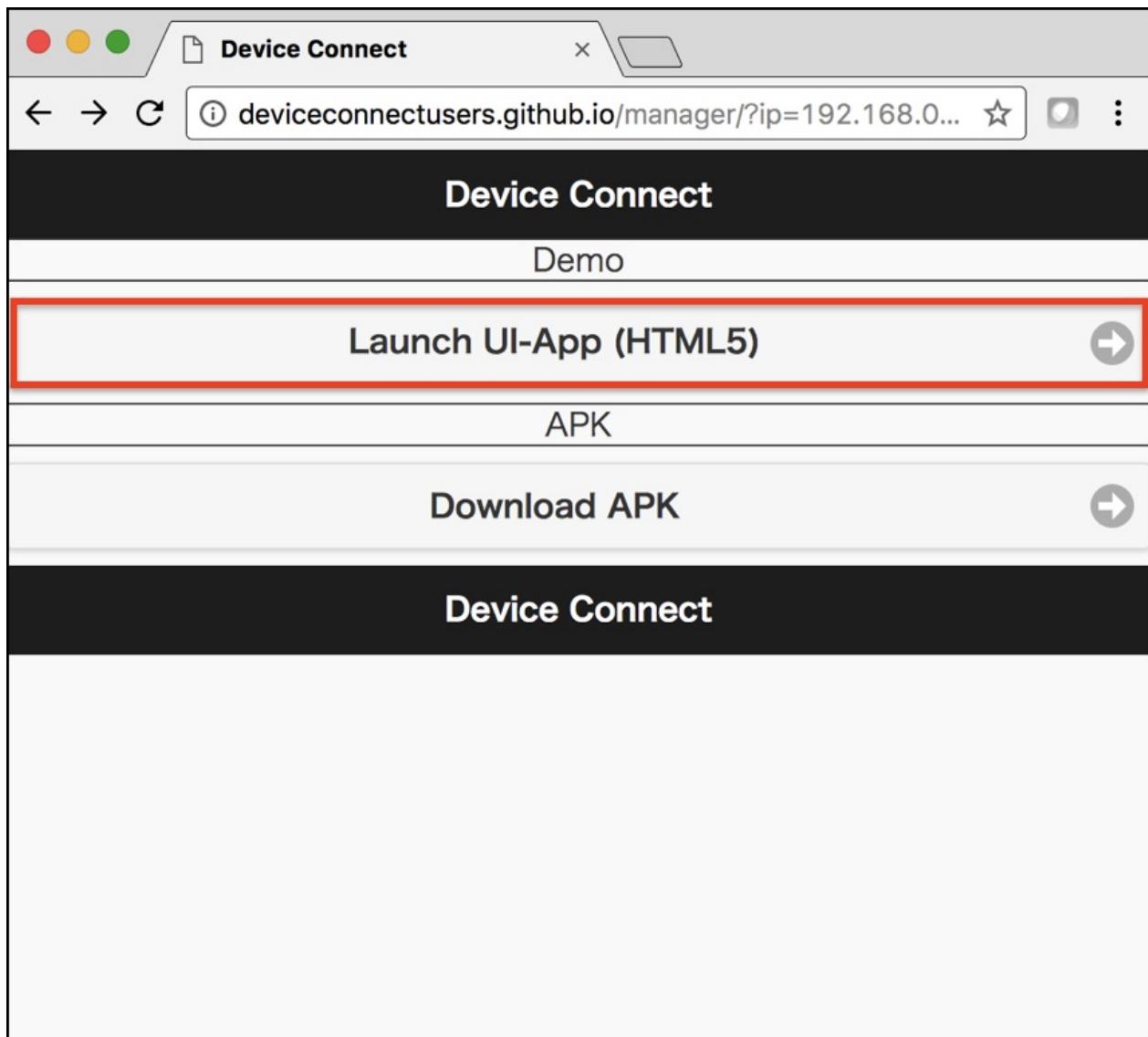
今回は、アプリの例として、スマートフォンに搭載されているライトを点灯させるアプリを起動してみます。

スマートフォンのライトを点灯させる

スマートフォンに搭載されているライトを点灯させるアプリです。

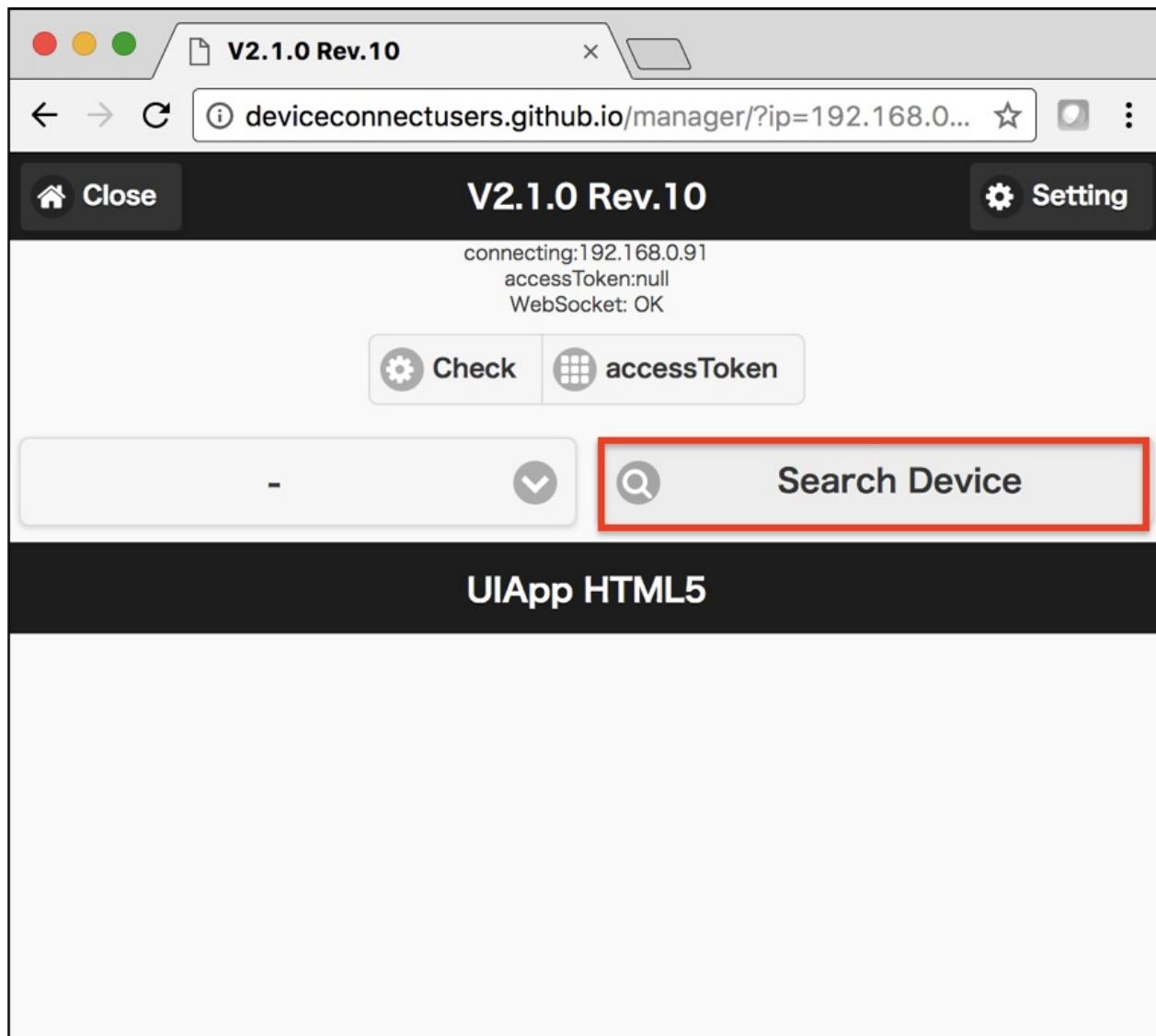
2.3 DEMOアプリで動作確認

トップ画面のLaunch UI-App(HTML5)をクリックします。



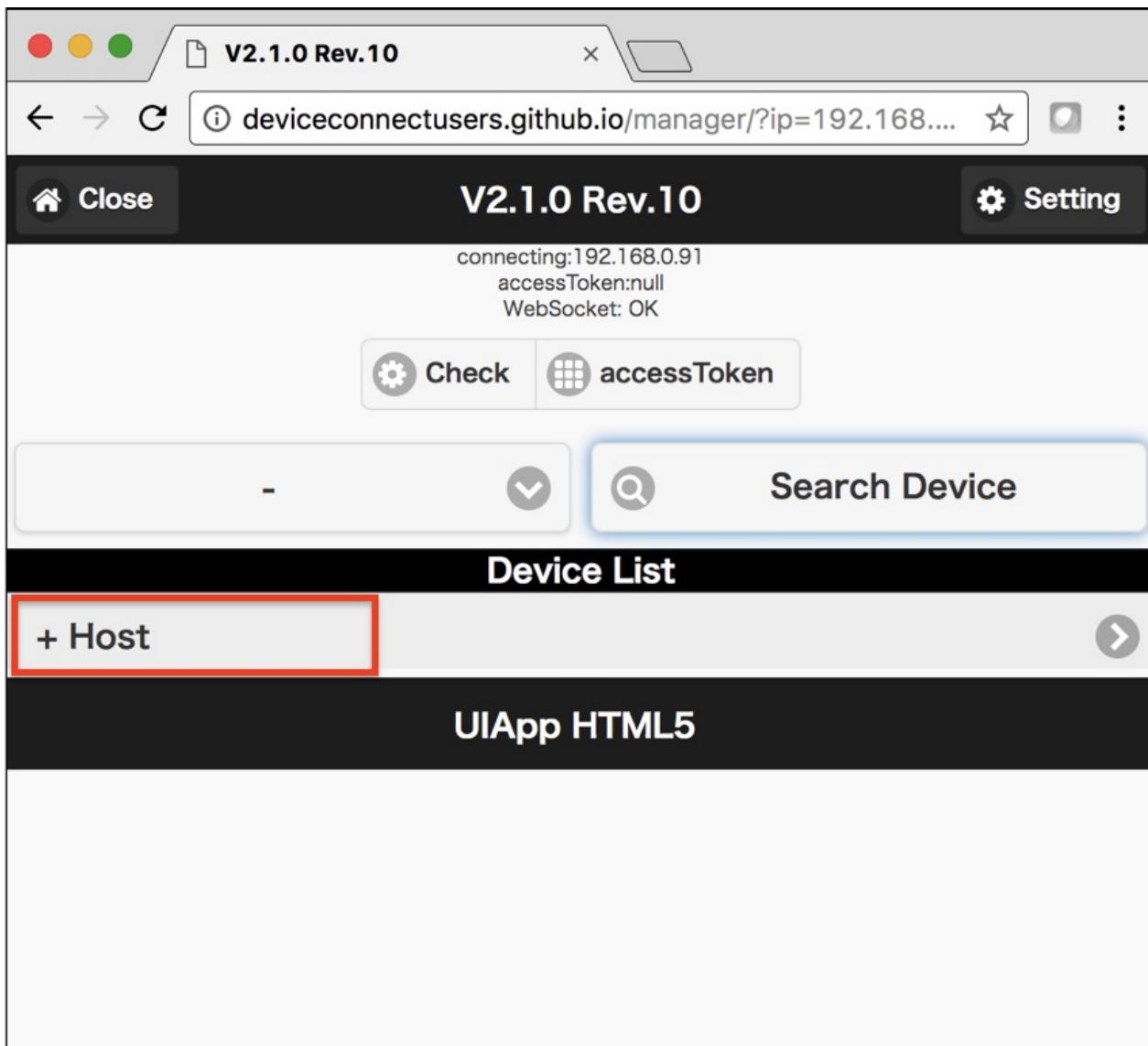
次に、Search Devicesをクリックします。

2.3 DEMOアプリで動作確認



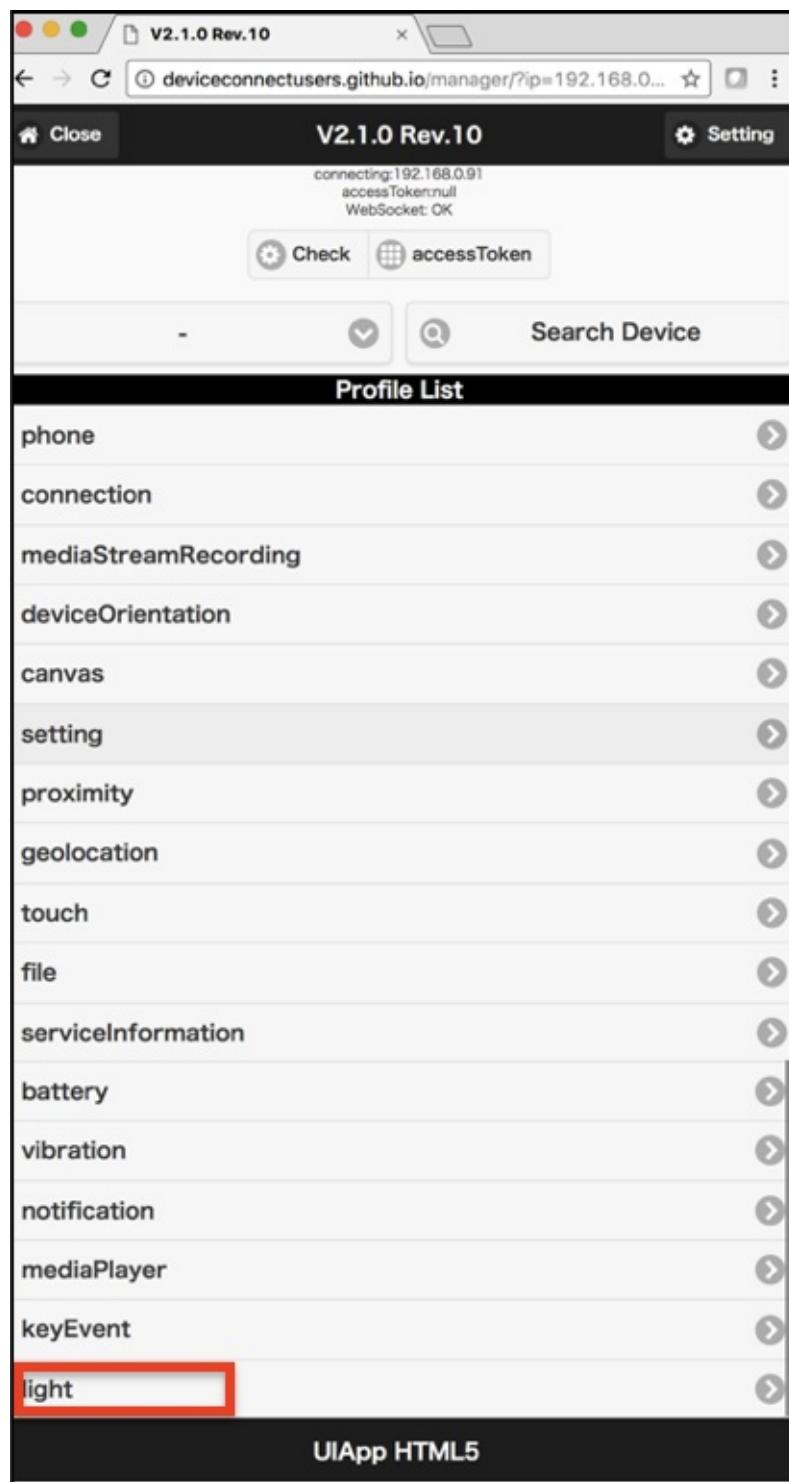
そして、Hostをクリックします。

2.3 DEMOアプリで動作確認

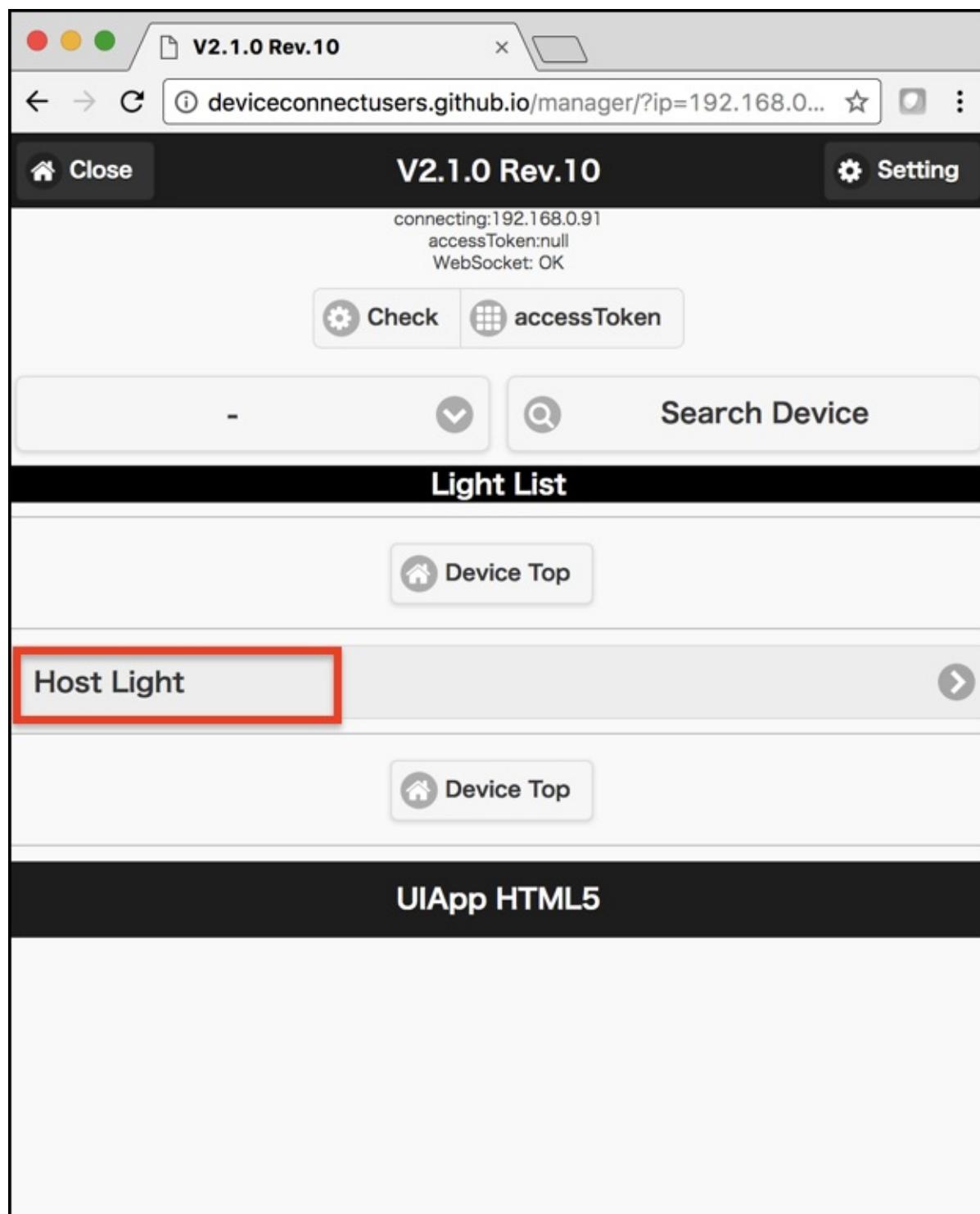


アプリ一覧が表示されるので、今回はスマートフォンのライトを点灯させる
light を選択します。

2.3 DEMOアプリで動作確認

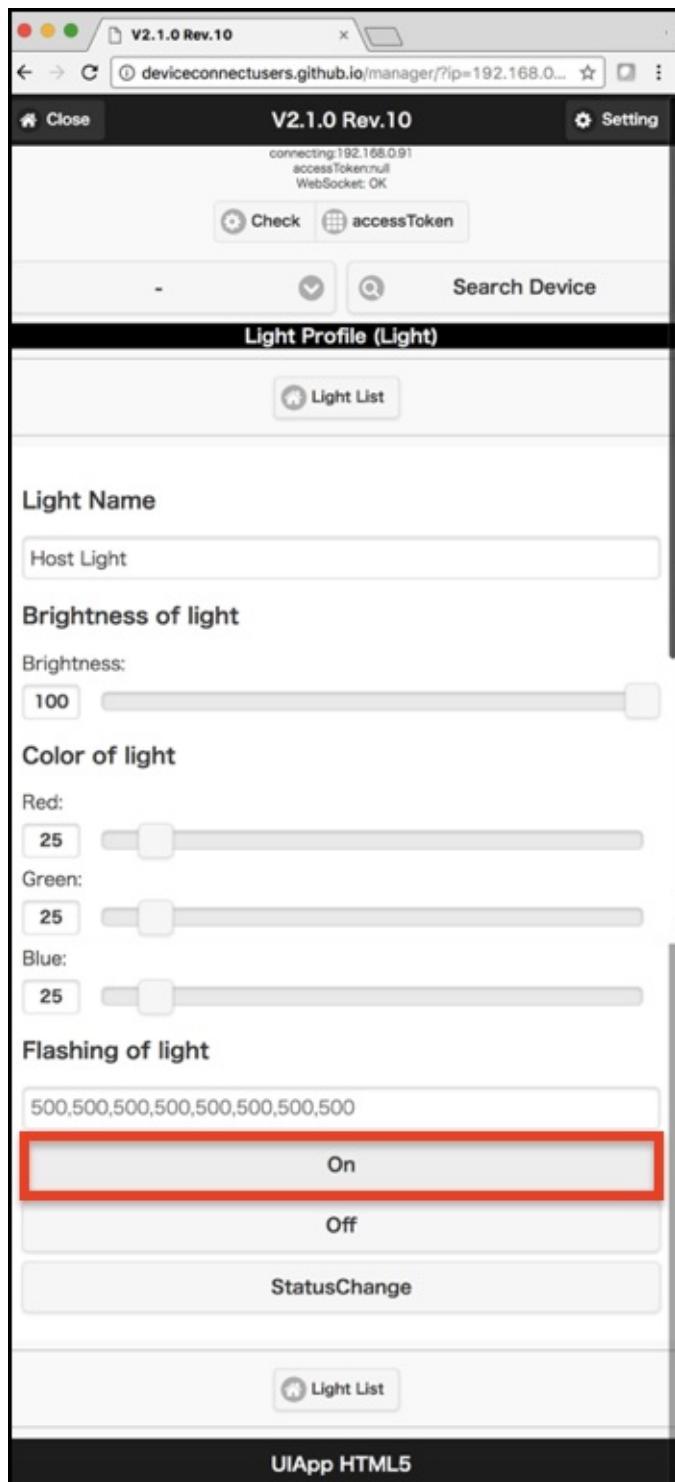


Host Lightを選択します。



このような画面が出てるので、実際に点灯させます。 点灯させるためには、下の On ボタンを押すことで、スマートフォンのライトが点灯します。

2.3 DEMOアプリで動作確認



点灯させた際の例

2.3 DEMOアプリで動作確認



2.3 DEMOアプリで動作確認

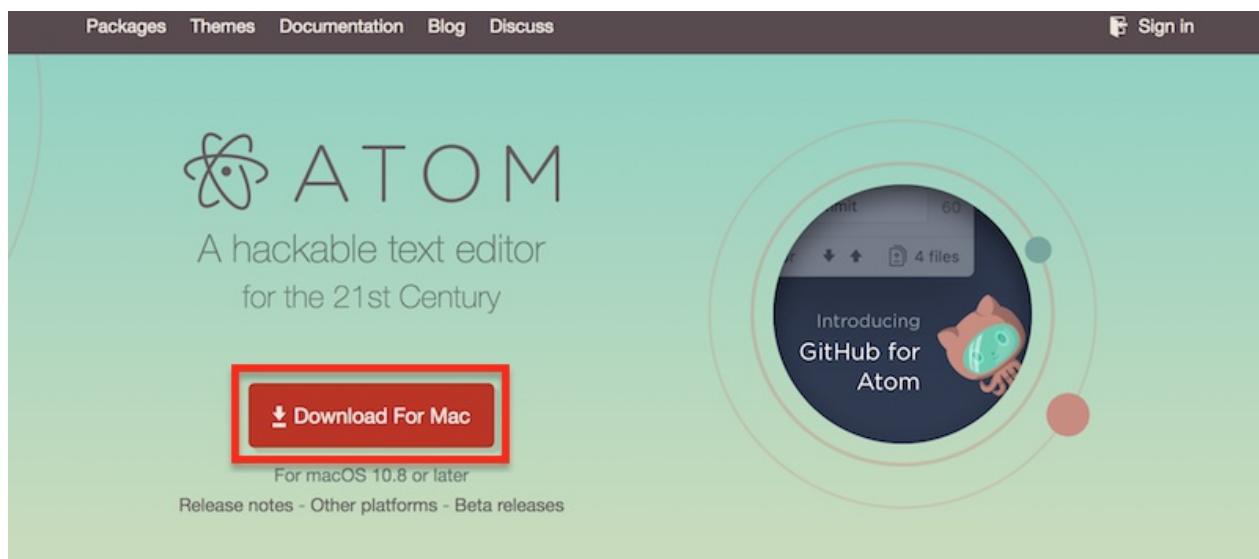
また、ライトを消灯するためには、On のボタンの下にあるOff ボタンでライトを消灯できます。スライダなどをいじることによって、光量や色を変更することも可能です。

2.4 ATOMのインストール

ATOMのダウンロード

ATOMを下記URLからダウンロードしてきます。

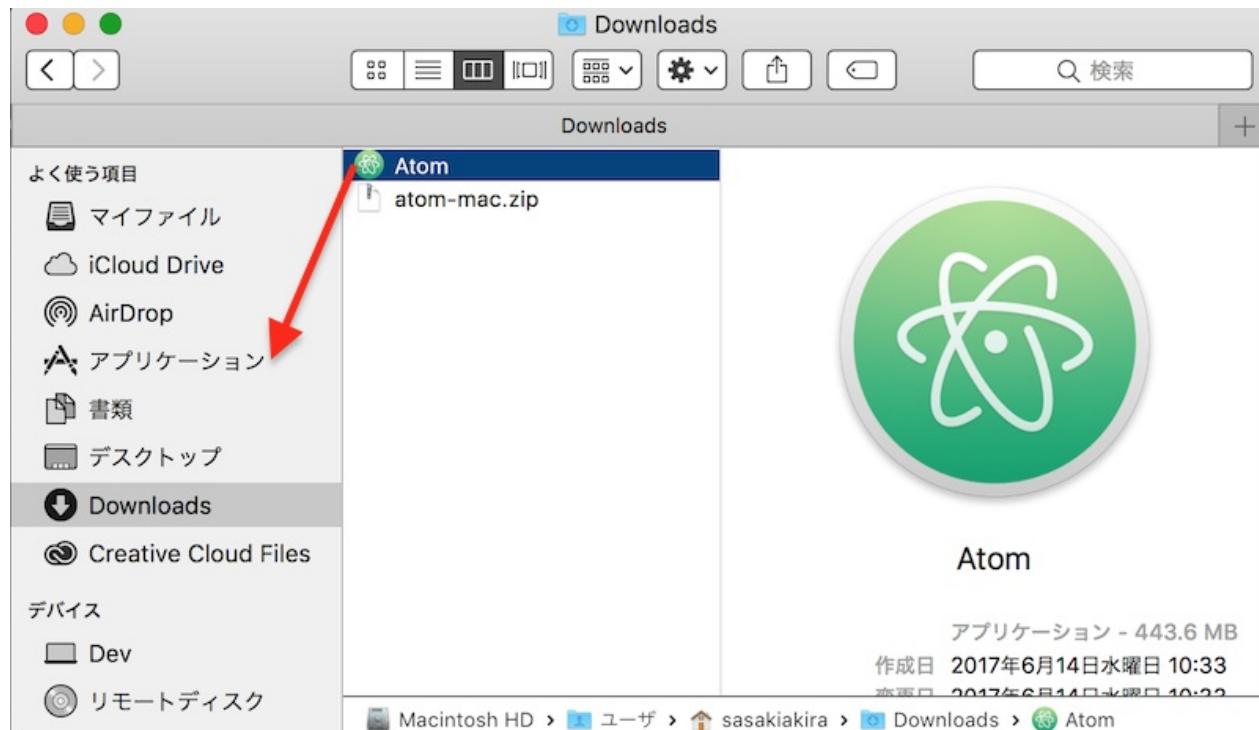
<https://atom.io/>



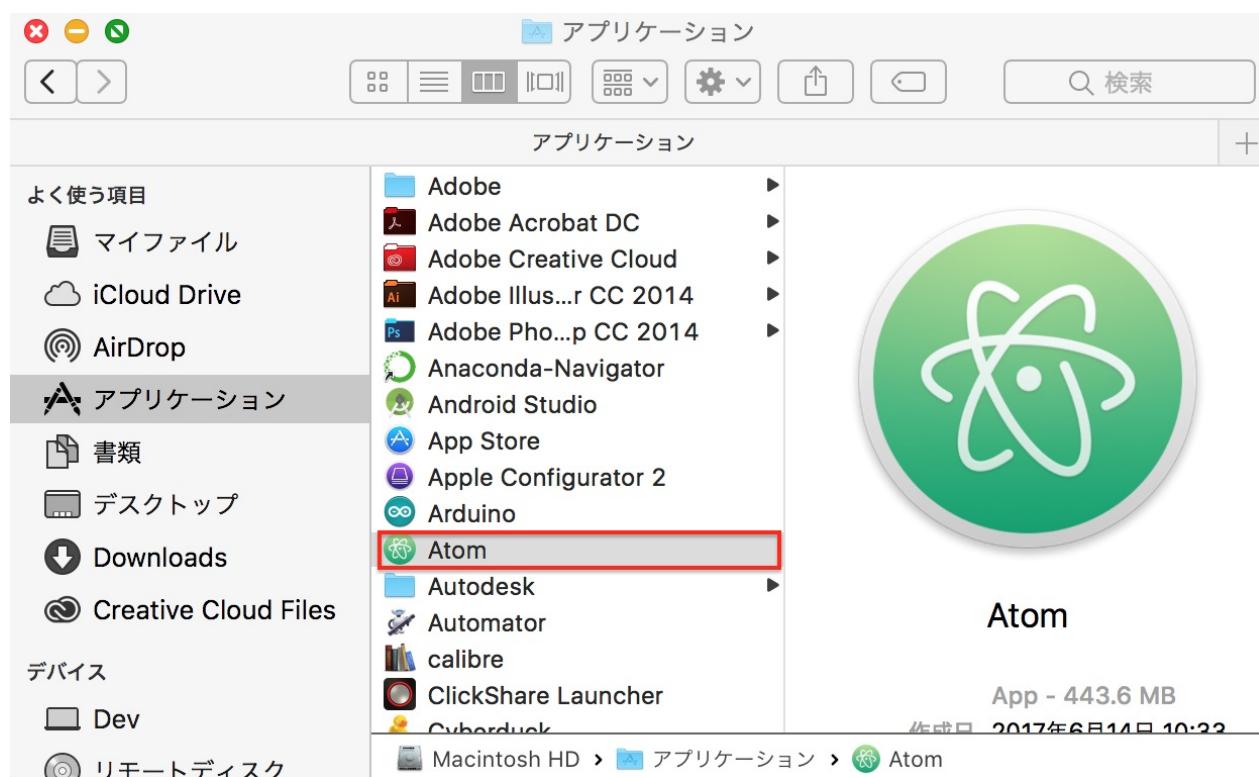
ATOMのインストール(OSXの場合)

解答し、生成されるATOMアイコンを/Applicationに移動します。

2.4 ATOMのインストール



ATOMアイコンをクリックし、ATOMを起動します。

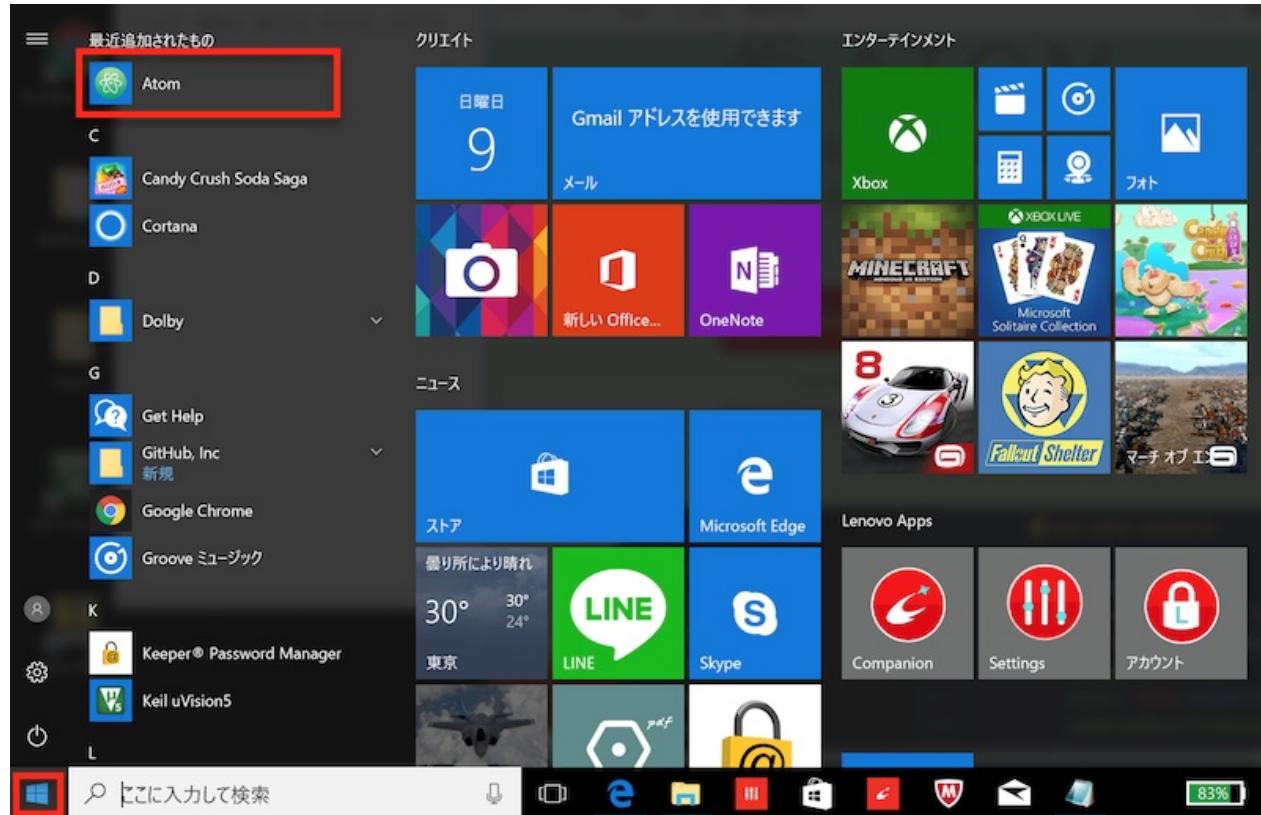


ATOMのインストール(Windowsの場合)

ダウンロードしたアイコンをクリックすると、インストールが開始されます。初回インストール時は自動的に実行されます。

2.4 ATOMのインストール

2回目以降は、Windowsボタンを選択し、Atomアイコンを選択し、実行します。



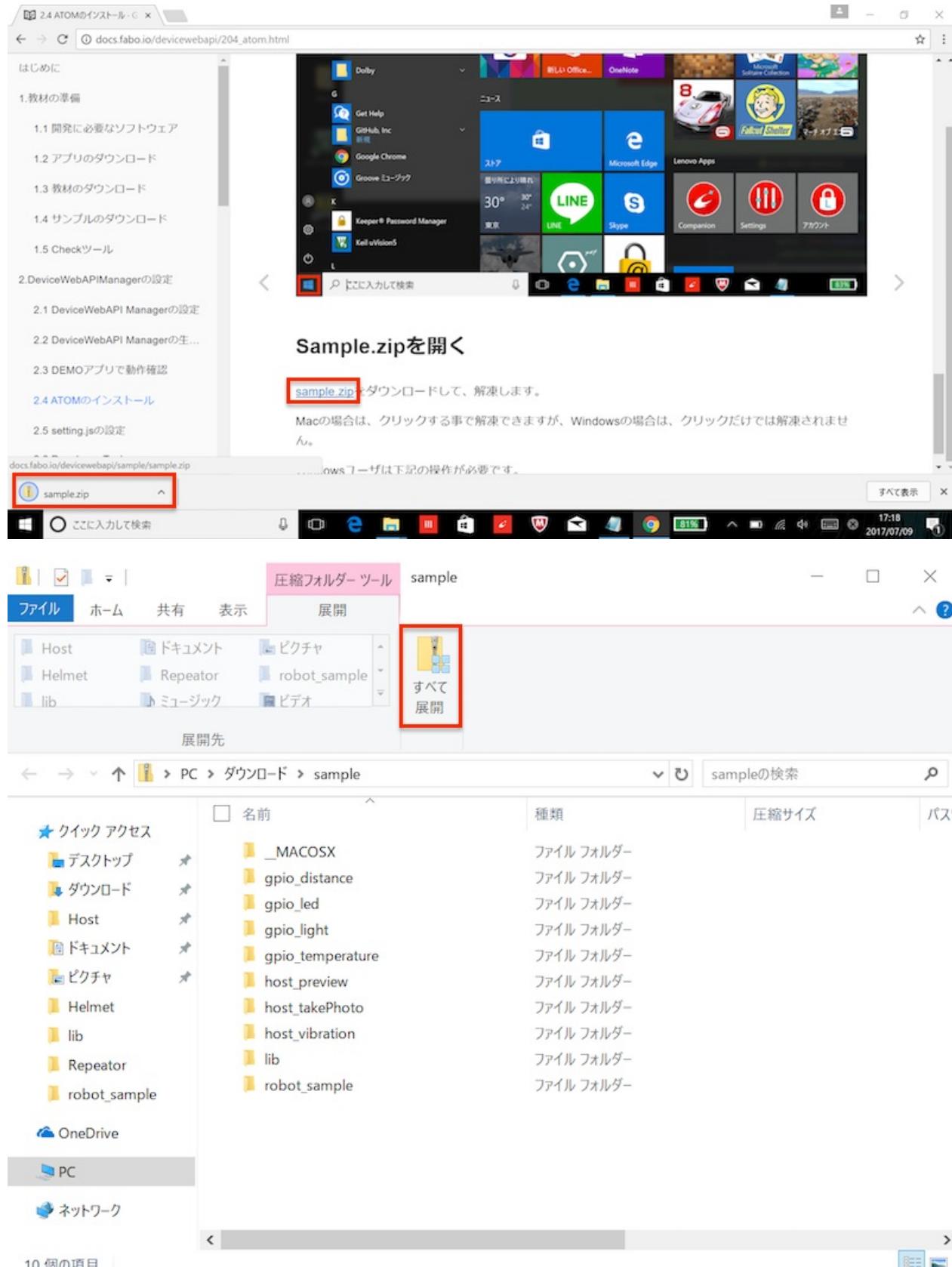
sample.zipを開く

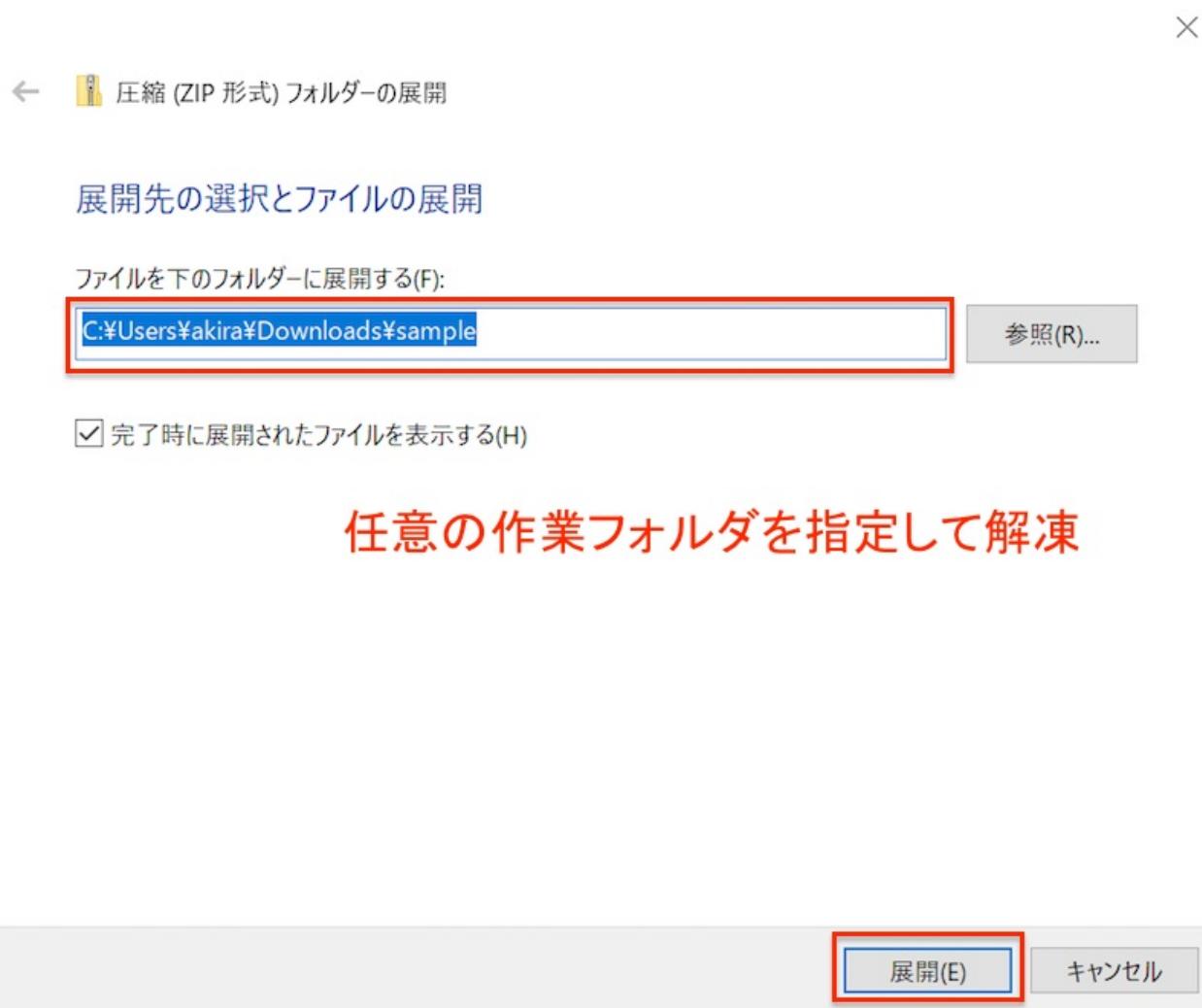
sample.zipをダウンロードして、解凍します。

Macの場合は、クリックする事で解凍できますが、Windowsの場合は、クリックだけでは解凍されません。

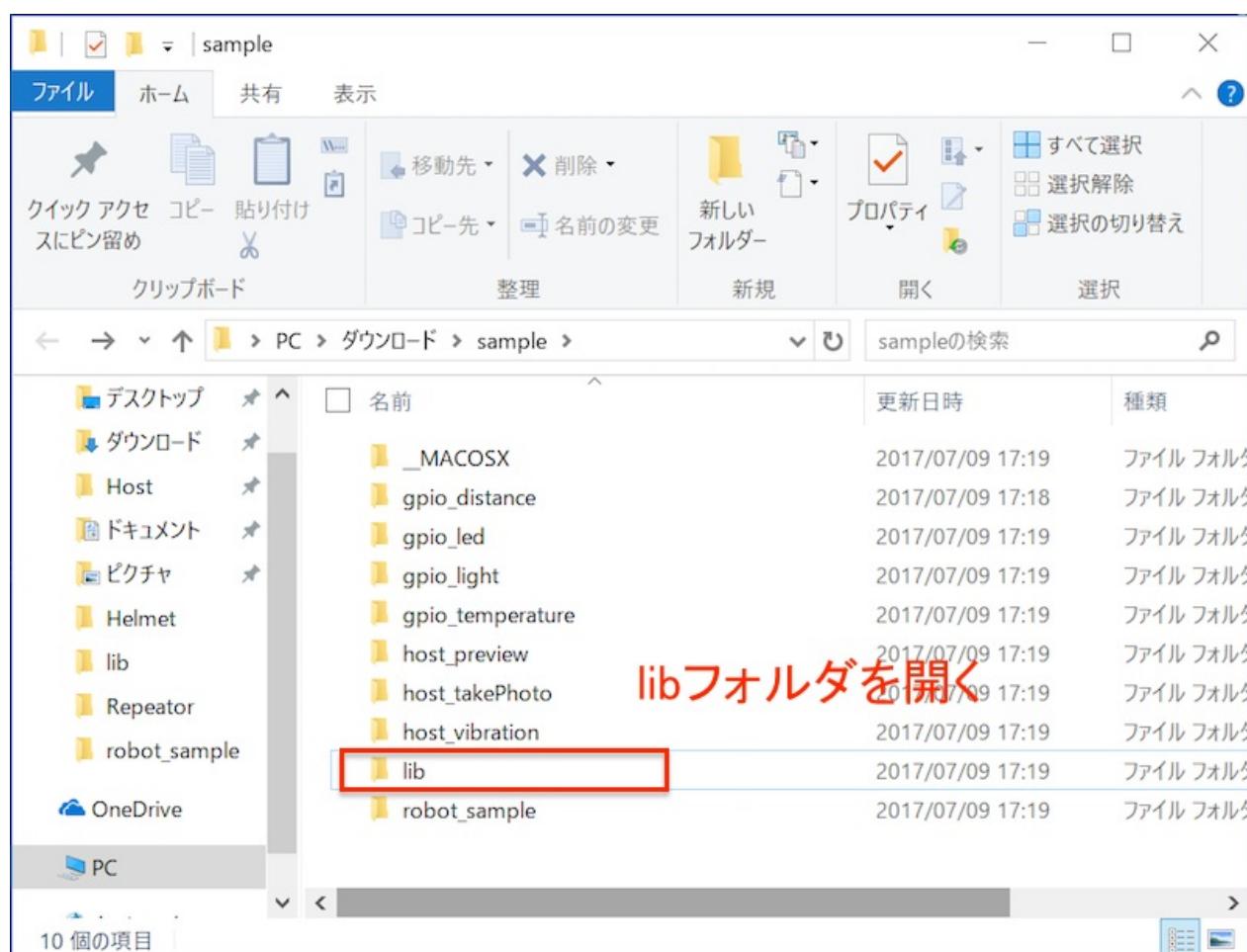
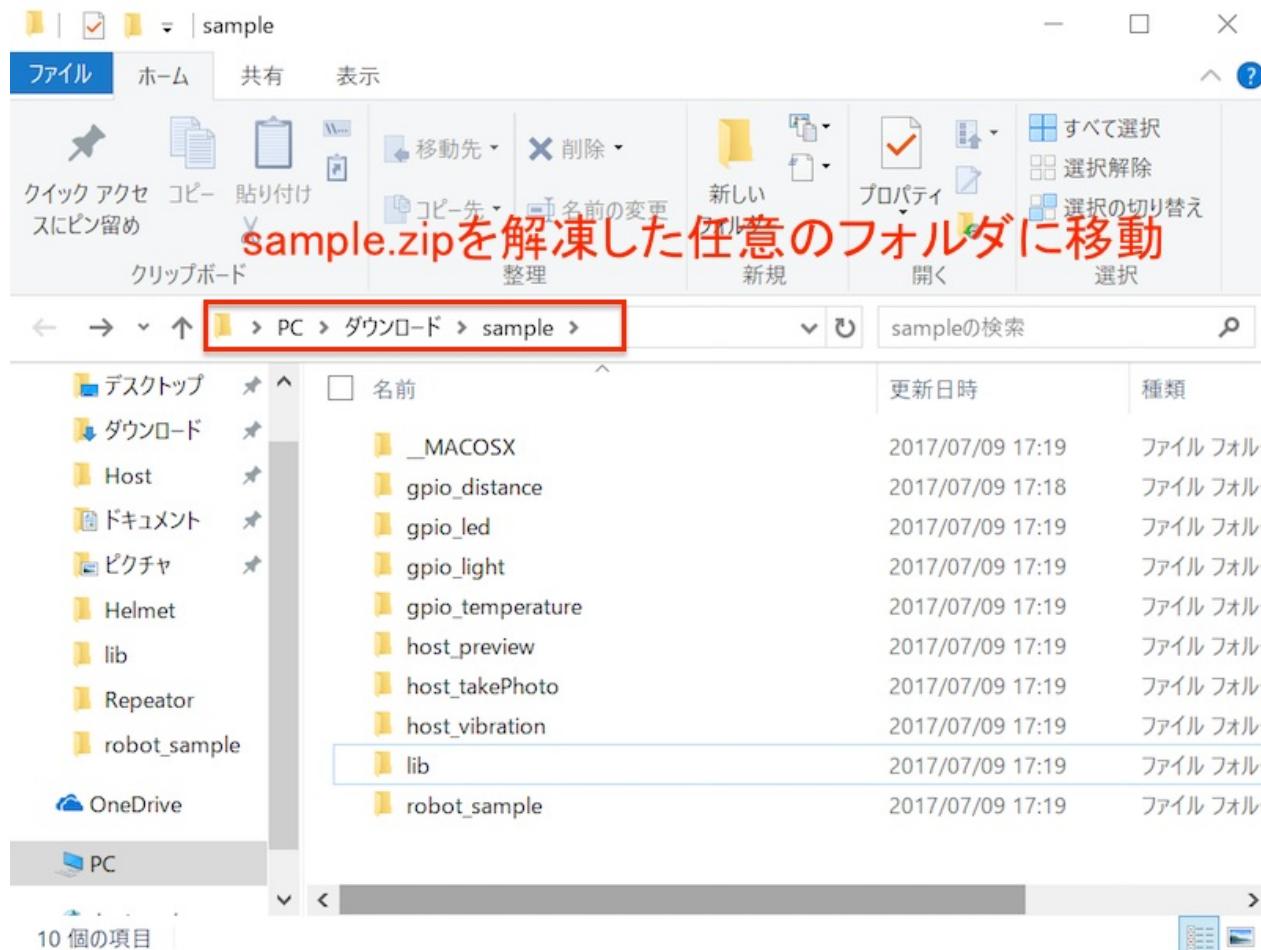
Windowsユーザは下記の操作が必要です。

2.4 ATOMのインストール



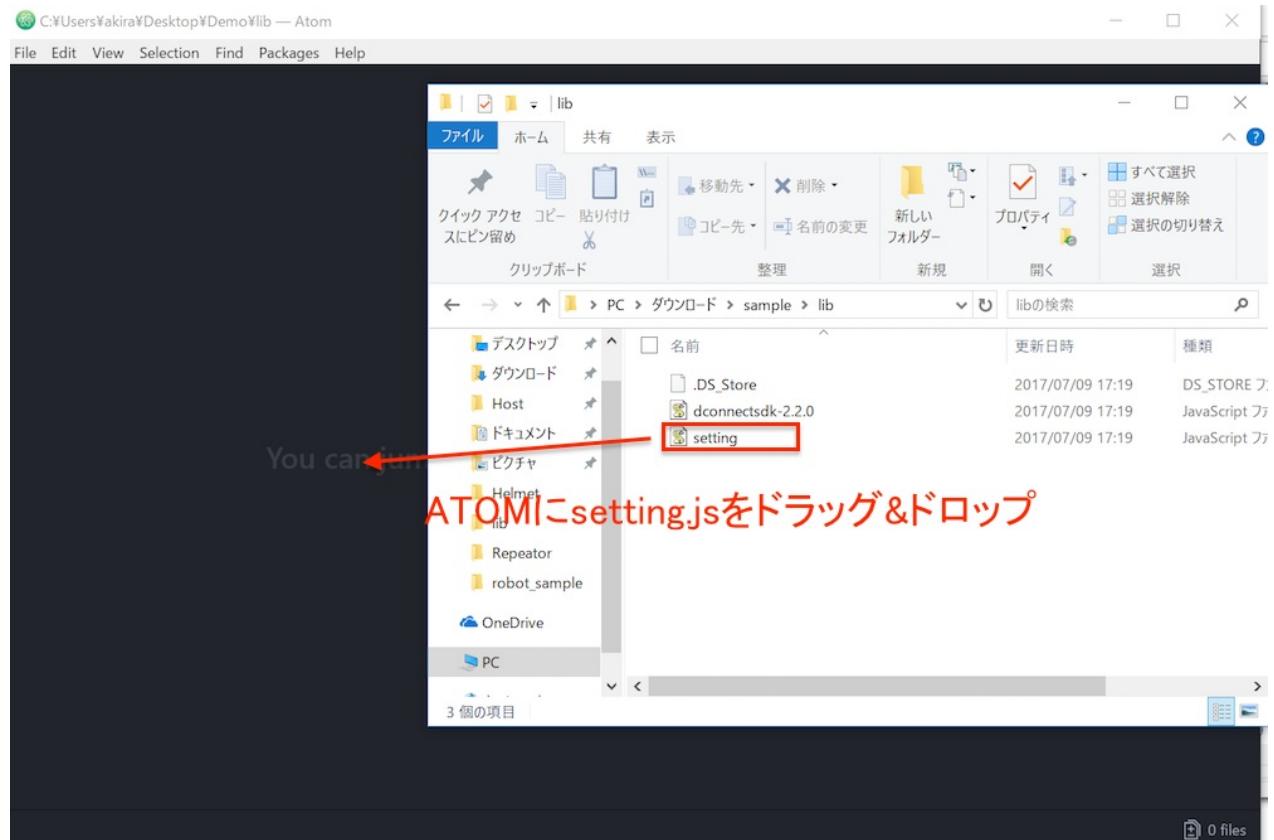


2.4 ATOMのインストール



ATOMで/lib/setting.jsを読み込む

setting.jsを選択し、ATOMエディタの編集画面にドラッグ&ドロップすると、ファイルの中身が読み込まれます。



2.4 ATOMのインストール

The screenshot shows the Atom code editor interface. The title bar reads "setting.js — C:\Users\akira\Downloads\sample\lib — Atom". The menu bar includes File, Edit, View, Selection, Find, Packages, and Help. The main editor area displays the following JavaScript code:

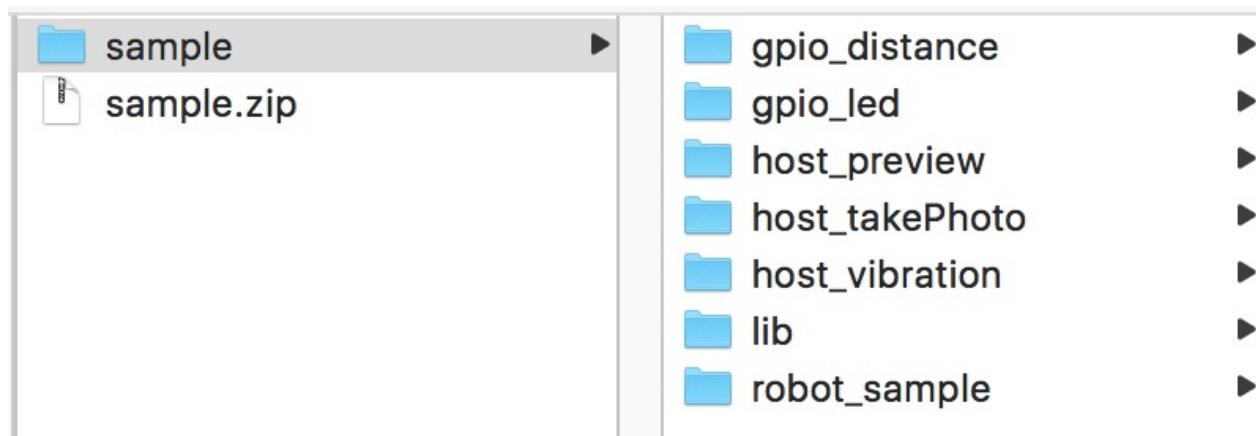
```
1 var gpioId = "gpio_service_id.f9f2e1cc69fe68eda65f4dac5a3f4.localhost.deviceconnect.org";
2 var hostId = "Host.ebc9aec2354491f929dd4b25abccb6.localhost.deviceconnect.org";
3 var mouseId = "mouse_service_id.f9f2e1cc69fe68eda65f4dac5a3f4.localhost.deviceconnect.org";
4 var ip = "172.19.1.1";
5 var port = "4035";
6 
```

The status bar at the bottom shows the file path "C:\Users\akira\Downloads\sample\lib\setting.js" and line number "6:1". It also indicates the encoding as "CRLF", the character set as "UTF-8", the language as "JavaScript", and the number of files as "0 files".

2.5 setting.jsの設定

設定する項目

<http://www.fabo.io/deviceconnect/js/sample.zip>をダウンロードし、解凍します。



sample/lib/setting.jsをSublime textで開き、設定値を自分の環境に書き直します。

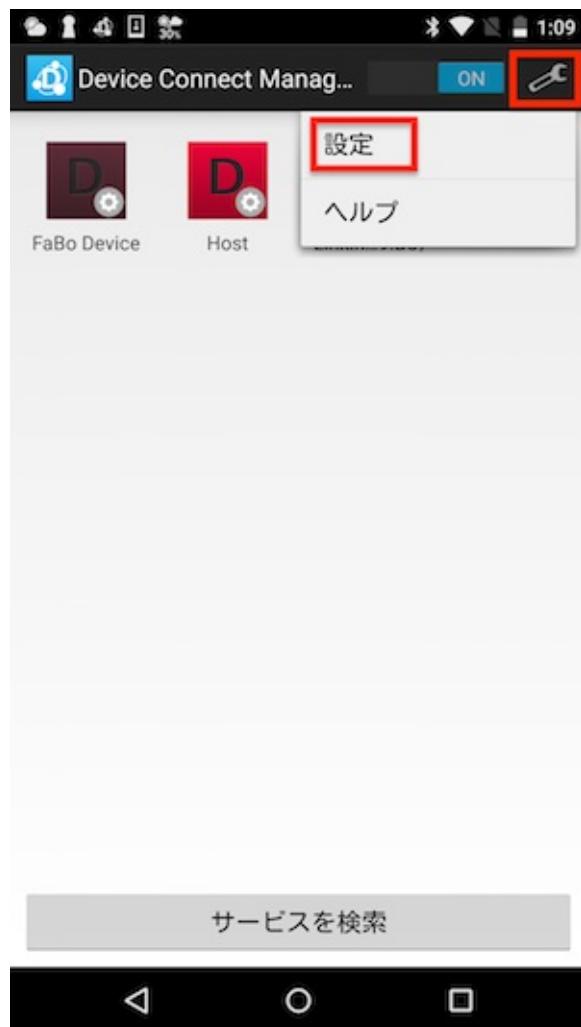
```
var ip = "192.168.0.46";
var port = "4035";
var gpioId = "gpio_service_id.4de8d7e836faab7ad1da5a7ea7737963.localhost.deviceconnect.org";
var hostId = "Host.ebc9a9ec2354491f929dd4b25abccb6.localhost.deviceconnect.org";
var mouseId = "mouse_service_id.4de8d7e836faab7ad1da5a7ea7737963.localhost.deviceconnect.org";
```

2.5 setting.jsの設定

Web名	URL
ip	DeviceWebAPI Managerが起動しているデバイスのIPアドレス(IPとPortの確認 の項目で取得方法は解説)
port	DeviceWebAPI Managerのport番号(IPとPortの確認 の項目で取得方法は解説)
gpioId	Gpio Device PluginのServiceId(ServiceIdの取得 の項目で取得方法は解説)
hostId	Host Device PluginのServiceId(ServiceIdの取得 の項目で取得方法は解説)
mouseId	Robot Mouse Device Plugin ServiceId(ServiceIdの取得 の項目で取得方法は解説)

の5つになります。

IPとPortの確認



2.5 setting.jsの設定



[全プラグインを再起動する]項目の下に表示されているのが、IPアドレスとPort番号になります。これをメモしておきます。

ServiceIdの取得

DeviceWebAPIでは、各機能にアクセスするためにServiceIdが必要となります。ServiceIdの一覧は、/gotapi/serviceDiscoveryで取得する事が可能です。

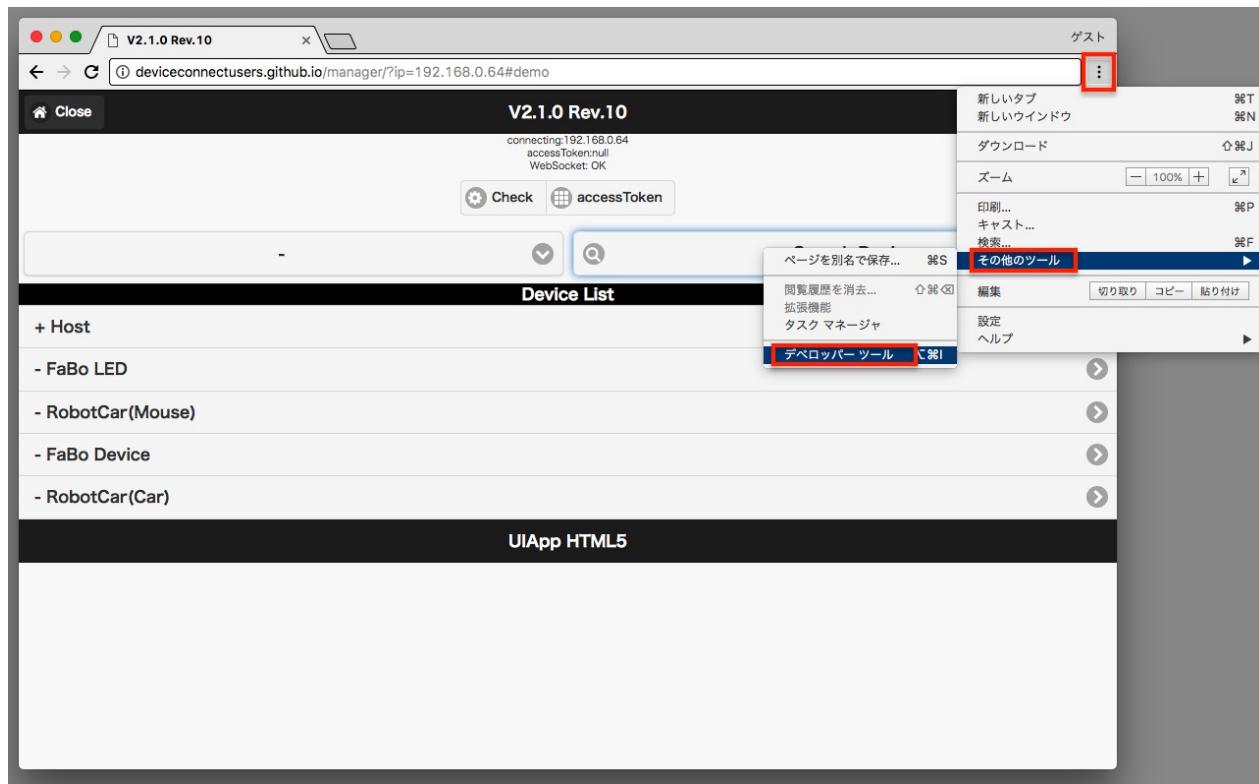
各機能のServiceIdは、Checkツールを用いておこないます。

<http://www.fabo.io/tool/>

gpioId , hostId , mouseId をコピーし、/lib/setting.jsに反映し、保存します。

2.6 Developer Tool

HTML5アプリの開発では、Chromeに内包されているDeveloper Toolを用いてデバックします。



2.6 Developer Tool

The screenshot shows the developer tools interface for V2.1.0 Rev.10. The Network tab is selected. A message at the top says "Recording network activity...". Below it, instructions say "Perform a request or hit ⌘ R to record the reload." The main pane displays a table of network requests:

URL	Type	Total B
/script_root/close	JS	3
/query_u-brndle	JS	2
..	HTML	2
https://devlop.../CS...	JS	1
idevite-google-b	CSS	1
/rs-AZTHYHWE	JS	1
/obs-gst-loadC...	JS	1
https://.../index	JS	1
resThmMy-wd-tp	CSS	1
https://.../wp.../js	JS	1

This screenshot is identical to the one above, but the "Host" item in the Device List is highlighted with a red box. The rest of the interface and the network activity table are the same.

2.6 Developer Tool

V2.1.0 Rev.10 ゲスト deviceconnectusers.github.io/manager/?ip=192.168.0.64#demo

- mediaStreamRecording
- deviceOrientation
- canvas
- setting
- proximity
- geolocation
- touch
- file
- serviceInformation
- battery
- vibration
- notification
- mediaPlayer
- keyEvent
- light**

UIApp HTML5
javascript:searchProfile('Host.ecb9a9ec2354491f929dd4b25abccb6.localhost.deviceconnect.org', 'light')

Elements Console Sources Network Performance 14 ノード

Filter: レジекс Hide data URLs

All XHR JS CSS Img Media Font Doc WS Manifest Other	100 ms	200 ms	300 ms	400 ms	500 ms	600 ms
serviceinformation	200	xhr	dconnect	25ms	55ms	400.00 ms 600ms

1 requests | 25.2 KB transferred

Console What's New

Highlights from Chrome 59 update

CSS and JS code coverage

Find unused CSS and JS with the new Coverage drawer.

Full-page screenshots

Take a screenshot of the entire page, from the top of the viewport to the bottom.

Block requests

V2.1.0 Rev.10 ゲスト deviceconnectusers.github.io/manager/?ip=192.168.0.64#demo

100

Color of light

Red: 25

Green: 25

Blue: 25

Flashing of light

500,500,500,500,500,500,500,500

On

Off

StatusChange

Light List

UIApp HTML5

Elements Console Sources Network Performance 14 ノード

Filter: レジекс Hide data URLs

All XHR JS CSS Img Media Font Doc WS Manifest Other	2000 ms	4000 ms	6000 ms	8000 ms	10000 ms	12000 ms	14000 ms
serviceinformation	200	xhr	dconnect	25ms	55ms	10.00 s	1ms
light?serviceld...	200	xhr	dconnect	39ms	18ms		
light?serviceld...	200	xhr	dconnect	32ms	29ms		

http://192.168.0.64:4035/gotap/light?
serviceld=Host.ecb9a9ec2354491f929dd4b25abccb6.localhost.deviceconnect.org&lightId=0&color
=191919&brightness=1

3 requests | 25.9 KB transferred

Console What's New

Highlights from Chrome 59 update

CSS and JS code coverage

Find unused CSS and JS with the new Coverage drawer.

Full-page screenshots

Take a screenshot of the entire page, from the top of the viewport to the bottom.

Block requests

3.1 バイブレーション

バイブレーションを動作させる

操作	Endpoint	メソッド
バイブレーション	http://IP:4035/gotapi/vibration/vibrate? serviceId=#####	PUT

URLによる擬似的なRESTFUL

操作	Endpoint
バイブレーション	http://IP:4035/gotapi/put/vibration/vibrate? serviceId=#####

バイブレーションを動作

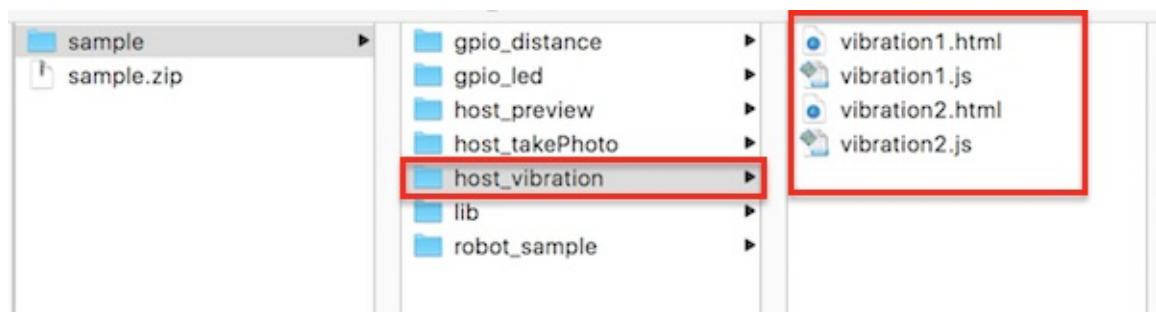
ブラウザからの起動(PUT)

ブラウザからの起動では、Chromeのアドレスバーに下記 Endpointを入力し、Enterで呼び出します。

```
http://192.168.0.15:4035/gotapi/put/vibration/vibrate?  
serviceId=Host.ecb9a9ec2354491f929dd4b25abccb6.localhost.deviceconnect.org
```

HTML/JavaScript

HTML/JavaScriptのサンプルは /sample/host_vibrtion/ フォルダに置かれています。 vibration1.htmlをChrome Browserにドラッグ&ドロップし、実行します。



サンプル1

vibration1.html

```
<html>
  <head>
    <title>Vibration</title>
    <script src="../lib/dconnectsdk-2.2.0.js" type="text/javascript"></script>
    <script src="../lib/setting.js" type="text/javascript"></script>
    <script src="vibration1.js" type="text/javascript"></script>
  </head>
  <body>
    <input type="button" value="バイブルーションを振動" onclick="vibration();"/><br />
  </body>
</html>
```

vibration1.js

```
function vibration() {  
  
    var uri = "http://" + ip + ":" + port + "/gotapi/vibration/vibrat  
e?serviceId=" + hostId;  
    var header = null;  
    var data = null;  
    dConnect.put(uri, header, data, function(json) {  
        console.log(json);  
    }, function(errorCode, errorMessage) {  
        console.log(errorMessage);  
    });  
}
```

バイブレーションを動作

ブラウザからの起動(PUT)

URLの最後に、&をつけてpattern=100,1000,100,100を追加します。

```
http://192.168.0.68:4035/gotapi/put/vibration/vibrate?  
serviceId=Host.ecb9a9ec2354491f929dd4b25abccb6.localhost.deviceconnec  
t.org&pattern=100,1000,100,1000
```

HTML/JavaScript

HTML/JavaScriptのサンプルの実行は /sample/host_vibration/ フォルダにサンプルが置かれています。 vibration1.htmlをChrome Browserにドラッグ&ドロップし、読み出します。

サンプル2

vibration2.html

3.1 バイブレーション

```
<html>
  <head>
    <title>Vibration</title>
    <script src="../../lib/dconnectsdk-2.2.0.js" type="text/javascript"></script>
    <script src="../../lib/setting.js" type="text/javascript"></script>
    <script src="vibration2.js" type="text/javascript"></script>
  </head>
  <body>
    <input type="button" value="バイブルーションを振動" onclick="vibration();"/>  

  </body>
</html>
```

vibration2.js

```
function vibration() {

  var uri = "http://" + ip + ":" + port + "/gotapi/vibration/vibrate?serviceId=" + hostId;
  var header = null;
  var data = "pattern=100,1000,100,1000";
  dConnect.put(uri, header, data, function(json) {
    console.log(json);
  }, function(errorCode, errorMessage) {
    alert(errorMessage);
  });
}
```

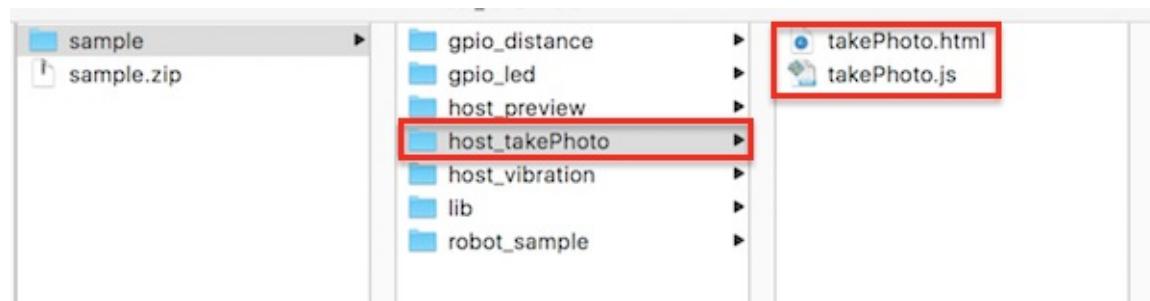
3.2 写真の撮影

写真の撮影

操作	Endpoint	メソッド
写真の撮影	http://IP:4035/gotapi/mediastreamRecording/takePhoto? serviceId=#####	POST

TakePhoto

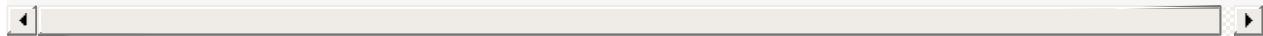
HTML/JavaScriptのサンプルは /sample/host_takePhoto/ フォルダに置かれています。 takePhoto.htmlをChrome Browserにドラッグ&ドロップし、実行します。



takePhoto.html

3.2 写真を撮影する

```
<html>
  <head>
    <title>takePhoto</title>
    <script src="../lib/dconnectsdk-2.2.0.js" type="text/javascript"></script>
    <script src="../lib/setting.js" type="text/javascript"></script>
    <script src="takePhoto.js" type="text/javascript"></script>
  </head>
  <body>
    <input type="button" value="写真を撮影" onclick="takePhoto()" /><br />
    <img id="image" width="500" />
  </body>
</html>
```



takePhoto.js

```
function takePhoto() {
  var imageElement = document.getElementById("image");
  var uri = "http://" + ip + ":" + port + "/gotapi/mediastream
Recording/takePhoto?serviceId=" + hostId;
  dConnect.post(uri , null, null, function(json) {
    if (json.result == 0) {
      var uri = json.uri;
      uri = uri.replace(/localhost/g , ip);
      imageElement.src = uri;
      console.log(uri);
    } else {
      console.log(json.result);
    }

  }, function(errorCode, errorMessage) {
    console.log(errorMessage);
  });
}
```

3.2 写真を撮影する

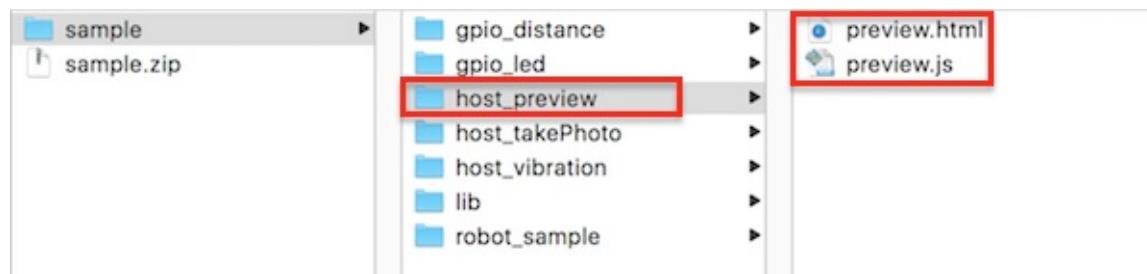
3.3 プレビュー

プレビュー

操作	Endpoint	メソッド
プレビューの開始	http://IP:4035/gotapi/mediastreamRecording/preview? serviceId=#####	PUT
プレビューの終了	http://IP:4035/gotapi/mediastreamRecording/preview? serviceId=#####	DELETE

Preview

HTML/JavaScriptのサンプルは /sample/host_preview/ フォルダに置かれています。 preview.htmlをChrome Browserにドラッグ&ドロップし、実行します。



preview.html

3.3 プレビューを開始

```
<html>
  <head>
    <title>takePhoto</title>
    <script src="../lib/dconnectsdk-2.2.0.js" type="text/javascript"></script>
    <script src="../lib/setting.js" type="text/javascript"></script>
    <script src="controller.js" type="text/javascript"></script>
  </head>
  <body>
    <input type="button" value="プレビューを開始" onclick="preview_start();"/><br />
    <input type="button" value="プレビューを終了" onclick="preview_stop();"/><br />
    <img id="image" width="500"/><br />
  </body>
</html>
```

preview.js

3.3 プレビューを開始

```
function preview_start() {
    var imageElement = document.getElementById("image");
    var uri = "http://" + ip + ":" + port + "/gotapi/mediastream
Recording/preview?serviceId=" + hostId;

    var header = null;
    var data = null;
    dConnect.put(uri, header, data, function(json) {
        if (json.result == 0) {
            var uri = json.uri;
            uri = uri.replace(/localhost/g , ip);
            imageElement.src = uri;
            console.log(uri);
        } else {
            console.log(json.result);
        }
    }, function(errorCode, errorMessage) {
        console.log(errorMessage);
    });
}

function preview_stop() {
    var imageElement = document.getElementById("image");
    var uri = "http://" + ip + ":" + port + "/gotapi/mediastream
Recording/preview?serviceId=" + hostId;

    var header = null;
    var data = null;
    dConnect.delete(uri, header, data, function(json) {
        console.log(json.result);
    }, function(errorCode, errorMessage) {
        console.log(errorMessage);
    });
}
```

4.1 Robot Carへの接続

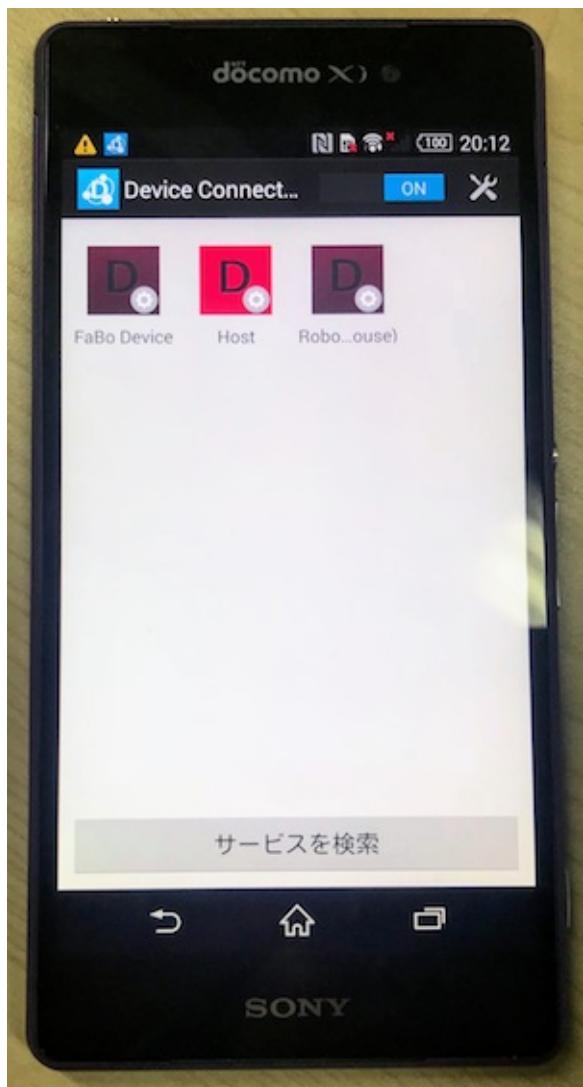
DeviceWebAPI Managerの起動

DeviceWebAPI Managerを起動します。



FaBo Device, Robo...ouse)のアイコンが非アクティブ状態で表示されているのを確認します。

4.1 Robotカーへの接続



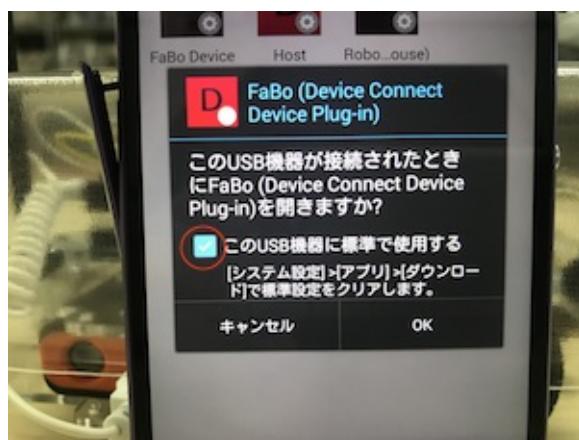
Robot Carへの接続

Robot Carから出ているUSBケーブルをスマートフォンに差し込みます。

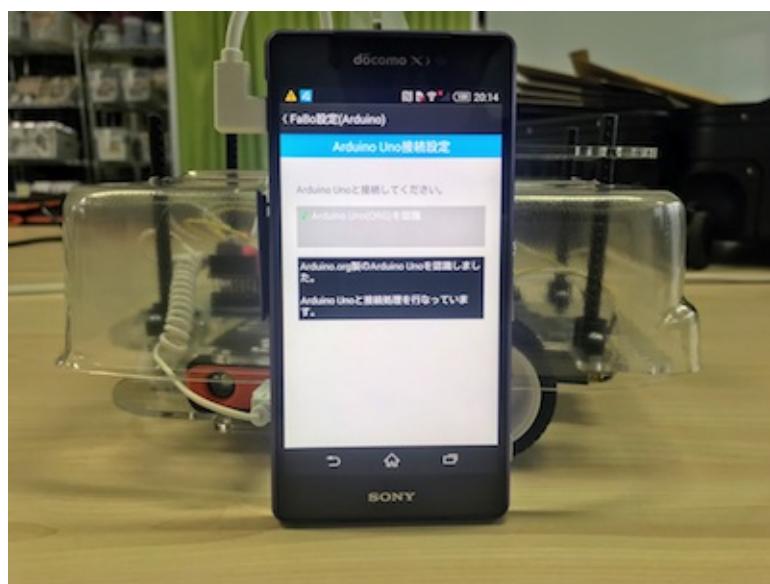


4.1 Robotカーへの接続

USBケーブルを差し込むと下記画面が表示されるので、赤丸の箇所をチェックし、OKを押します。



画面が切り替わり、認識と接続が始まります。(2-3秒程度)



FaBo Device, Robo...ouse)のアイコンがアクティブ状態で表示されれば認識成功です！

4.1 Robotカーへの接続



スマートフォンをRobotCarにマウント

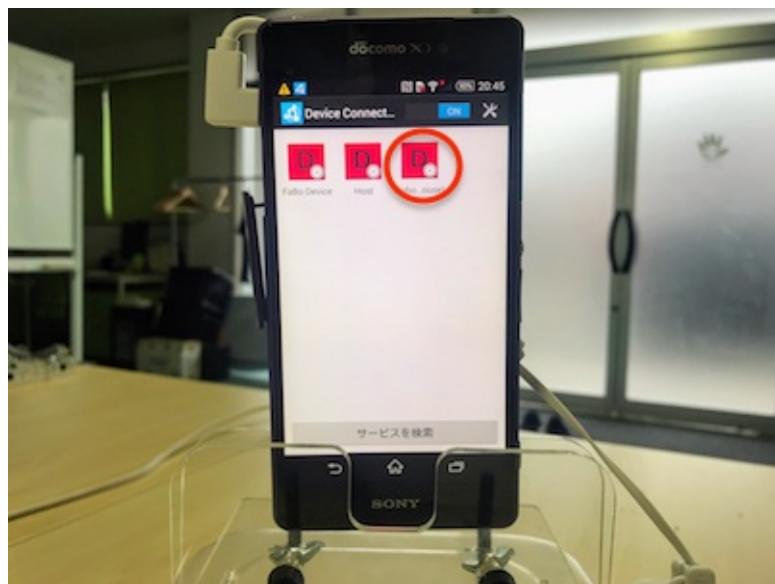
スマートフォンをRobot Carにマウントします。



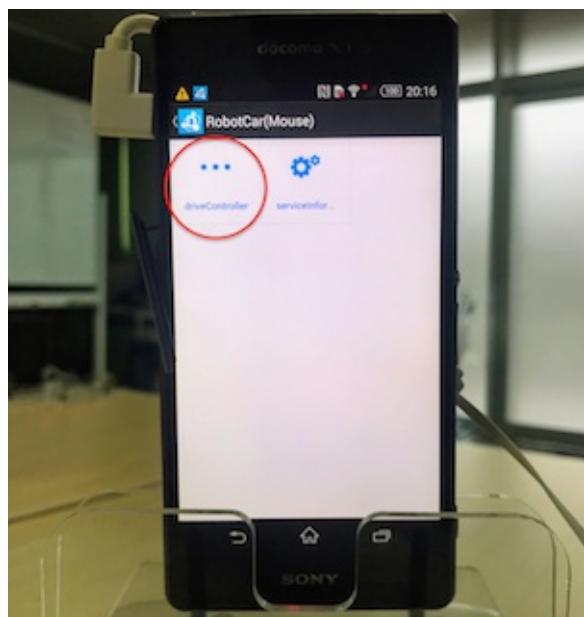
動作確認

Robo...ouse)のアイコンを選択します。

4.1 Robotカーへの接続

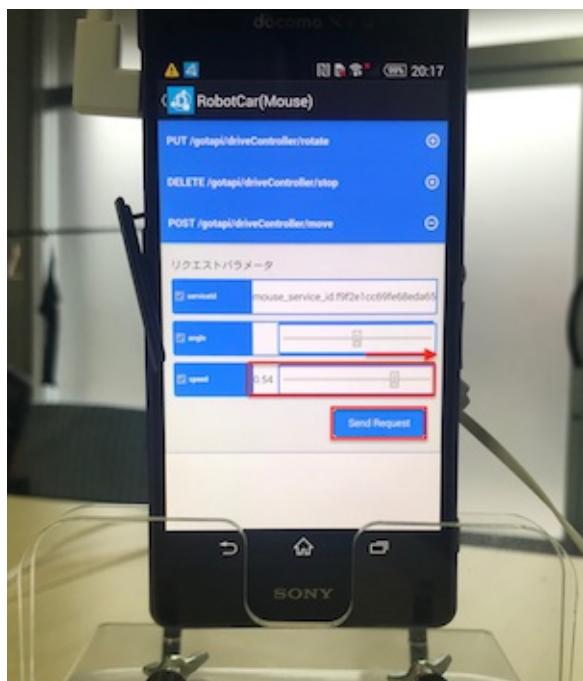
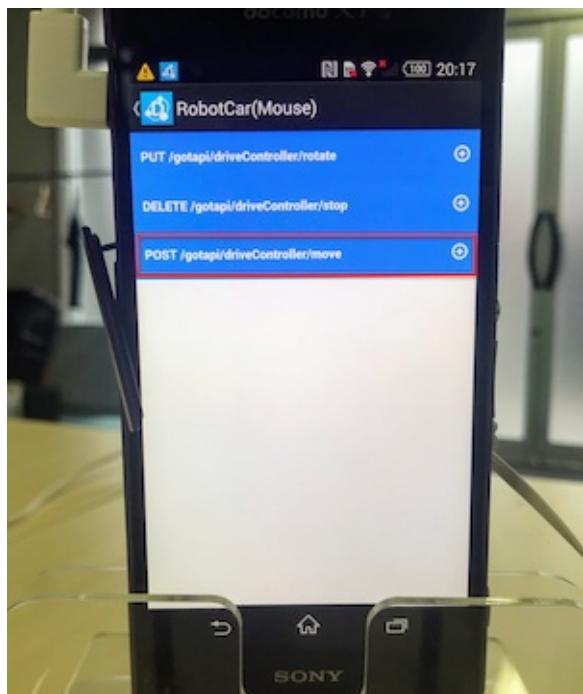


driveControllerのアイコンを選択します。



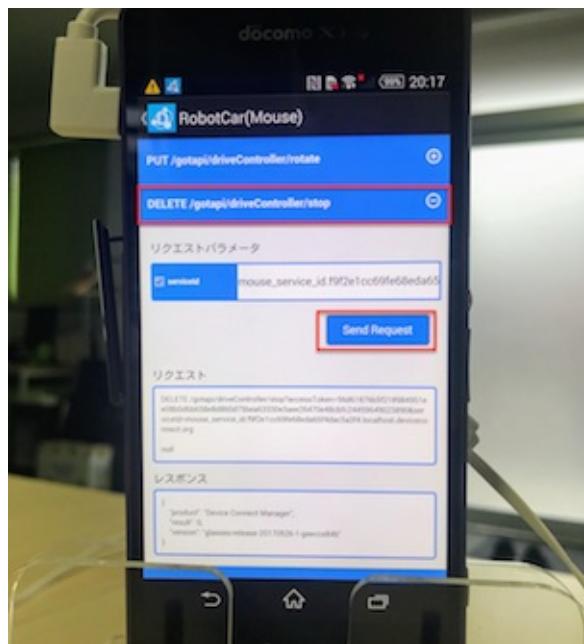
ロボットカーを動作させるには、POST /gotapi/driveController/move の項目を選択し、speedを0.50以上にし、Send Requestを押します。

4.1 Robotカーへの接続



ロボットカーを停止させるには、DELETE /gotapi/driveController/stop の項目を選択し、Send Requestを押します。

4.1 Robotカーへの接続



4.2 RobotCarの操作(アドレスバー)

Robotの操作

操作	Endpoint	メソッド
前進 (speed=1)	http://IP:4035/gotapi/driveController/move? serviceId=####&speed=1&angle=0	POST
後進 (speed=-1)	http://IP:4035/gotapi/driveController/move? serviceId=####&speed=-1&angle=0	POST
停止 (speed=0)	http://IP:4035/gotapi/driveController/move? serviceId=####&speed=0&angle=0	POST
回転 (angle=360)	http://IP:4035/gotapi/driveController/move? serviceId=####&speed=1&angle=360	POST
回転 (angle=-360)	http://IP:4035/gotapi/driveController/move? serviceId=####&speed=1&angle=-360	POST

URLによる擬似的なRESTFUL

操作	Endpoint
前進 (speed=1)	http://IP:4035/gotapi/post/driveController/move? serviceId=####&speed=1&angle=0
後進 (speed=-1)	http://IP:4035/gotapi/post/driveController/move? serviceId=####&speed=-1&angle=0
停止 (speed=0)	http://IP:4035/gotapi/post/driveController/move? serviceId=####&speed=0&angle=0
回転 (angle=360)	http://IP:4035/gotapi/post/driveController/move? serviceId=####&speed=1&angle=360
回転 (angle=-360)	http://IP:4035/gotapi/post/driveController/move? serviceId=####&speed=1&angle=-360

※この時serviceIdにはFaBo PluginのIDを指定します

引数

4.2 Robot操作の基礎

変数名	範囲
speed	-1～1
angle	-360～360

4.3 RobotCarの操作(JavaScript)

RESTFUL

操作	Endpoint	メソッド
前進 (speed=1)	http://IP:4035/gotapi/driveController/move? serviceId=####&speed=1&angle=0	POST
後進 (speed=-1)	http://IP:4035/gotapi/driveController/move? serviceId=####&speed=-1&angle=0	POST
停止 (speed=0)	http://IP:4035/gotapi/driveController/move? serviceId=####&speed=0&angle=0	POST
回転 (angle=360)	http://IP:4035/gotapi/driveController/move? serviceId=####&speed=1&angle=360	POST
回転 (angle=-360)	http://IP:4035/gotapi/driveController/move? serviceId=####&speed=1&angle=-360	POST

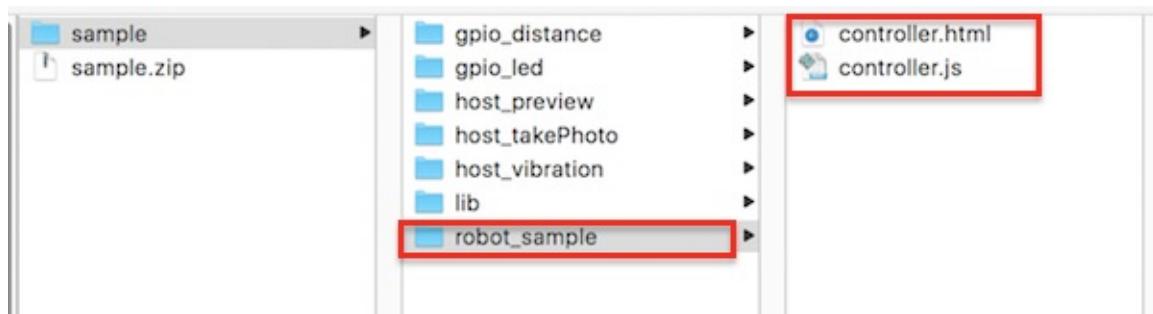
URLによる擬似的なRESTFUL

操作	Endpoint
前進 (speed=1)	http://IP:4035/gotapi/post/driveController/move? serviceId=####&speed=1&angle=0
後進 (speed=-1)	http://IP:4035/gotapi/post/driveController/move? serviceId=####&speed=-1&angle=0
停止 (speed=0)	http://IP:4035/gotapi/post/driveController/move? serviceId=####&speed=0&angle=0
回転 (angle=360)	http://IP:4035/gotapi/post/driveController/move? serviceId=####&speed=1&angle=360
回転 (angle=-360)	http://IP:4035/gotapi/post/driveController/move? serviceId=####&speed=1&angle=-360

Sample

4.3 Robot操作のサンプル

HTML/JavaScriptのサンプルは /sample/robot_sample/ フォルダに置かれています。 controller.htmlをChrome Browserにドラッグ&ドロップし、実行します。



controller.html

```
<html>
  <head>
    <title>takePhoto</title>
    <script src="../lib/dconnectsdk-2.2.0.js" type="text/javascript"></script>
    <script src="controller.js" type="text/javascript"></script>
  </head>
  <body>
    <input type="button" value="プレビューを開始" onclick="preview_stop();"/><br />
    <input type="button" value="プレビューを終了" onclick="preview_start();"/><br />
    <img id="image" width="500"/><br />
    <input type="button" value="↑" onclick="move(1);"/><br />

    <input type="button" value="■" onclick="move(0);"/><br />

    <input type="button" value="↓" onclick="move(-1);"/><br />
  </>
    <input type="range" min="-360" max="360" step="10" onchange="changeAngle(this.value);">
  </body>
</html>
```

controller.js

4.3 Robot操作のサンプル

```
var angle = 0;
function preview_start() {
    var imageElement = document.getElementById("image");
    var uri = "http://" + ip + ":" + port + "/gotapi/mediastream
Recording/preview?serviceId=" + hostId;

    var header = null;
    var data = null;
    dConnect.put(uri, header, data, function(json) {
        if (json.result == 0) {
            var uri = json.uri;
            uri = uri.replace(/localhost/g , ip);
            imageElement.src = uri;
            console.log(uri);
        } else {
            console.log(json.result);
        }
    }, function(errorCode, errorMessage) {
        console.log(errorMessage);
    });
}

function preview_stop() {
    var imageElement = document.getElementById("image");
    var uri = "http://" + ip + ":" + port + "/gotapi/mediastream
Recording/preview?serviceId=" + hostId;

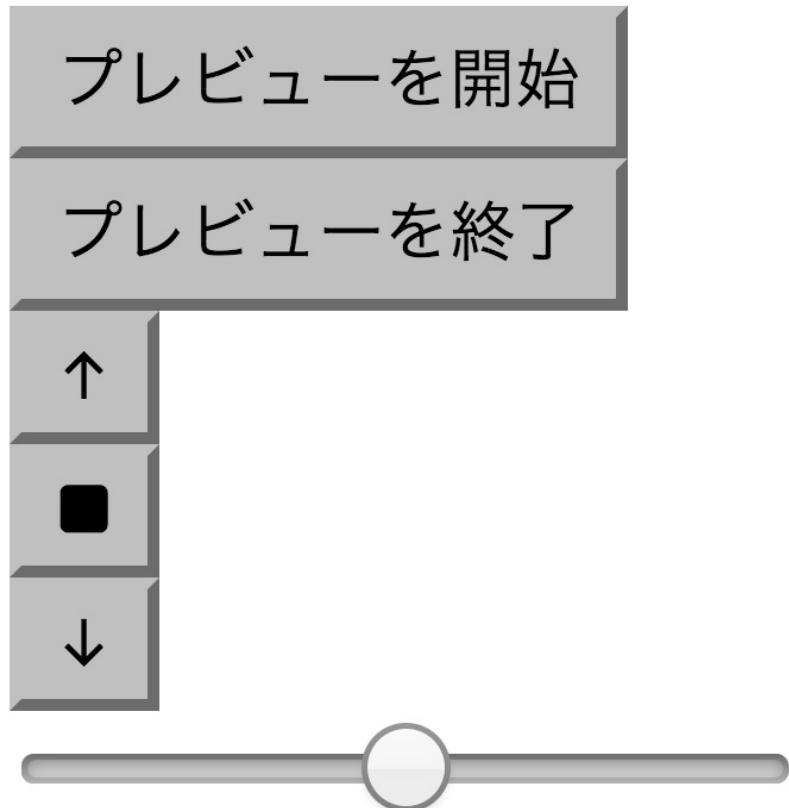
    var header = null;
    var data = null;
    dConnect.delete(uri, header, data, function(json) {
        console.log(json);
    }, function(errorCode, errorMessage) {
        console.log(errorMessage);
    });
}

function changeAngle(mAngle) {
    console.log(mAngle);
```

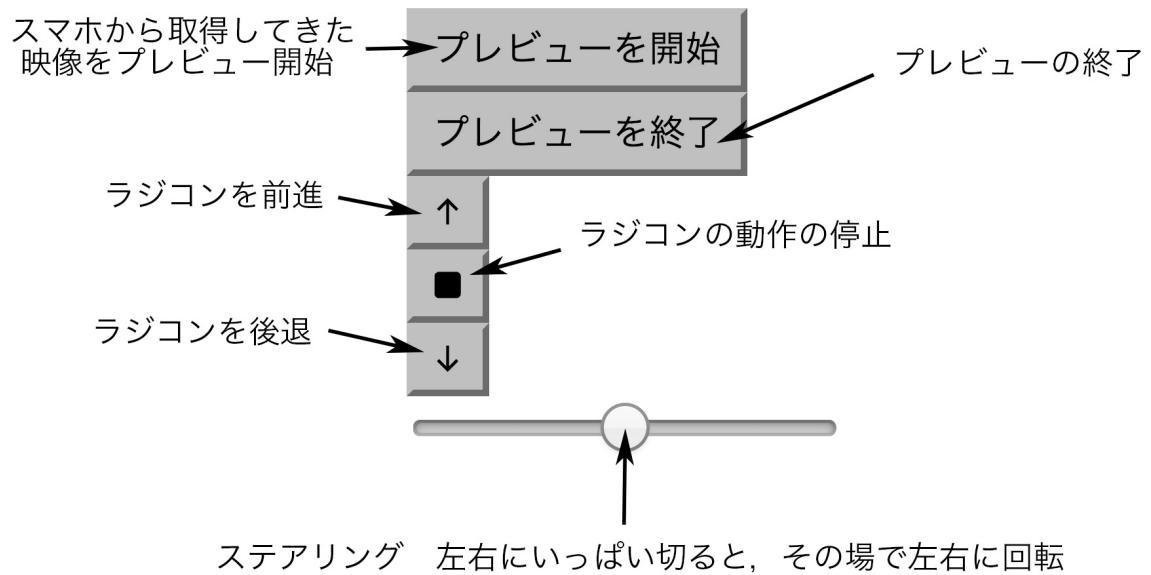
```
    angle = mAngle;  
}  
  
function move(speed) {  
    var uri = "http://" + ip + ":" + port + "/gotapi/driveController/move?serviceId=" + mouseId;  
    var header = null;  
    var data = "speed=" + speed;  
    data += "&angle=" + angle;  
    dConnect.post(uri, header, data, function(json) {  
        console.log(json.result);  
    }, function(errorCode, errorMessage) {  
        console.log(errorMessage);  
    });  
}
```

実際の操作

controller.htmlをブラウザで開くと、以下のような画面が開く。



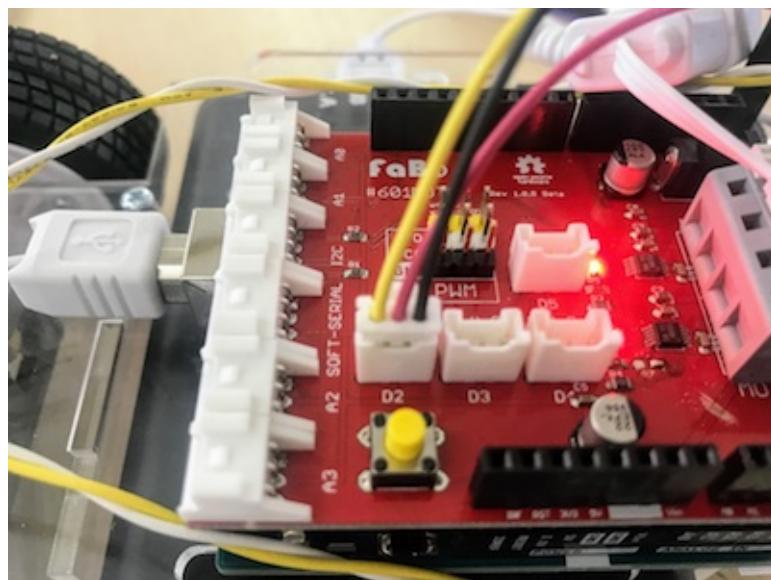
このボタンなどを押すことで、ラジコンカーを実際に操作できる。以下はコントローラーの各ボタンなどの役割である。



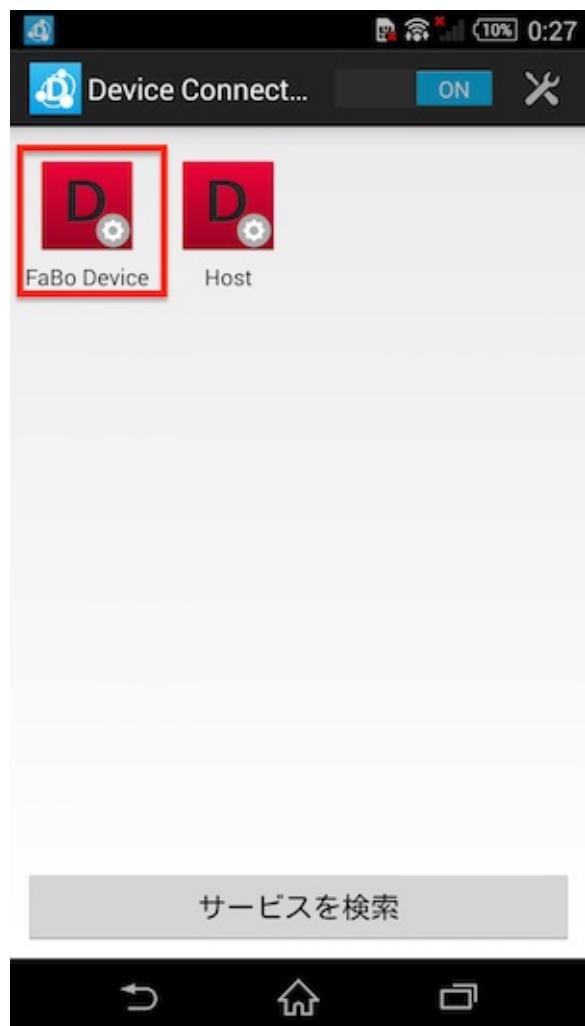
5.1 FaBo Pluginの設定

動作確認

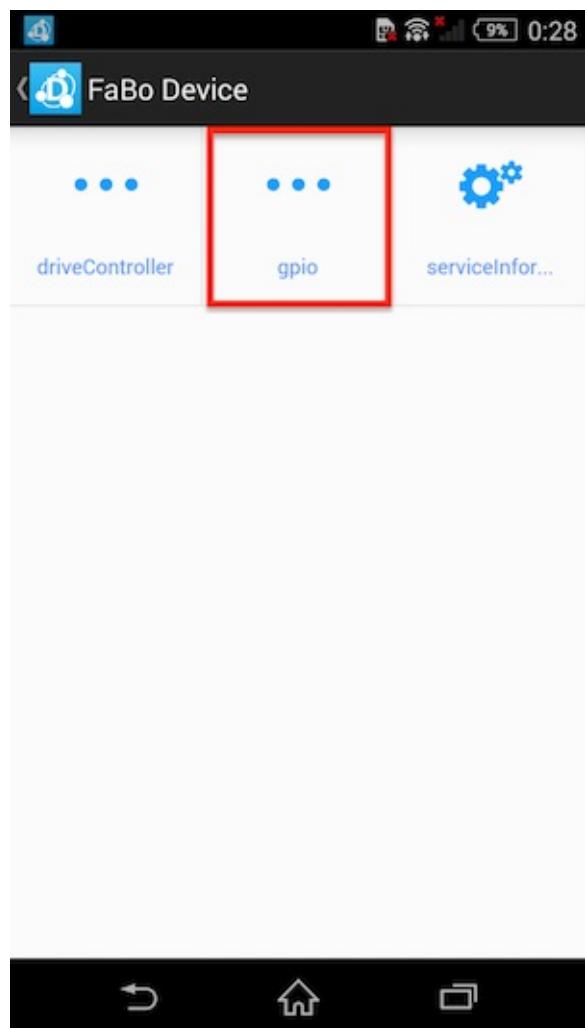
FaBoのLED Brickを以下の写真のようにD2に接続します。 初期状態ではLEDは消えています。



5.1 FaBo Pluginの接続

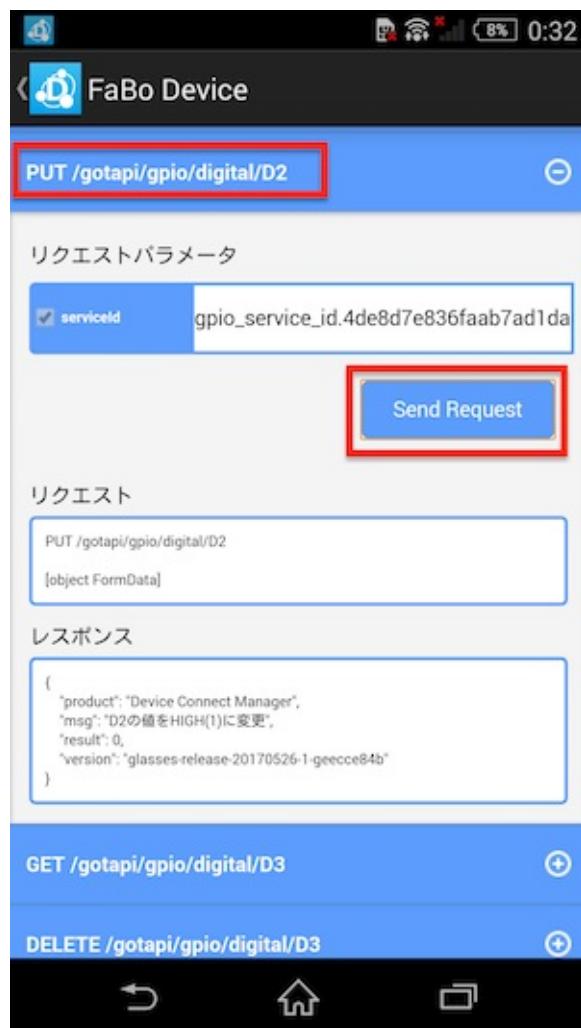


5.1 FaBo Pluginの接続



PUT /gotapi/gpio/digital/D2を選択し、Send Requestをタップします。

5.1 FaBo Pluginの接続



すると以下の写真のようにLEDが点灯します。



次にDELETE /gotapi/gpio/digital/D2を選択し、Send Requestをタップします。

5.1 FaBo Pluginの接続



すると今度は以下の写真のようにLEDが消灯します。



5.2 LEDの操作(アドレスバー)

LEDの操作

操作	Endpoint	メソッド
LEDの状態の変更	http://IP:4035/gotapi/gpio/digital/D2? serviceId#####	POST
LEDの点灯	http://IP:4035/gotapi/gpio/digital/D2? serviceId#####	PUT
LEDの消灯	http://IP:4035/gotapi/gpio/digital/D2? serviceId#####	DELETE
LEDの状態の取得	http://IP:4035/gotapi/gpio/digital/D2? serviceId#####	GET

URLによる擬似的なRESTFUL

操作	Endpoint
LEDの状態の変更	http://IP:4035/gotapi/post/gpio/digital/D2? serviceId#####&value=1
LEDの点灯	http://IP:4035/gotapi/put/gpio/digital/D2? serviceId#####
LEDの消灯	http://IP:4035/gotapi/delete/gpio/digital/D2? serviceId#####
LEDの状態の取得	http://IP:4035/gotapi/gpio/digital/D2? serviceId#####

※この時serviceIdにはFaBo PluginのIDを指定します

LEDを点灯する(PUT)

D2にFaBo LED Brickを接続します。

メモしていたIPアドレスをここから使用します。IPアドレスを自分の環境において呼び出しをおこないます。

5.2 LEDの操作(アドレスバー)

ブラウザーのアドレスバーに、下記Endpointを入力します。

```
http://192.168.0.10:4035/gotapi/put/gpio/digital/D2?  
serviceId=#####
```

LEDを消灯する(DELETE)

ブラウザーのアドレスバーに、下記Endpointを入力します。

```
http://192.168.0.10:4035/gotapi/delete/gpio/digital/D2?  
serviceId=#####
```

LEDの状態を変える(POST)

ブラウザーのアドレスバーに、下記Endpointを入力します。

点灯

```
http://192.168.0.10:4035/gotapi/post/gpio/digital/D2?  
serviceId=#####&value=1
```

消灯

```
http://192.168.0.10:4035/gotapi/post/gpio/digital/D2?  
serviceId=#####&value=0
```

LEDの状態を取得する(GET)

```
http://192.168.0.10:4035/gotapi/gpio/digital/D2?serviceId=#####
```

5.3 LEDの操作(JavaScript)

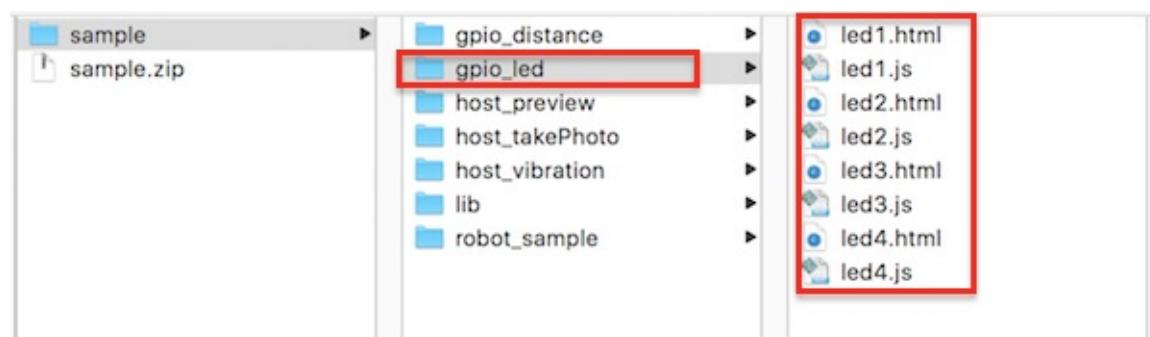
LEDの操作

操作	Endpoint	メソッド
LEDの状態の変更	http://IP:4035/gotapi/gpio/digital/D2? serviceId=#####	POST
LEDの点灯	http://IP:4035/gotapi/gpio/digital/D2? serviceId=#####	PUT
LEDの消灯	http://IP:4035/gotapi/gpio/digital/D2? serviceId=#####	DELETE
LEDの状態の取得	http://IP:4035/gotapi/gpio/digital/D2? serviceId=#####	GET

LEDを点灯する(PUT)

まず、LEDを点灯するサンプルを作成します。LEDの点灯にはPUTメソッドを用います。

HTML/JavaScriptのサンプルは `/sample/gpio_led/` フォルダに置かれています。`led1.html`をChrome Browserにドラッグ&ドロップし、実行します。



`led1.html`

```

<html>
  <head>
    <title>LED</title>
    <script src="../lib/dconnectsdk-2.2.0.js" type="text/javascript"></script>
    <script src="../lib/setting.js" type="text/javascript"></script>
    <script src="led1.js" type="text/javascript"></script>
  </head>
  <body>
    <input type="button" value="LEDを点灯" onclick="LEDOn();"
  /><br />
  </body>
</html>

```



led1.js

```

function LEDOn() {
  var uri = "http://" + ip + ":" + port + "/gotapi/gpio/digital/D2?serviceId=" + gpioId;
  var header = null;
  var data = null;
  dConnect.put(uri, header, function(json) {
    console.log(json);
  }, function(errorCode, errorMessage) {
    console.log(errorMessage);
  });
}

```

LEDを消灯する(DELETE)

まず、LEDを点灯するサンプルを作成します。LEDの点灯にはPUTメソッドを用います。

led2.html

```
<html>
  <head>
    <title>LED</title>
    <script src="dconnectsdk-2.2.0.js" type="text/javascript"></script>
    <script src="../lib/setting.js" type="text/javascript"></script>
    <script src="led2.js" type="text/javascript"></script>
  </head>
  <body>
    <input type="button" value="LEDを点灯" onclick="LEDOn();"
    /><br />
    <input type="button" value="LEDを消灯" onclick="LEDOff();"
    /><br />
  </body>
</html>
```

led2.js

```
function LEDOn() {
    var uri = "http://" + ip + ":" + port + "/gotapi/gpio/digital/D2?serviceId=" + gpioId;
    var header = null;
    var data = null;
    dConnect.put(uri, header, function(json) {
        console.log(json);
    }, function(errorCode, errorMessage) {
        console.log(errorMessage);
    });
}

function LEDOff() {
    var uri = "http://" + ip + ":" + port + "/gotapi/gpio/digital/D2?serviceId=" + gpioId;
    var header = null;
    var data = null;
    dConnect.delete(uri, header, function(json) {
        console.log(json);
    }, function(errorCode, errorMessage) {
        console.log(errorMessage);
    });
}
```

LEDの状態を変える(POST)

次に、LEDの状態を変えてみます。

led3.html

```
<html>
  <head>
    <title>LED</title>
    <script src="dconnectsdk-2.2.0.js" type="text/javascript"></script>
    <script src="../lib/setting.js" type="text/javascript"></script>
    <script src="led3.js" type="text/javascript"></script>
  </head>
  <body>
    <input type="button" value="LEDを点灯" onclick="LEDOn();"
    /><br />
    <input type="button" value="LEDを消灯" onclick="LEDOff();"
    /><br />
    <input type="button" value="LEDを点灯" onclick="LEDChange
    (1);"/><br />
    <input type="button" value="LEDを消灯" onclick="LEDChange
    (0);"/><br />
  </body>
</html>
```

led3.js

```
function LEDOn() {
    var uri = "http://" + ip + ":" + port + "/gotapi/gpio/digital/D2?serviceId=" + gpioId;
    var header = null;
    var data = null;
    dConnect.put(uri, header, function(json) {
        console.log(json);
    }, function(errorCode, errorMessage) {
        console.log(errorMessage);
    });
}

function LEDOff() {
    var uri = "http://" + ip + ":" + port + "/gotapi/gpio/digital/D2?serviceId=" + gpioId;
    var header = null;
    var data = null;
    dConnect.delete(uri, header, function(json) {
        console.log(json);
    }, function(errorCode, errorMessage) {
        console.log(errorMessage);
    });
}

function LEDChange(value) {
    var uri = "http://" + ip + ":" + port + "/gotapi/gpio/digital/D2?serviceId=" + gpioId;
    var header = null;
    var data = "value=" + value;
    dConnect.post(uri, header, data, function(json) {
        console.log(json);
    }, function(errorCode, errorMessage) {
        console.log(errorMessage);
    });
}
```

LEDの状態を取得する(GET)

次に、LEDの状態を取得してみます。

led4.html

```
<html>
  <head>
    <title>LED</title>
    <script src="dconnectsdk-2.2.0.js" type="text/javascript"></script>
    <script src="../lib/setting.js" type="text/javascript"></script>
    <script src="led4.js" type="text/javascript"></script>
  </head>
  <body>
    <input type="button" value="LEDを点灯" onclick="LEDOn();"
/><br />
    <input type="button" value="LEDを消灯" onclick="LEDOff();"
/><br />
    <input type="button" value="LEDを点灯" onclick="LEDChange
(1);"/><br />
    <input type="button" value="LEDを消灯" onclick="LEDChange
(0);"/><br />
    <input type="button" value="LEDの状態を取得" onclick="LEDS
tatus();"/><br />
  </body>
</html>
```

led4.js

```
function LEDOn() {
  var uri = "http://" + ip + ":" + port + "/gotapi/gpio/digital/D2?serviceId=" + gpioId;
  var header = null;
  var data = null;
  dConnect.put(uri, header, function(json) {
    console.log(json);
  }, function(errorCode, errorMessage) {
    console.log(errorMessage);
  });
}
```

```
}

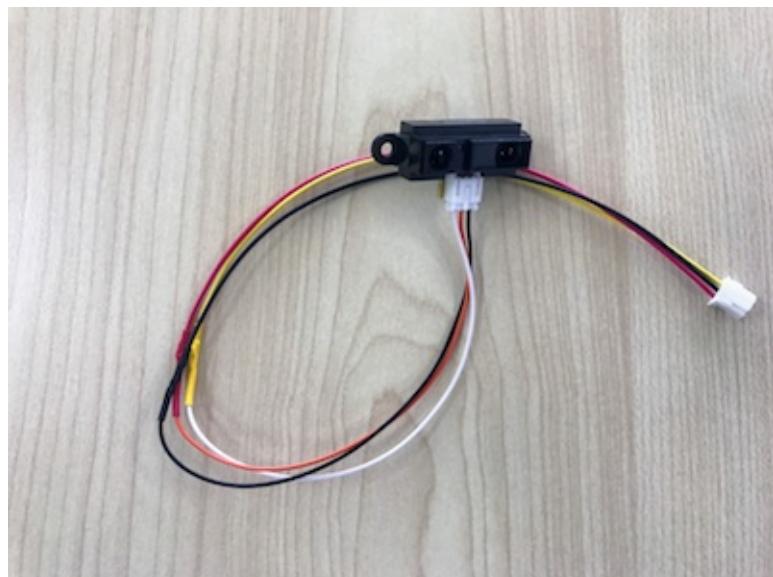
function LEDOff() {
    var uri = "http://" + ip + ":" + port + "/gotapi/gpio/digital/D2?serviceId=" + gpioId;
    var header = null;
    var data = null;
    dConnect.delete(uri, header, function(json) {
        console.log(json);
    }, function(errorCode, errorMessage) {
        console.log(errorMessage);
    });
}

function LEDChange(value) {
    var uri = "http://" + ip + ":" + port + "/gotapi/gpio/digital/D2?serviceId=" + gpioId;
    var header = null;
    var data = "value=" + value;
    dConnect.post(uri, header, data, function(json) {
        console.log(json);
    }, function(errorCode, errorMessage) {
        console.log(errorMessage);
    });
}

function LEDStatus() {
    var uri = "http://" + ip + ":" + port + "/gotapi/gpio/digital/D2?serviceId=" + gpioId;
    var header = null;
    dConnect.get(uri, header, function(json) {
        console.log(json);
    }, function(errorCode, errorMessage) {
        console.log(errorMessage);
    });
}
```

5.4 距離を取得

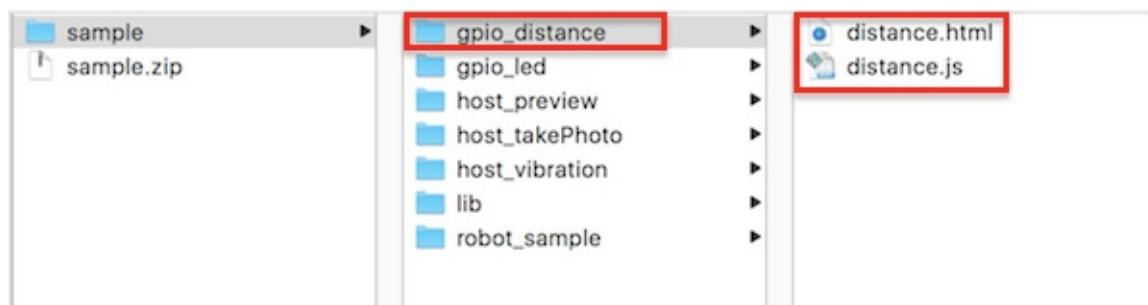
距離センサーをA0のPINに差し込みます。



距離を取得

HTML/JavaScriptのサンプルは `/sample/gpio_distance/` フォルダに置かれています。 `distance.html`をChrome Browserにドラッグ&ドロップし、実行します。

5.4 距離を取得



distance.html

```
<html>
    <head>
        <title>distance</title>
        <script src="../lib/dconnectsdk-2.2.0.js" type="text/javascript"></script>
        <script src="../lib/setting.js" type="text/javascript"></script>
        <script src="distance.js" type="text/javascript"></script>
    >
    </head>
    <body>
        <input type="button" value="距離を取得" onclick="distance();"/><br />
        <div id="value"></div>
    </body>
</html>
```

distance.js

```
function arduino_map(x, in_min, in_max, out_min, out_max){  
    return (x - in_min) * (out_max - out_min) / (in_max - in_min)  
} + out_min;  
  
}  
  
function distance() {  
    var valueElement = document.getElementById("value");  
    var uri = "http://" + ip + ":" + port + "/gotapi/gpio/analog  
/A0?serviceId=" + gpioId;  
    console.log(uri);  
    dConnect.get(uri, null, function(json) {  
        console.log(json);  
        if (json.result == 0) {  
            var value = json.value;  
            volt = arduino_map(value, 0, 1023, 0, 5000);  
            distanceValue = arduino_map(volt, 3200, 500, 5, 80);  
            valueElement.innerHTML = "<h1>" + distanceValue + "<  
/h1>";  
        }  
    }, function(errorCode, errorMessage) {  
        console.log(errorMessage);  
    });  
}
```

5.5 温度を取得

温度センサーをA0のPINに差し込みます。



温度を取得

HTML/JavaScriptのサンプルは /sample/gpio_temperature/ フォルダに置かれています。 temperature.htmlをChrome Browserにドラッグ&ドロップし、実行します。

temperature.html

5.5 溫度を取得

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>temperature</title>
    <script src="../lib/dconnectsdk-2.2.0.js" type="text/javascript"></script>
    <script src="../lib/setting.js" type="text/javascript"></script>
    <script src="temperature.js" type="text/javascript"></script>
  </head>
  <body>
    <input type="button" value="温度を取得" onclick="temperature();"/>  

    <div id="value"></div>
  </body>
</html>
```

temperature.js

```
function arduino_map(x, in_min, in_max, out_min, out_max){  
    return (x - in_min) * (out_max - out_min) / (in_max - in_min)  
} + out_min;  
  
function temperature() {  
    var valueElement = document.getElementById("value");  
    var uri = "http://" + ip + ":" + port + "/gotapi/gpio/analog  
/A0?serviceId=" + faboId;  
    console.log(uri);  
    dConnect.get(uri, null, function(json) {  
        console.log(json);  
        if (json.result == 0) {  
            var value = json.value;  
            console.log(value);  
            volt = arduino_map(value, 0, 1023, 0, 5000);  
            temperatureValue = arduino_map(volt, 300, 1600, -30,  
100);  
            temperatureValue = Math.round(temperatureValue*10)/10  
;  
            valueElement.innerHTML = "<h1>" + temperatureValue +  
"</h1>";  
        }  
    }, function(errorCode, errorMessage) {  
        console.log(errorMessage);  
    });  
}
```

5.6 照度を取得

照度センサーをA0のピンに差し込みます。



照度を取得

light.html

5.6 照度を取得

```
<html>
  <head>
    <title>light</title>
    <script src="../lib/dconnectsdk-2.2.0.js" type="text/javascript"></script>
    <script src="light.js" type="text/javascript"></script>
    <script src="../lib/setting.js" type="text/javascript"></script>
  </head>
  <body>
    <input type="button" value="照度を取得" onclick="light();"
    /><br />
    <div id="value"></div>
  </body>
</html>
```

light.js

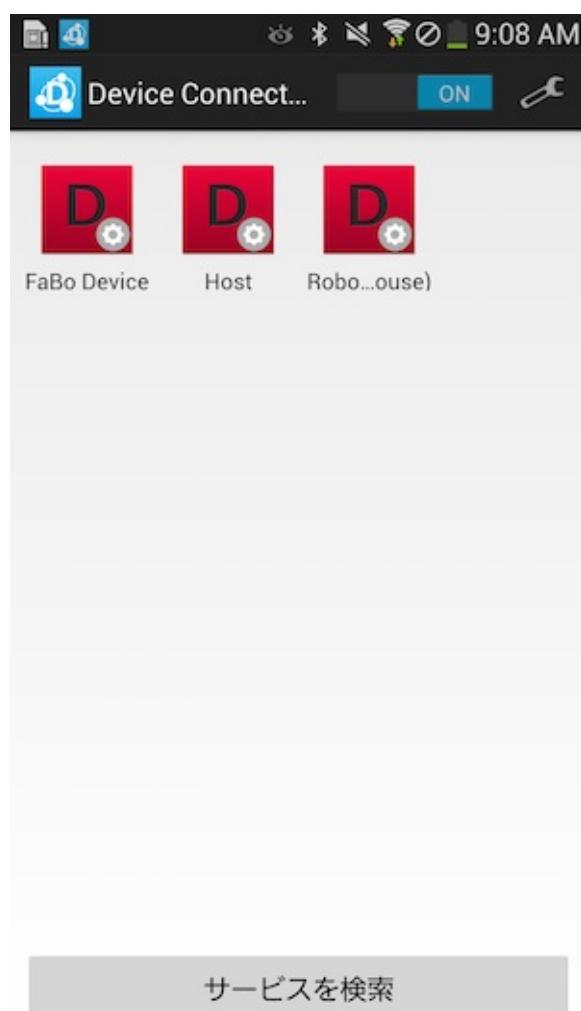
```
function arduino_map(x, in_min, in_max, out_min, out_max){  
    return (x - in_min) * (out_max - out_min) / (in_max - in_min  
) + out_min  
}  
  
function light() {  
    var valueElement = document.getElementById("value");  
    var uri = "http://" + ip + ":" + port + "/gotapi/gpio/analog  
/A0?serviceId=" + faboId;  
    console.log(uri);  
    dConnect.get(uri, null, function(json) {  
        console.log(json);  
        if (json.result == 0) {  
            var value = json.value;  
            console.log(value);  
            valueElement.innerHTML = "<h1>" + value + "</h1>";  
        }  
    }, function(errorCode, errorMessage) {  
        console.log(errorMessage);  
    });  
}
```

6.1 3軸加速度センサー

現在の設定の確認

設定画面から、現在有効なデバイスの一覧を表示します。

- FaBo Device
- Host
- Robot(Mouse)



の3デバイスが表示されています。

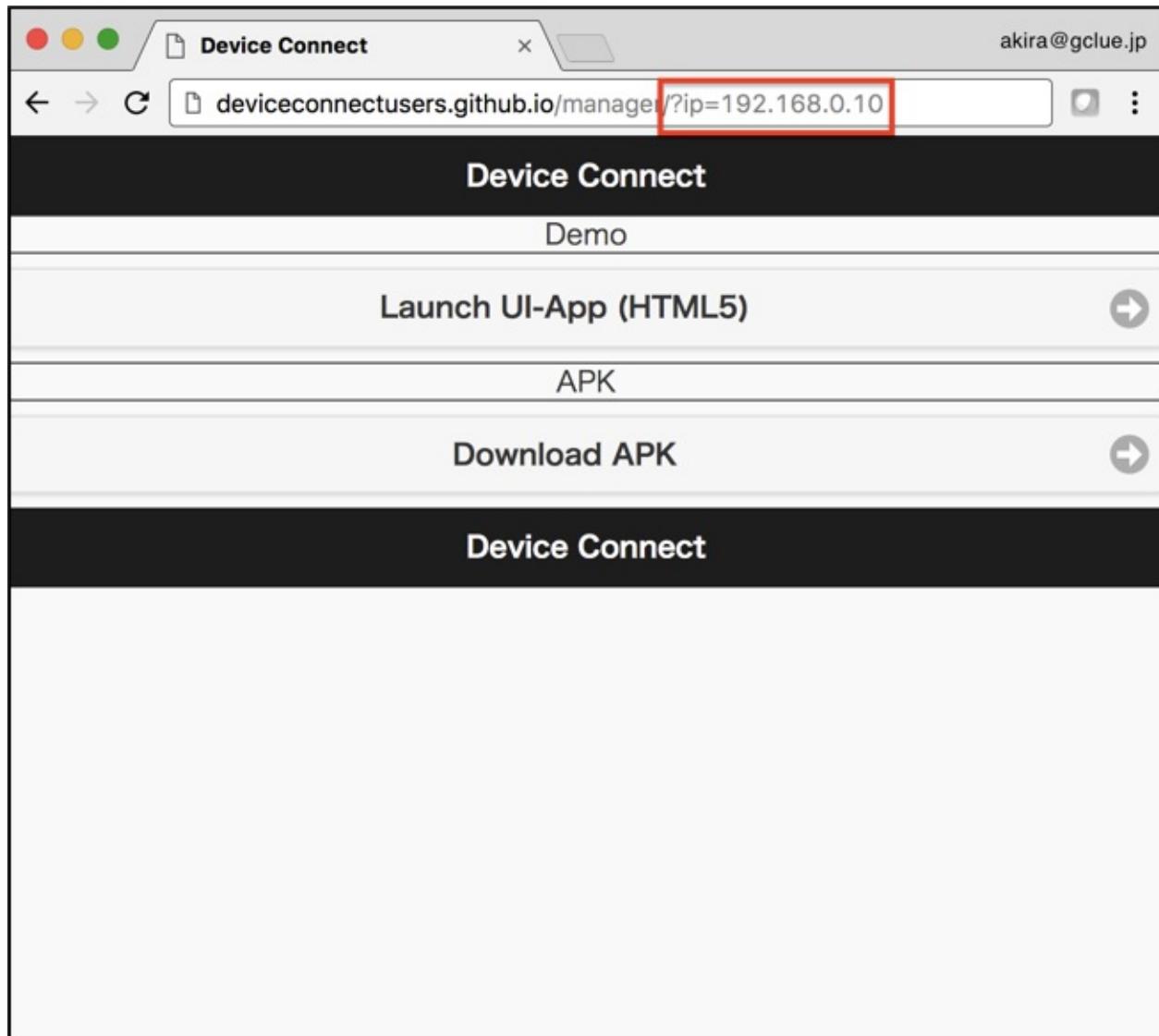
仮想デバイスの作成

仮想デバイスの作成は、

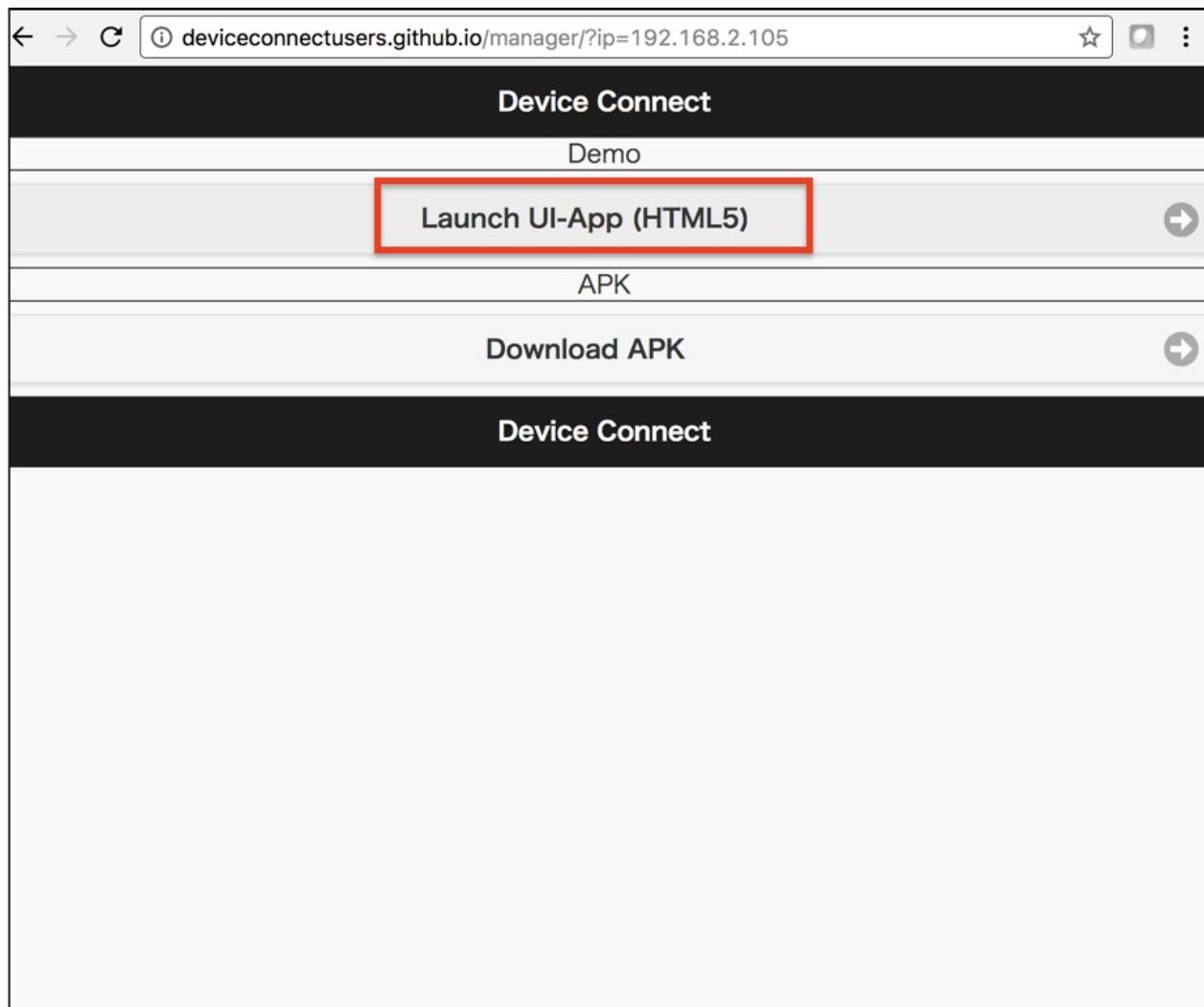
6.1 3軸加速度センサーの登録

<http://deviceconnectusers.github.io/manager/>

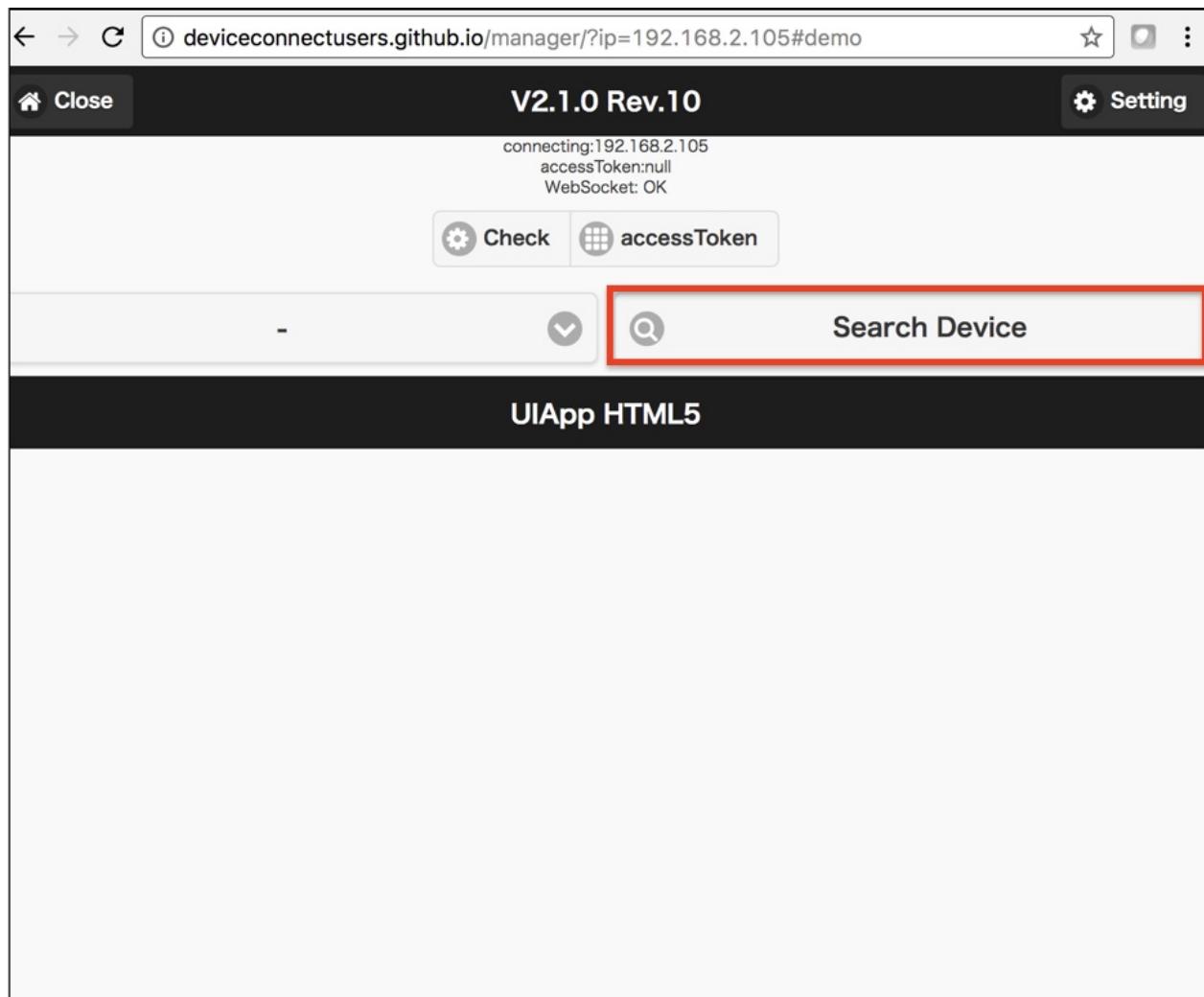
を用いておこないます。現在のIPアドレスを確認し、 をURLの最後に追加して、接続します。



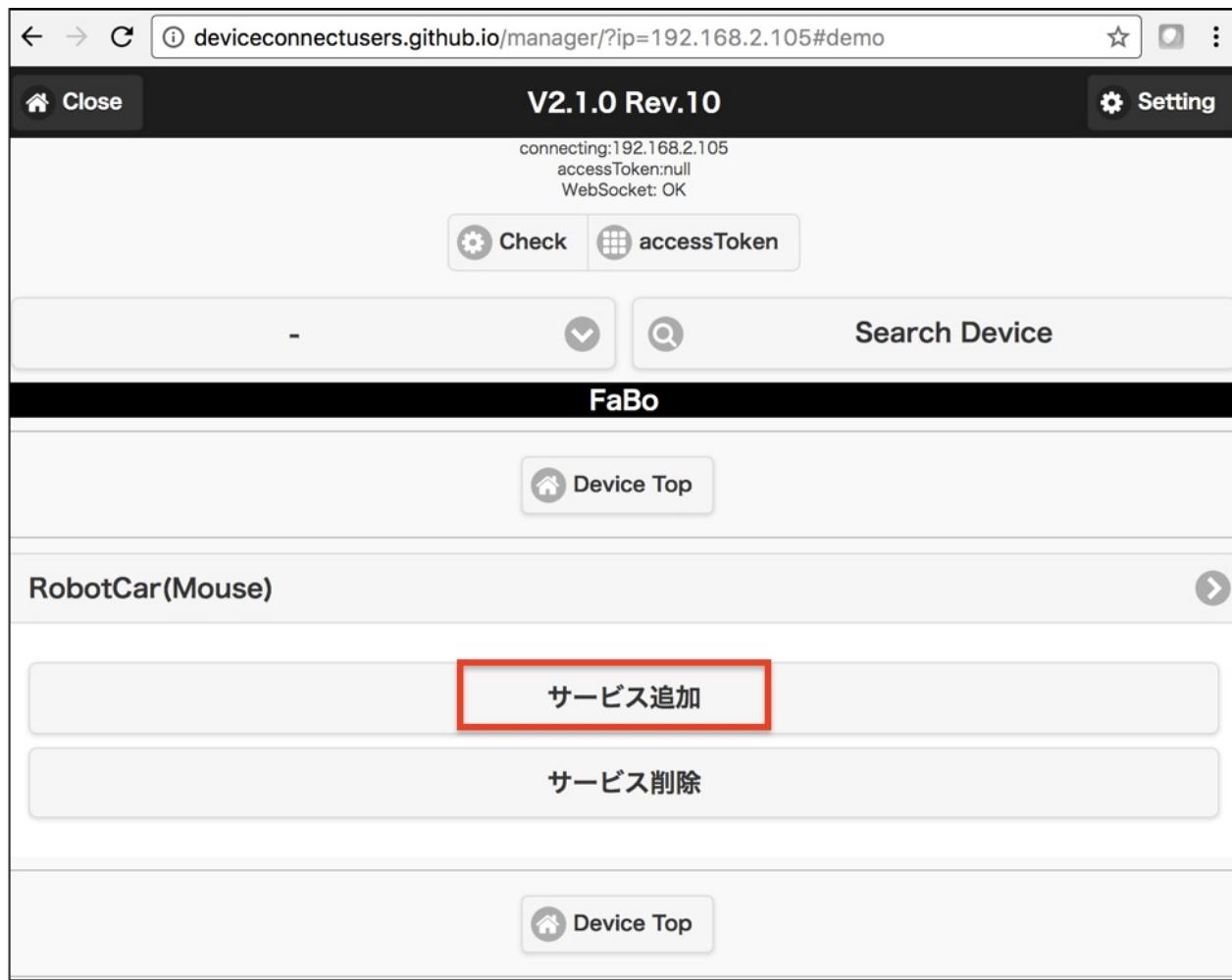
6.1 3軸加速度センサーの登録



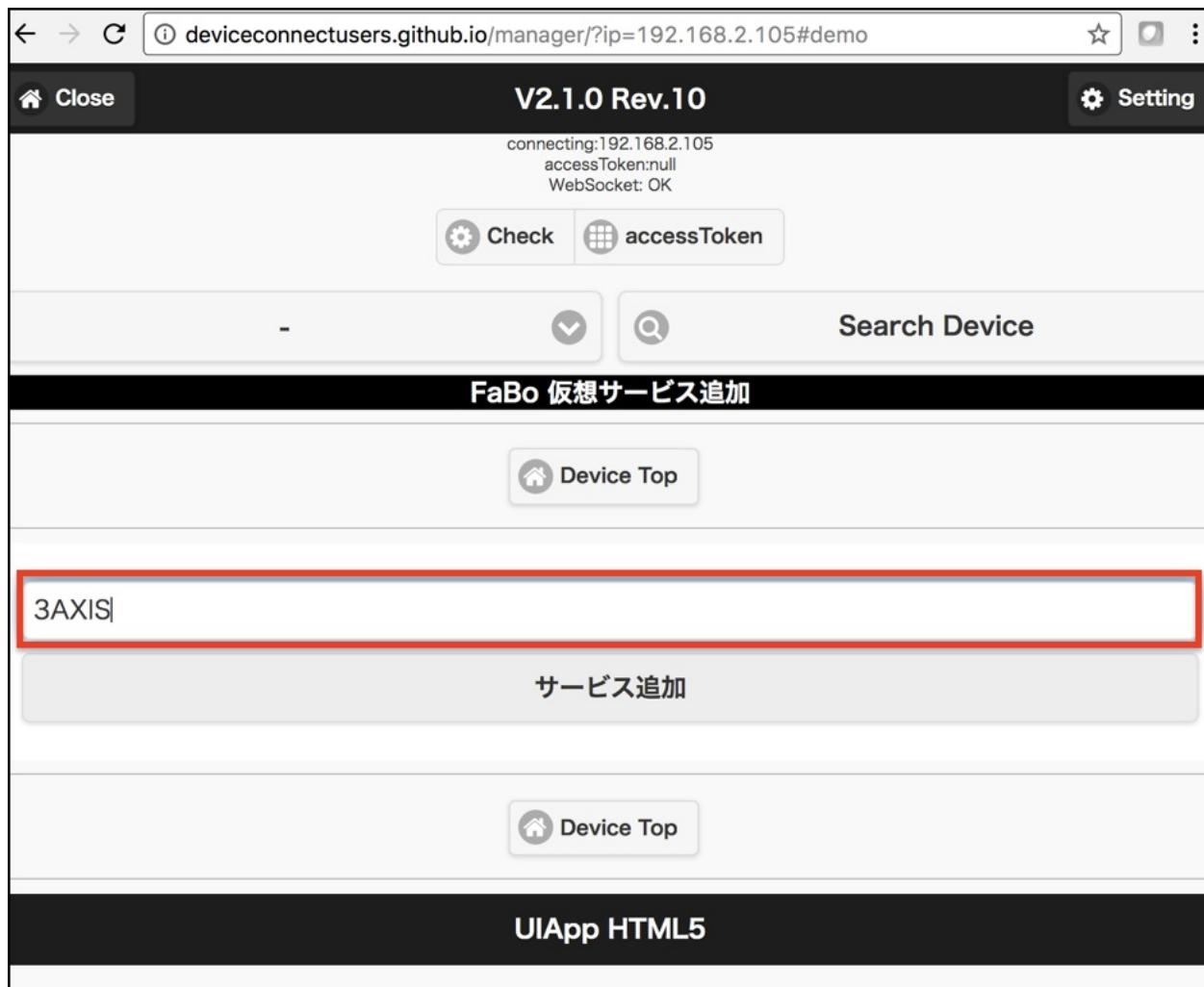
6.1 3軸加速度センサーの登録



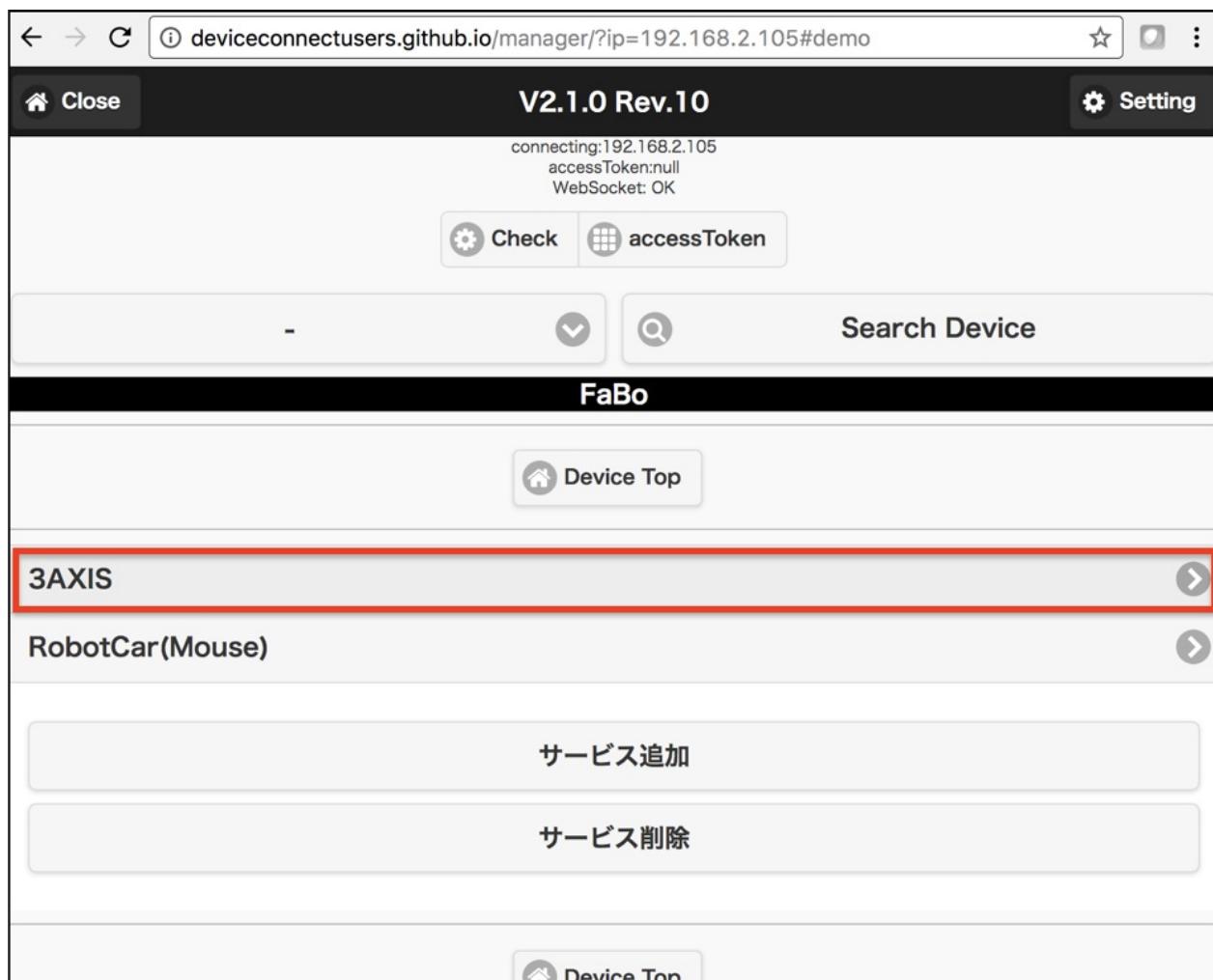
6.1 3軸加速度センサーの登録



6.1 3軸加速度センサーの登録



6.1 3軸加速度センサーの登録



6.1 3軸加速度センサーの登録



6.1 3軸加速度センサーの登録

The screenshot shows a web-based device management interface titled "V2.1.0 Rev.10". At the top, there are status messages: "connecting:192.168.2.105", "accessToken:null", and "WebSocket: OK". Below this are two buttons: "Check" and "accessToken". There are also "Search Device" and "Device Top" buttons.

The main area displays a list of registered devices:

- Light(GPIO) #101
- Temperature(GPIO) #108
- Vibration(GPIO) #105
- Illuminace(GPIO) #109
- KeyEvent(GPIO) #103
- Humidity(GPIO) #115
- Proximity(GPIO) #116
- DriveController(RobotCar)
- DriveController(MouseCar)
- DeviceOrientation(3AXIS) #201

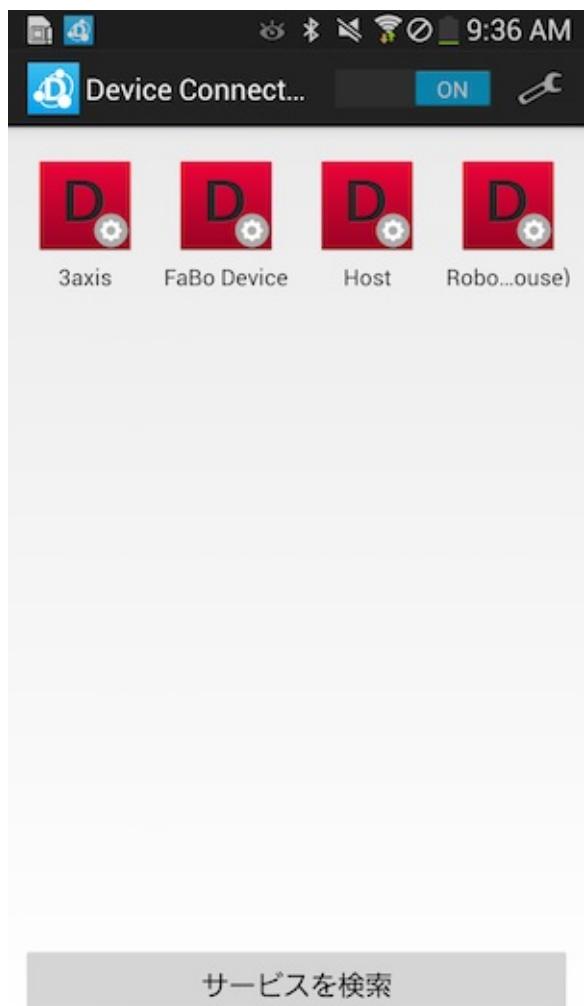
The last item, "DeviceOrientation(3AXIS) #201", is highlighted with a red border.

6.1 3軸加速度センサーの登録



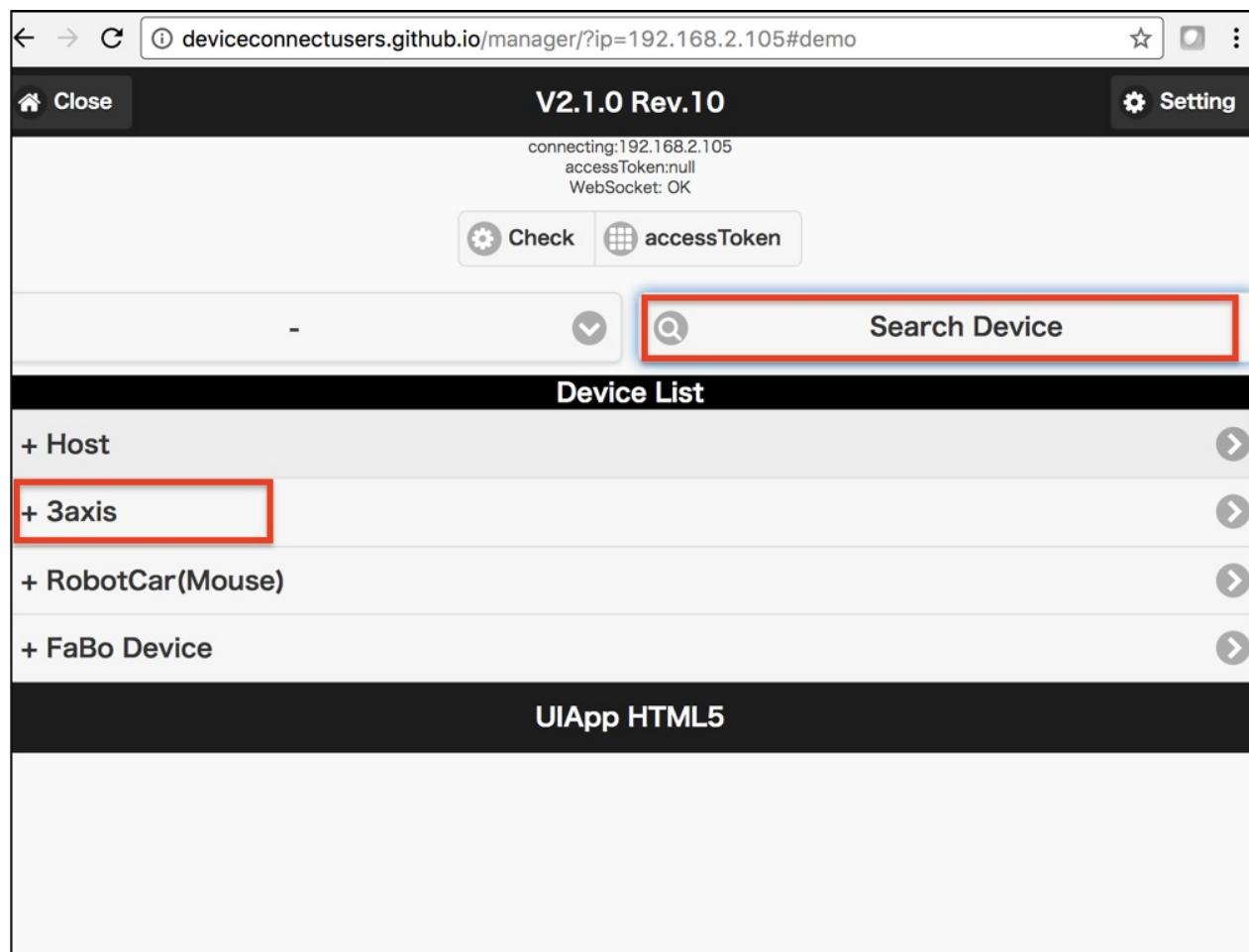
仮想デバイスが追加されると、3axis(自分で指定した名前の)デバイスが追加されます。

6.1 3軸加速度センサーの登録



6.2 3軸加速度センサーの取得

3軸加速度センサーの取得



6.2 3軸加速度センサーの取得

The screenshot shows a web-based management interface for a device. At the top, there's a header bar with a back arrow, forward arrow, refresh button, and a URL bar containing the address `deviceconnectusers.github.io/manager/?ip=192.168.2.105#demo`. To the right of the URL are icons for star, square, and three dots.

The main content area has a dark header labeled "V2.1.0 Rev.10". Below it, a status message reads:
connecting:192.168.2.105
accessToken:null
WebSocket: OK

Below the status are two buttons: "Check" and "accessToken".

On the left side, there's a dropdown menu with a minus sign and a search bar with a magnifying glass icon. To the right of the search bar is the text "Search Device".

The central part of the interface is titled "Profile List". It contains a list of service profiles:

- serviceInformation
- deviceOrientation

The "deviceOrientation" item is highlighted with a red rectangular border. To the right of each profile name is a circular arrow icon.

At the bottom of the list is a dark header labeled "UIApp HTML5".

6.2 3軸加速度センサーの取得

The screenshot shows a web-based management interface for a device. At the top, it displays the URL `deviceconnectusers.github.io/manager/?ip=192.168.2.105#demo`. The title bar indicates the version **V2.1.0 Rev.10**. Below the title, status information includes `connecting:192.168.2.105`, `accessToken:null`, and `WebSocket: OK`. There are two buttons: **Check** and **accessToken**.

On the right side of the header is a **Setting** icon.

In the center, there is a search bar labeled **Search Device** with a dropdown arrow icon.

The main content area has a header **DeviceOrientation Profile(Event)**. Below it is a button labeled **Device Top**.

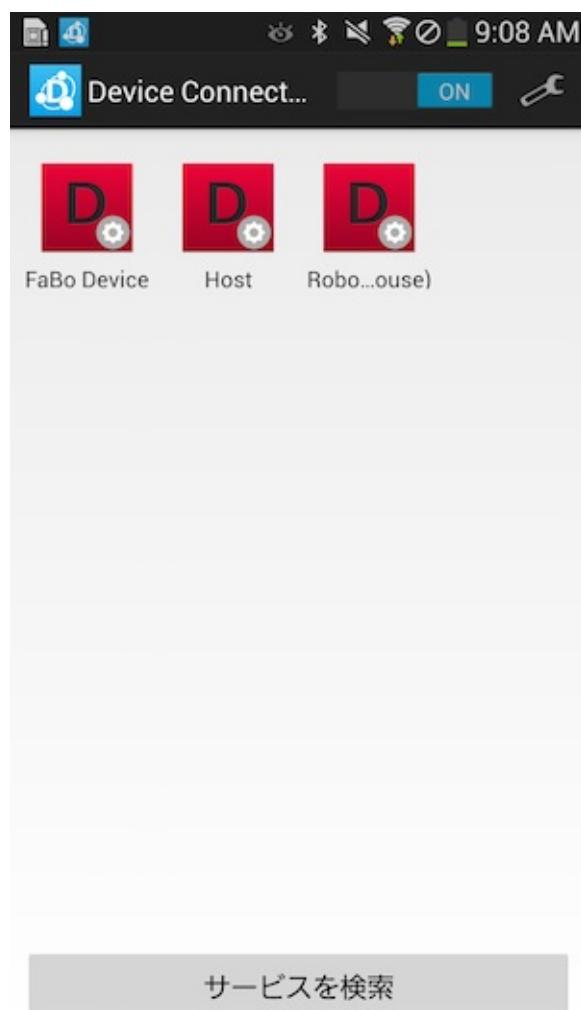
The interface includes several sections for different sensors:

- Get** (with a magnifying glass icon)
- interval** (with a magnifying glass icon)
- Accelerometer** (with a magnifying glass icon): This section contains a prominent **Register** button, which is highlighted with a red rectangular border. To its right is an **Unregister** button.
- Accelerometer (Gravity)** (with a magnifying glass icon)

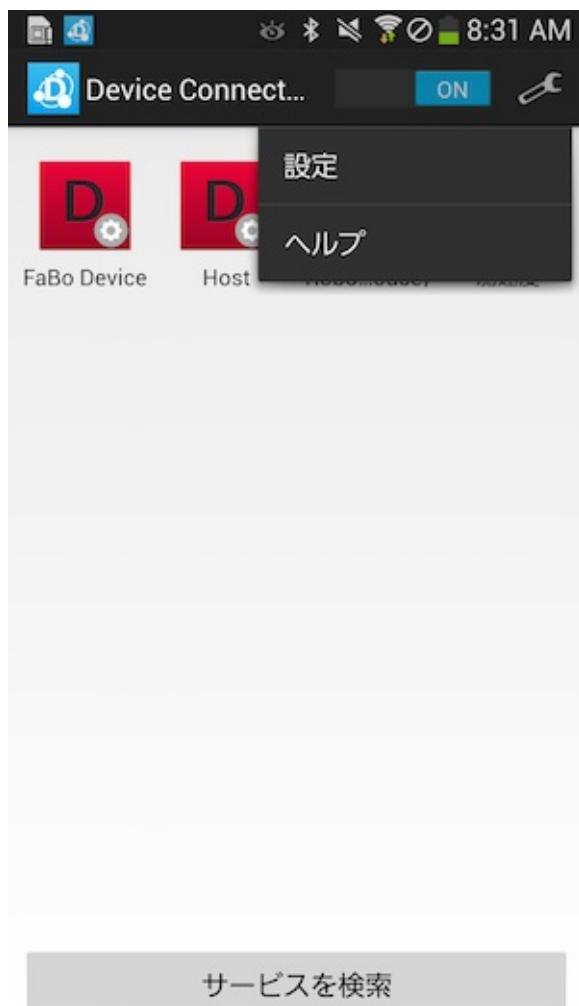
7.1 認証ダイアログの表示/非表示設定

F認証ダイアログの表示/非表示設定

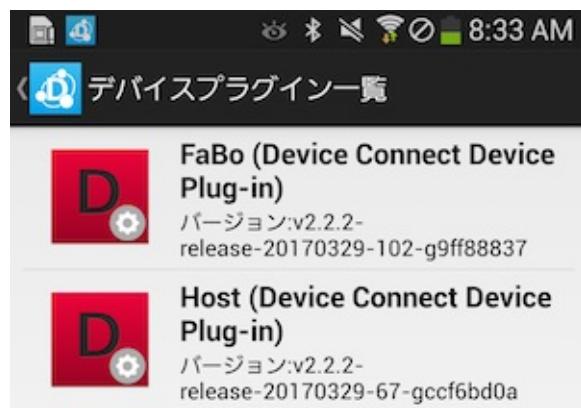
認証ダイアログの表示/非表示設定は、設定画面でおこないます。



7.1 認証ダイアログの表示/非表示設定



7.1 認証ダイアログの表示/非表示設定

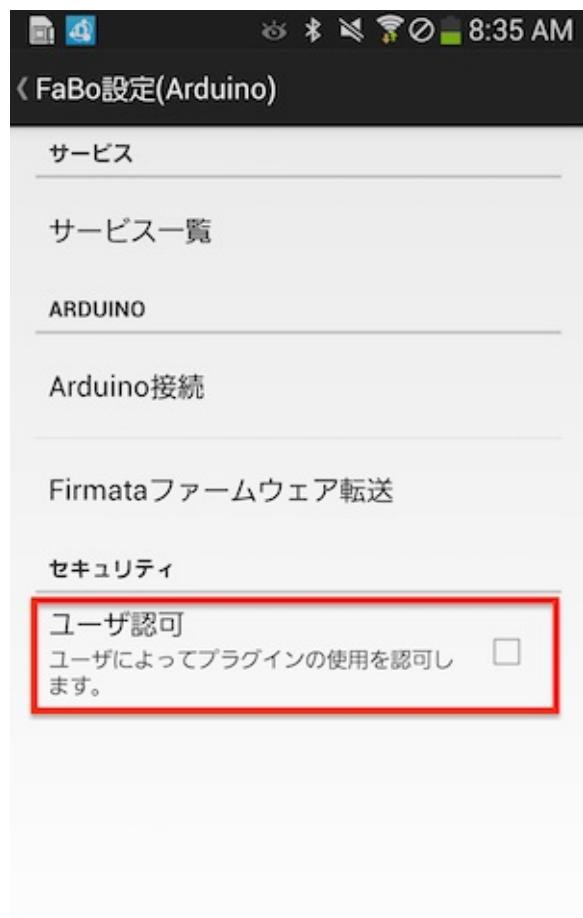


7.1 認証ダイアログの表示/非表示設定



ユーザ許可のチェックマークをいれると、ユーザ認証の画面が表示されます。

7.1 認証ダイアログの表示/非表示設定



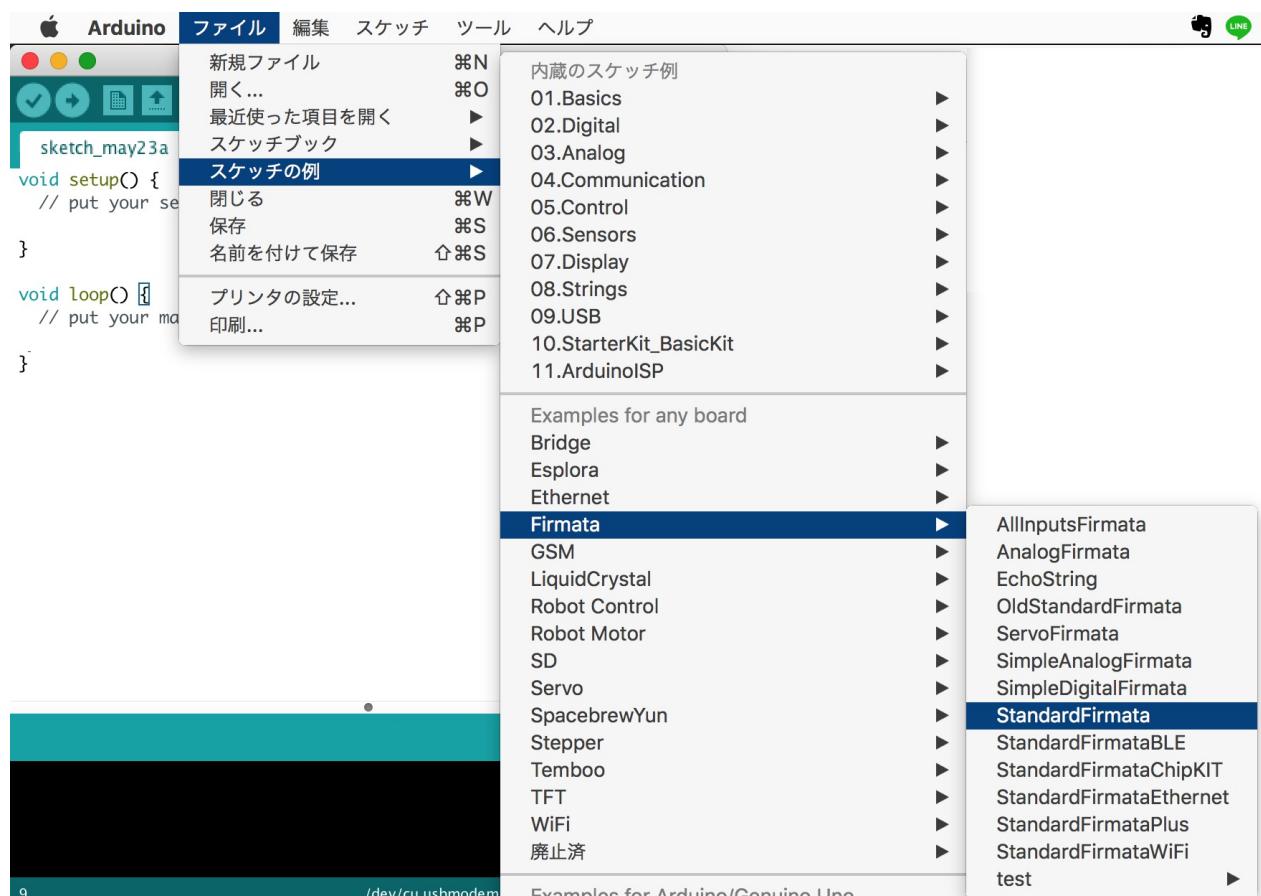
ユーザ認証の画面。

7.1 認証ダイアログの表示/非表示設定



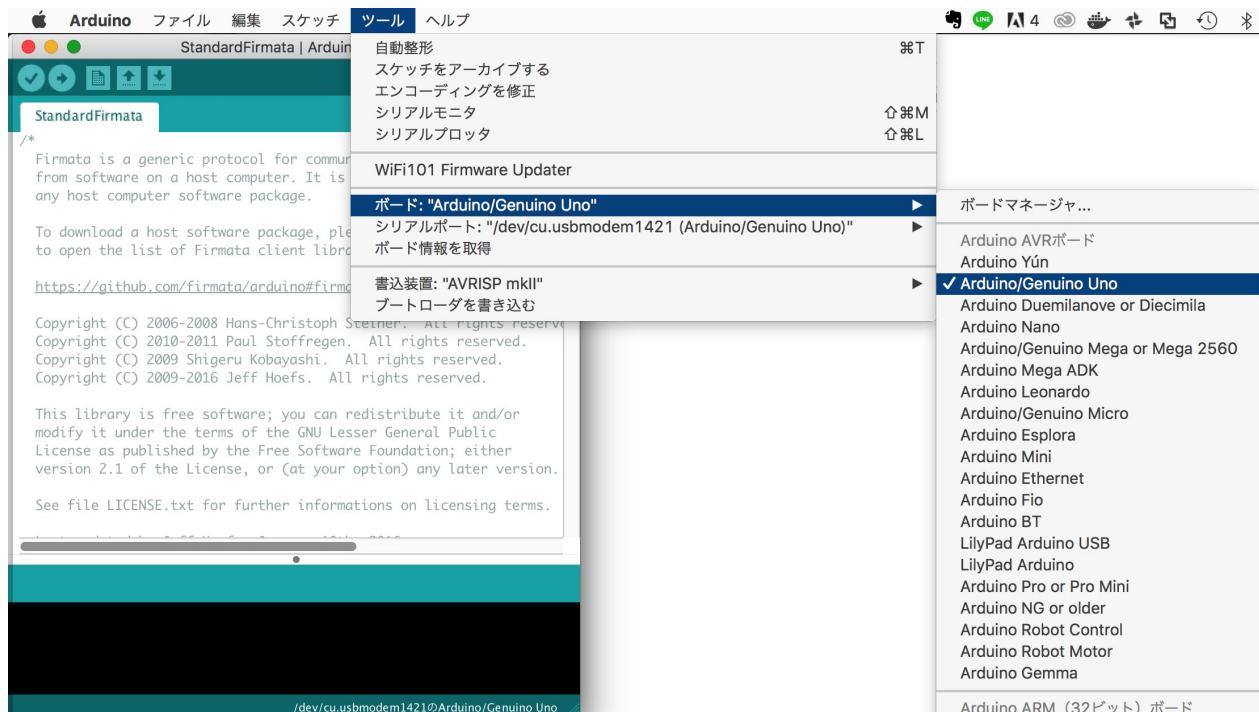
7.2 Firmataのインストール

Firmataのインストール

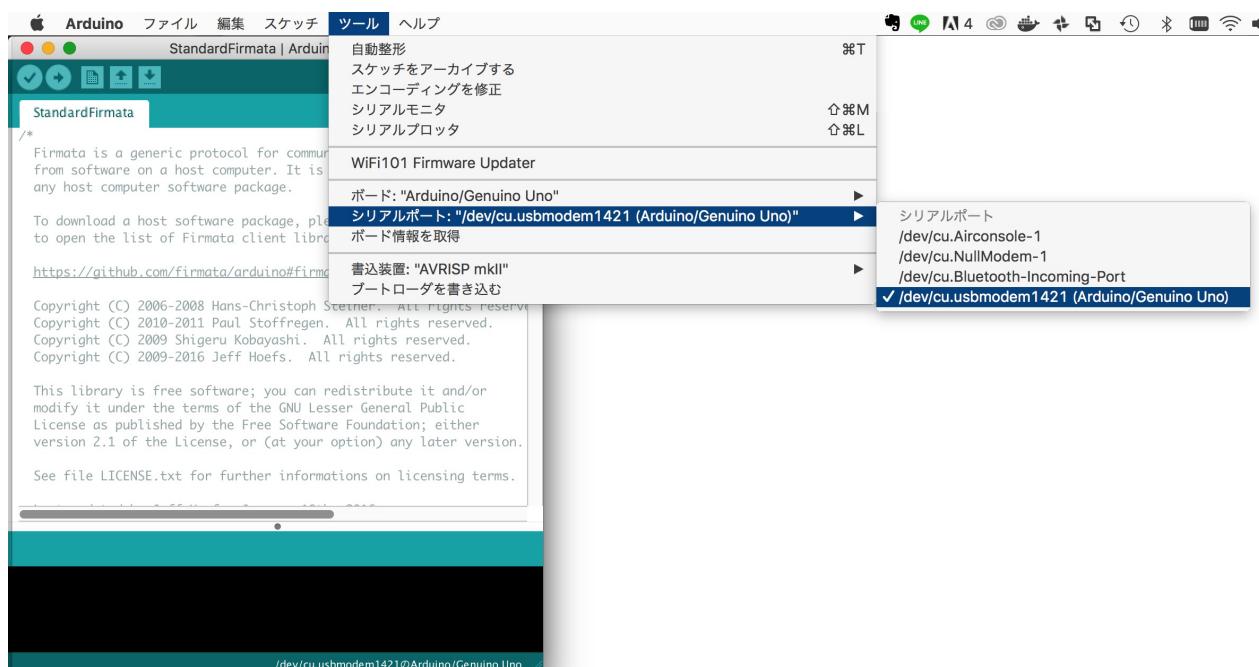


Arduinoの種類の設定

7.2 Firmataのインストール

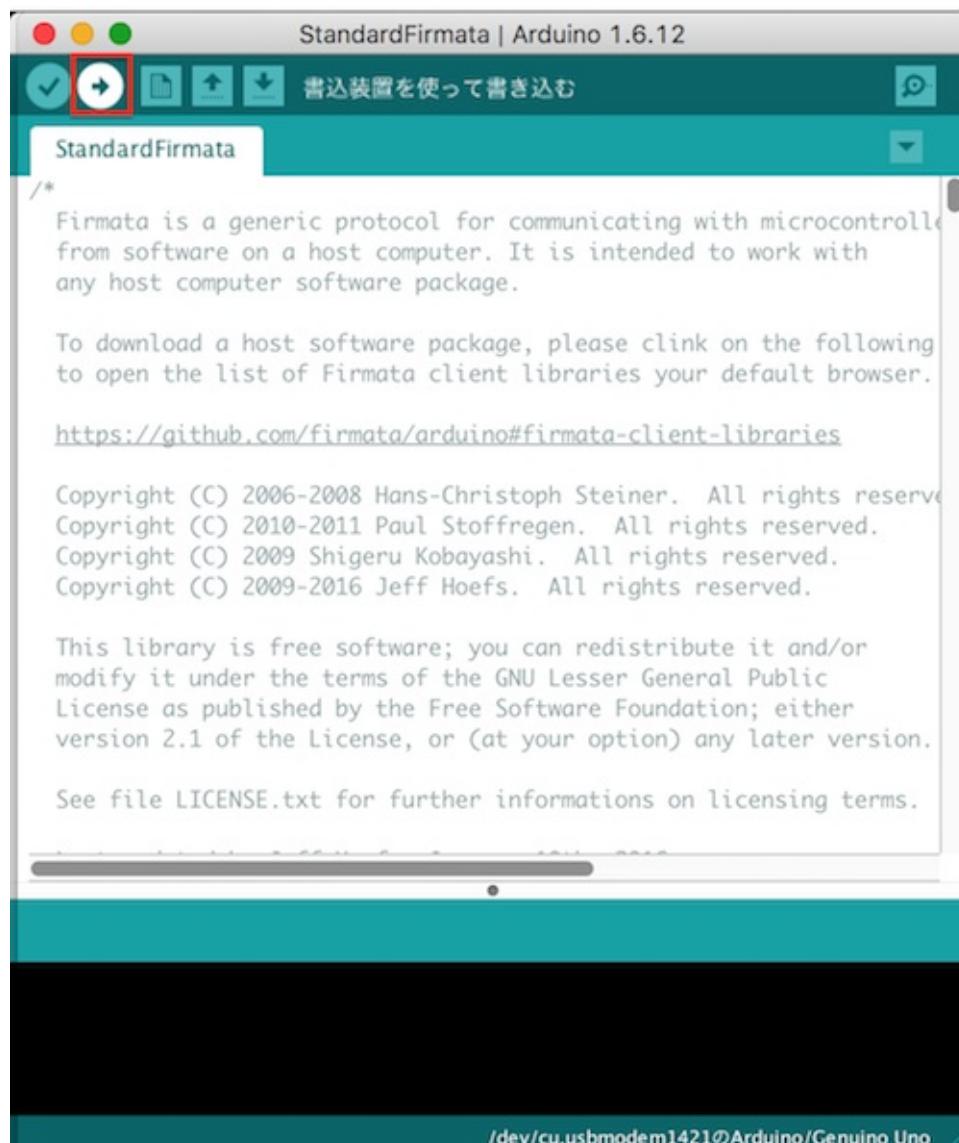


Serialポートの設定



Arduinoに転送

7.2 Firmataのインストール

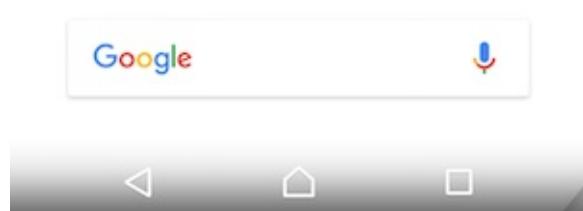


7.3 USB Deviceの認識

USB Host非対応の機種

端末名	対応状況
Fujitsu F-03A	USB Host非搭載
Xperia S0-02H	設定-機器接続-USB接続設定でUSB機器を認識

Xperia S0-02HでのUSBデバイスの認識方法



7.3 USB Deviceの認識





7.3 USB Deviceの認識



7.3 USB Deviceの認識



7.4 デモサイト

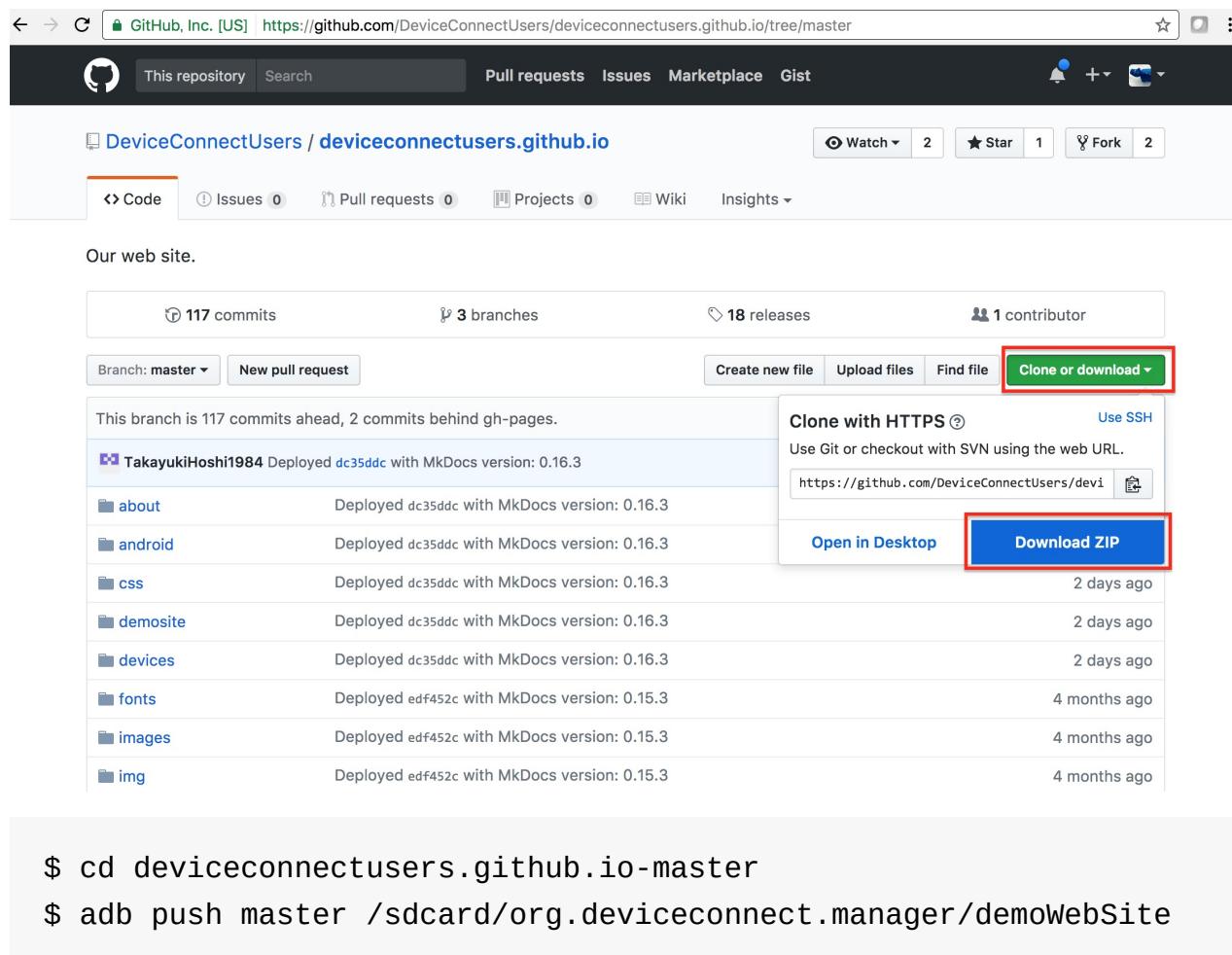
デモサイトをスマフォにいれる方法を解説します。

デモサイトをスマフォにいれる

<https://github.com/DeviceConnectUsers/deviceconnectusers.github.io/tree/master>

にアクセスします。

[Clone or Download]を選択し、Download ZIPを選び、RepositoryをDownloadします。



The screenshot shows a GitHub repository page for 'DeviceConnectUsers / deviceconnectusers.github.io'. The repository has 117 commits, 3 branches, 18 releases, and 1 contributor. The 'Clone or Download' button is highlighted with a red box. Below it, the 'Download ZIP' button is also highlighted with a red box. A terminal window at the bottom shows the command to clone the repository and push it to an Android device.

```
$ cd deviceconnectusers.github.io-master
$ adb push master /sdcard/org.deviceconnect.manager/demoWebSite
```

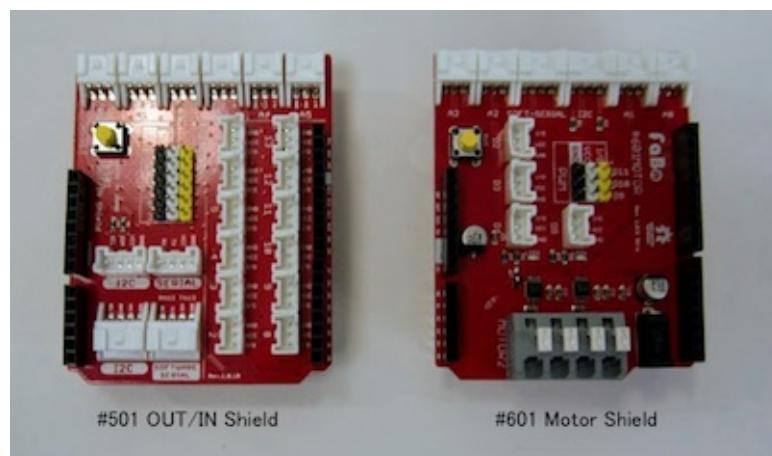
PCからのアクセス

7.4 デモサイトをスマフォにいれる

スマフォのIPアドレスを指定し、PCからアクセスします。

8.1 トラブルシューティング

I2Cの3軸加速度センサーから値が戻ってこない



#601 Motor Sheild を使用する場合、DCジャックでの電源供給がないと、I2Cへの電源供給が行われません。#601 Motor Sheild を使用する場合は、DCジャックにモバイルバッテリーを接続の上、モバイルバッテリーの電源をOnにして、FaBo Device Pluginからアクセスするようにしてください。



ポートがつながらない

Androidでは1-1023の間のポート番号が使えません。1024番以上のポートを使用しているかを確認してください。

8.1 トラブルシューティング
