



IoTデバイス活用マニュアル

センサ種類: 温湿度センサー

型番: Grove SHT31 / 35

ストーリー: 温湿度の値をGoogleSpreadシートへ記載する



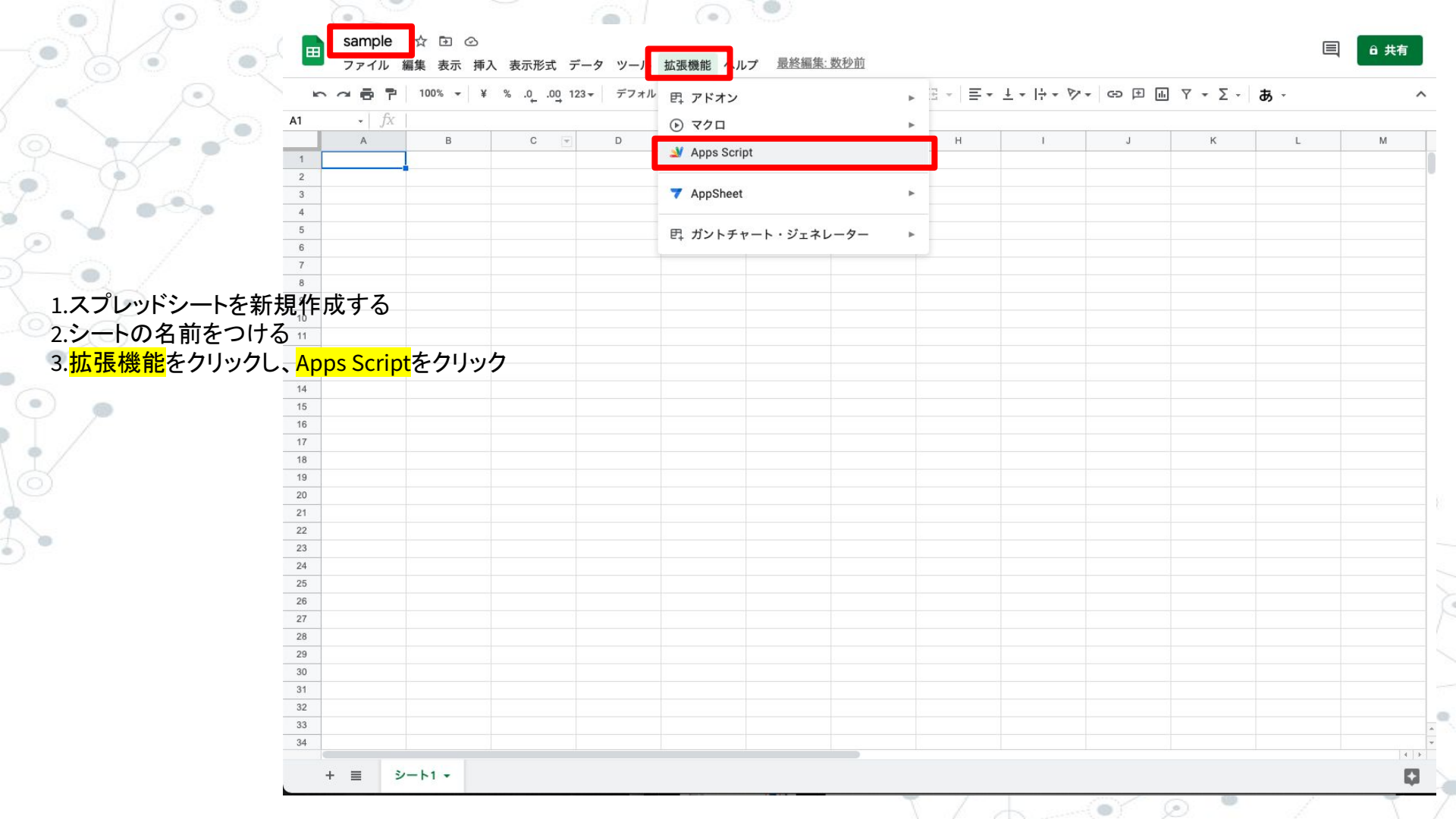
改訂記録:

2022/04/05 初版 作成: 豊住



1. GoogleSheetシートの準備

- 組織アカウントでの使用は情シスに要確認
- データ記載専用のアカウントを準備することをおすすめします。



1. スプレッドシートを新規作成する
2. シートの名前をつける
3. 拡張機能をクリックし、Apps Scriptをクリック

- 4.無題のプロジェクトをクリックし、名前をつける
* 資料では、Temp&Humiとする



ファイル



実行



デバッグ



関数なし



実行ログ

クラシック エディタを使用する



コード.gs



ライブラリ



サービス



プロジェクトを保存

```

1  var SS = SpreadsheetApp.getActiveSpreadsheet(); var SHEET = SS.getSheets()[0];
2  var str = "";
3
4  function doPost(e) { var parsedData; var result = {};
5  try {
6  parsedData = JSON.parse(e.postData.contents); }
7  catch(f){
8  return ContentService.createTextOutput("Error in parsing request body: " + f.message); }
9  if (parsedData !== undefined){ switch (parsedData.command) {
10 case "appendRow":
11 var dataArr = parsedData.values.split(",");
12 var now = Utilities.formatDate(new Date(), "JST", "yyyy/MM/dd HH:mm:ss"); dataArr.unshift(now);
13 SHEET.appendRow(dataArr);
14 str = "Success(appendRow)"; SpreadsheetApp.flush(); break;
15 }
16 return ContentService.createTextOutput(str); }
17 else{
18 return ContentService.createTextOutput("Error! Request body empty or in incorrect format."); }
19 }
20

```

5.次ページにあるプログラムをコピーペーストする

* デフォルトで記載されている `function My function(){}` の部分は削除して構わない



```
var SS = SpreadsheetApp.getActiveSpreadsheet(); var SHEET = SS.getSheets()[0];
var str = "";

function doPost(e) { var parsedData; var result = {};

try {
  parsedData = JSON.parse(e.postData.contents);
} catch(f){

  return ContentService.createTextOutput("Error in parsing request body: " + f.message);
}

if (parsedData !== undefined){ switch (parsedData.command) {

  case "appendRow":

    var dataArr = parsedData.values.split(",");

    var now = Utilities.formatDate(new Date(), "JST", "yyyy/MM/dd HH:mm:ss"); dataArr.unshift(now);

    SHEET.appendRow(dataArr);

    str = "Success(appendRow)"; SpreadsheetApp.flush(); break;

  }

  return ContentService.createTextOutput(str);
}

else{

  return ContentService.createTextOutput("Error! Request body empty or in incorrect format.");
}
```



ファイル



doPost



実行ログ

コード.gs

ライブラリ



サービス



```
1 var SS = SpreadsheetApp.getActiveSpreadsheet(); var SHEET = SS.getSheets()[0];
2 var str = "";
3
4 function doPost(e) { var parsedData; var result = {};
5 try {
6 parsedData = JSON.parse(e.postData.contents); }
7 catch(f){
8 return ContentService.createTextOutput("Error in parsing request body: "
9 if (parsedData !== undefined){ switch (parsedData.command) {
10 case "appendRow":
11 var dataArr = parsedData.values.split(",");
12 var now = Utilities.formatDate(new Date(), "JST", "yyyy/MM/dd HH:mm:ss");
13 SHEET.appendRow(dataArr);
14 str = "Success(appendRow)"; SpreadsheetApp.flush(); break;
15 }
16 return ContentService.createTextOutput(str); }
17 else{
18 return ContentService.createTextOutput("Error! Request body empty or in i
19 }
20
```

デプロイ



新しいデプロイ



ディタを使用する

デプロイを管理

デプロイをテスト

6.画面右上のデプロイをクリック

7.出てきたリストの中から新しいデプロイをクリック

Apps Script Temp&Humi

デプロイ

クラシック エディタを使用する

ファイル +

コード.gs

ライブラリ +

サービス +

新しいデプロイ

種類の選択 設定

デプロイタイプを有効にする

ウェブアプリ

実行可能 API

アドオン

ライブラリ

キャンセル デプロイ

```
1 var SS = SpreadsheetApp.getActiveSpreadsheet(); var SHEET = SS.getSheets()[0];
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

デプロイタイプ

8. 種類の選択右側の歯車アイコンをクリック
9. リストからウェブアプリをクリック



Apps Script Temp&Humi

デプロイ

クラシック エディタを使用する

新しいデプロイ

種類の選択

ウェブアプリ

設定

説明

新しい説明文

WioNode + SHT35のテスト用

ウェブアプリ

次のユーザーとして実行:

自分 ()gmail.com

このウェブ アプリケーションを実行するために、あなたのアカウント データを使用することを許可します。

アクセスできるユーザー

全員

自分のみ

Google アカウントを持つ全員

全員

キャンセル

デプロイ

```
1 var SS = SpreadsheetApp.getActiveSpreadsheet(); var SHEET = SS.getSheets()[0];
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

10.新しい説明文には任意の説明を入れる

11.アクセスできるユーザーをクリックし、全員に変更する

Apps Script Temp&Humi

デブロイ

クラシック エディタを使用する

ファイル +

コード.gs

ライブラリ +

サービス +

新しいデブロイ

このウェブ アプリケーションを使用するには、データへのアクセスを許可する必要があります。

アクセスを承認

Google にログイン

アカウントの選択

「Temp&Humi」に移動

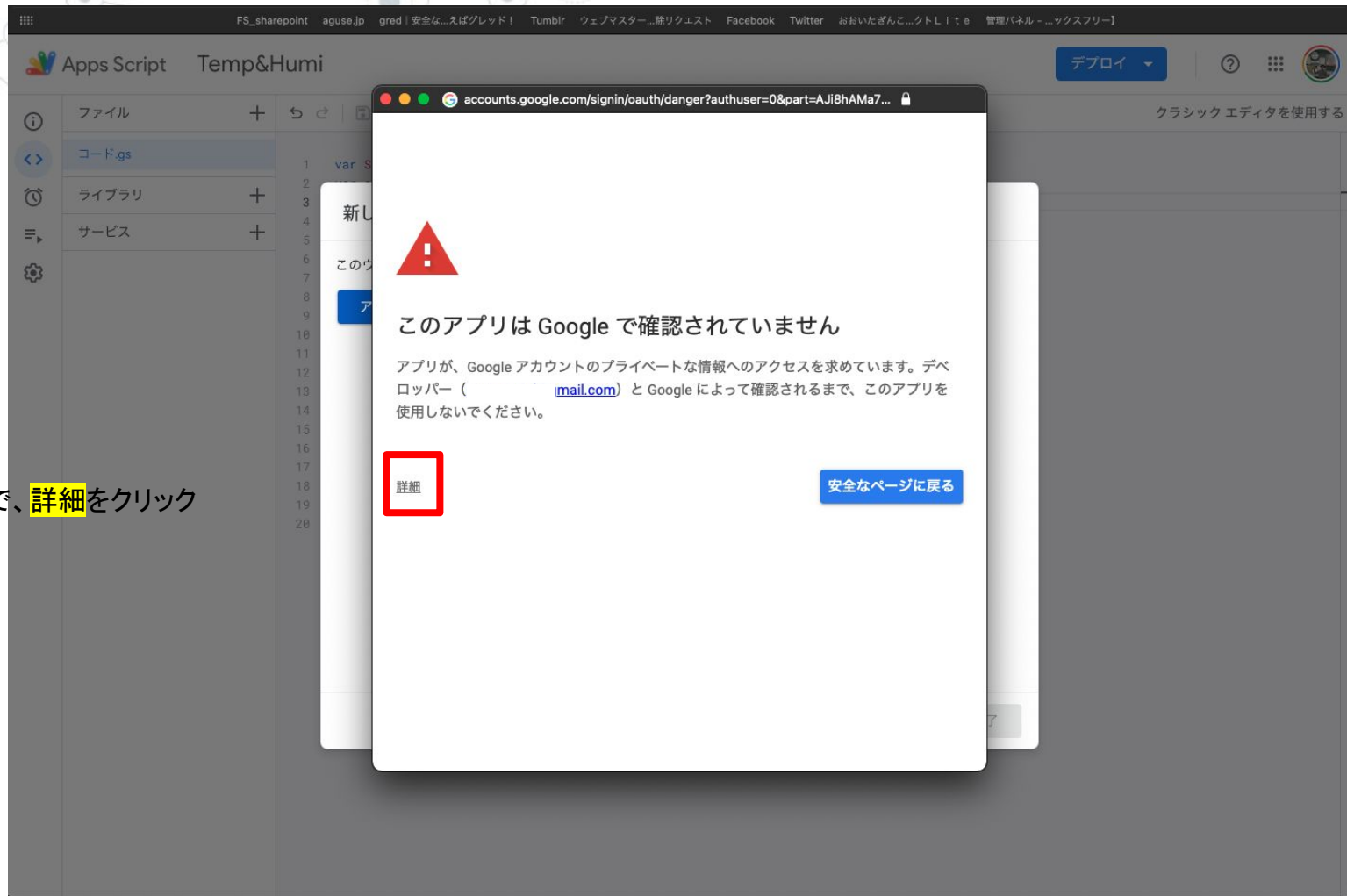
別のアカウントを使用

日本語 ヘルプ プライバシー 規約

```
1 var SS = SpreadsheetApp.getActiveSpreadsheet(); var SHEET = SS.getSheets()[0];
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

12. **アクセスを承認**をクリックする

13. アカウントの選択で使用するアカウントをクリック



14. 注意画面が出るので、**詳細**をクリック

Apps Script Temp&Humi

デブロイ

クラシック エディタを使用する

ファイル +

コード.gs

ライブラリ +

サービス +

新し

このウ


ア

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18

var S

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18

accounts.google.com/signin/oauth/danger?authuser=0&part=Aji8hAMa7...



このアプリは Google で確認されていません

アプリが、Google アカウントのプライベートな情報へのアクセスを求めています。デベロッパー (toyozumi.d@gmail.com) と Google によって確認されるまで、このアプリを使用しないでください。

詳細を非表示

安全なページに戻る

リスクを理解し、デベロッパー (toyozumi.d@gmail.com) を信頼できる場合のみ、続行してください。

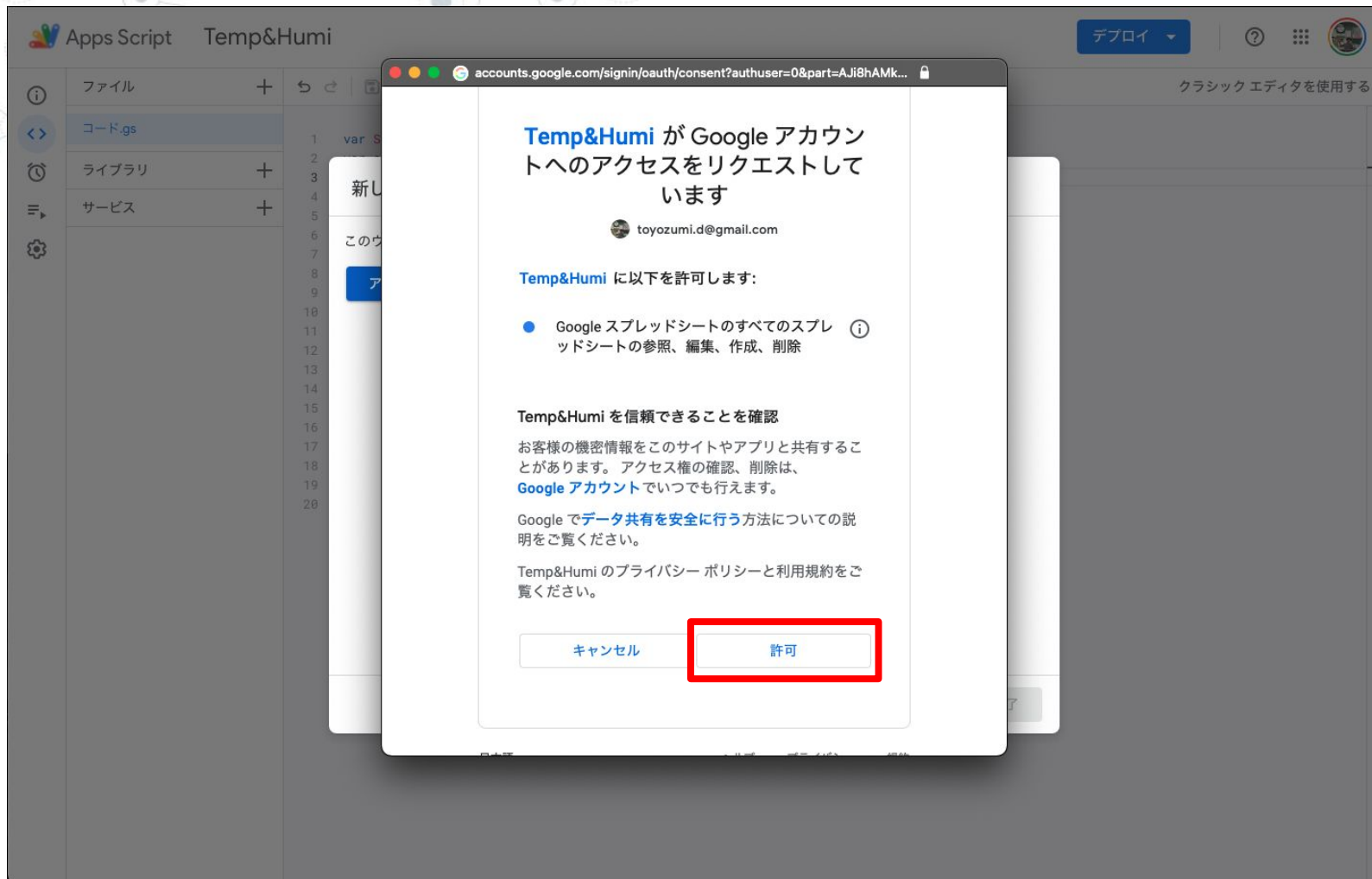
Temp&Humi (安全ではないページ) に移動

*https://accounts.google.com/#" を新規タブで開く

15. Temp&Humi (安全ではないページ) に移動をクリック

* 自分でつけた名前が表示される

16.許可をクリック



Apps Script Temp&Humi

デプロイ

クラシック エディタを使用する

ファイル +

コード.gs

ライブラリ +

サービス +

新しいデプロイ

デプロイを更新しました。

バージョン 1 (4月5日 11:36)

デプロイ ID

AKfycbx

AKfycbx

コピー

ウェブアプリ

URL

<https://script.google.com/macros/s/AKfycbx-vnk6Pp>

コピー

```
1 var SS = SpreadsheetApp.getActiveSpreadsheet(); var SHEET = SS.getSheets()[0];
2 var STR = "";
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

Apps Script Temp&Humi

ファイル +

コード.gs

ライブラリ +

サービス +

デプロイ

新しいデプロイ

デプロイを管理

デプロイをテスト

クラシック エディタを使用する

```
1 var SS = SpreadsheetApp.getActiveSpreadsheet(); var SHEET = SS.getSheets()[0];
2 var STR = "";
3
4 function doPost(e) { var parsedData; var result = {};
5   try {
6     parsedData = JSON.parse(e.postData.contents);
7     catch(e){
8       return ContentService.createTextOutput("Error in parsing request body: " + e.message); }
9   if (parsedData != undefined){ switch (parsedData.command) {
10     case "appendRow":
11       var dataArray = parsedData.values.split(",");
12       var now = Utilities.formatDate(new Date(), "GMT", "yyyy/MM/dd HH:mm:ss"); dataArray.unshift(now);
13       SHEET.appendRow(dataArray);
14       str = "Success(appendRow)"; SpreadsheetApp.flush(); break;
15     }
16   }
17   return ContentService.createTextOutput(str);
18 }
19
20 return ContentService.createTextOutput("Error! Request body empty or in incorrect format."); }
21 }
```

17. **デプロイID**をこの後使用するのでコピーを取っておく

* デプロイ>デプロイを管理 から何度も確認できるので、後からでもOK

sample ☆ □ ☁

ファイル 編集 表示 挿入 表示形式 データ ツール 拡張機能 ヘルプ 最終編集: 数秒前

100% ▼ ¥ % .0 .00 123 ▼ デフォルト... ▼ 10 ▼ B I S A ▼ 田 ▼ ≡ ▼ ⊥ ▼ | ▼ ▼ ▼ ☰ ▼ ☑ ▼ ☒ ▼ ▼ ▼ Σ ▼ あ ▼

C2 ▼ fx

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	時間	Temp	Humi										
2													
3													
4													
5													
6													
7													
8													
9													
10													
11													
12													
13													
14													
15													
16													
17													
18													
19													
20													
21													
22													
23													
24													
25													
26													
27													
28													
29													
30													
31													
32													
33													
34													

+ ≡ シート1 ▼

18.スプレッドシートに戻り、上記のように項目名を入力しておく



2. Arduino IDEの設定

- HTTPSredirect.hのインストール
- esp8266ボードマネージャの更新

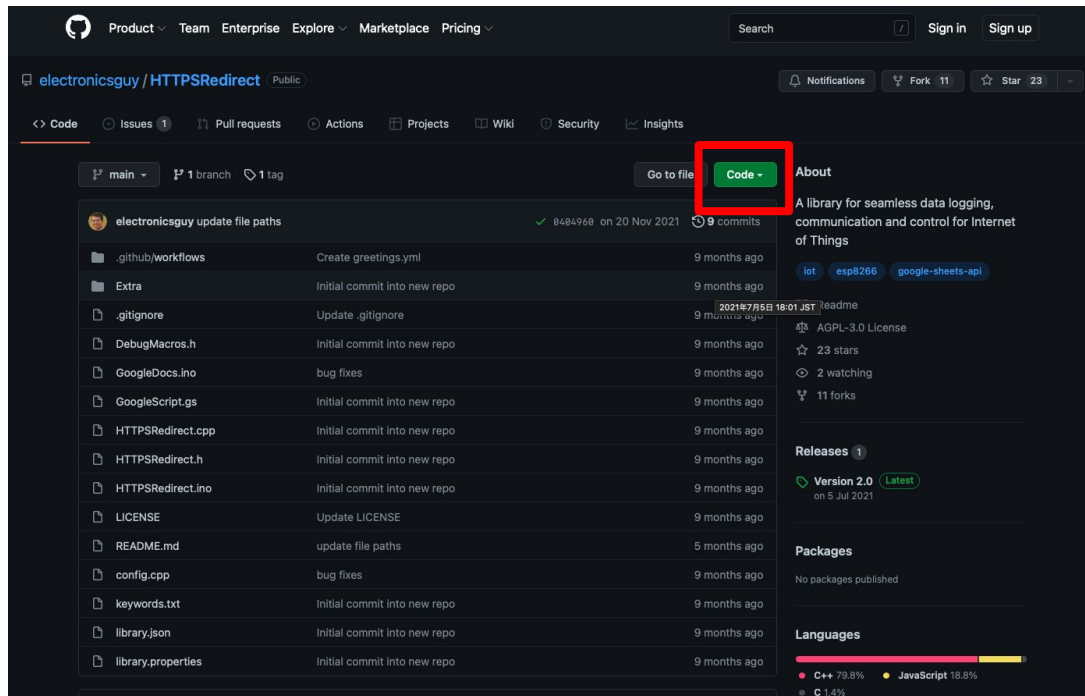


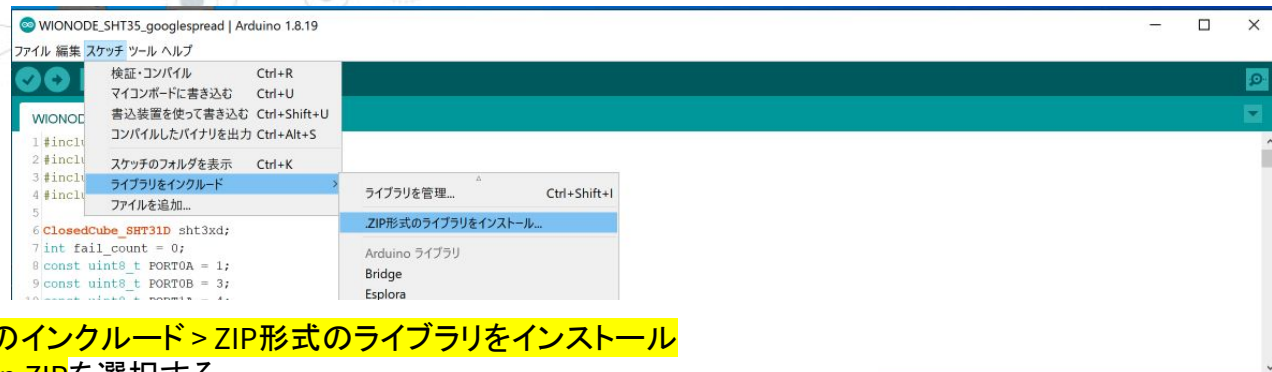
HTTPSredirect.hのインストール

1. 下記リンクからgithubを開く

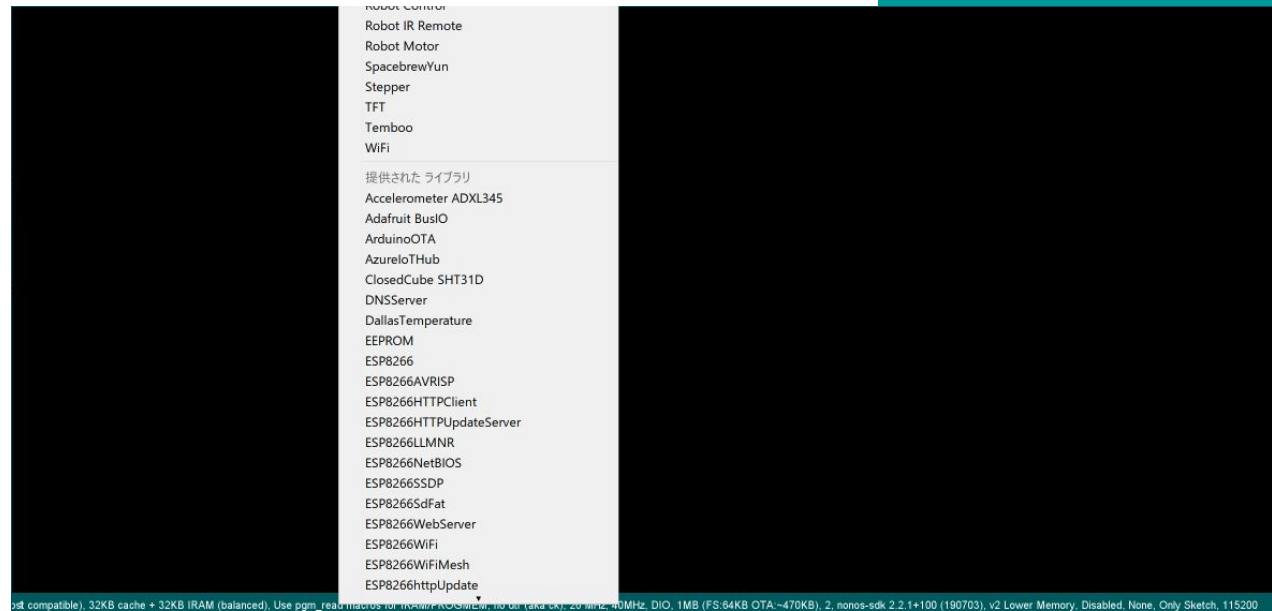
<https://github.com/electronicsguy/HTTPSRedirect>

2. **code**▼をクリックし、**Download ZIP**をクリック



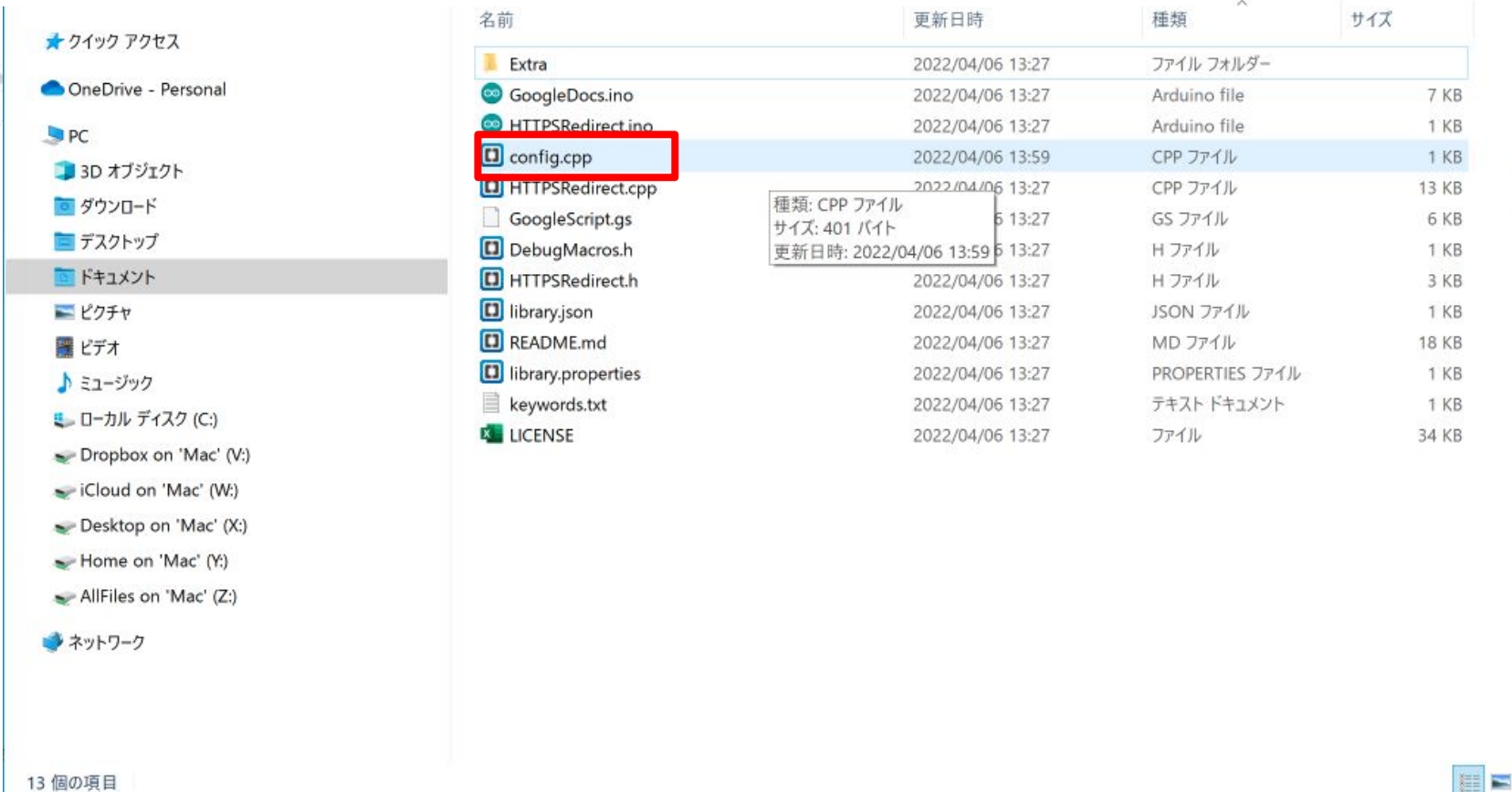


- 3.ArduinoIDEを開き、**スケッチ > ライブラリのインクルード > ZIP形式のライブラリをインストール**
- 4.先程ダウンロードした**HTTPSredirect-main.ZIP**を選択する



5.PC > Arduino > libraries > HTTPSRedirect-mainフォルダ内のconfig.cppを削除する

* フォルダ階層については、各自の設定箇所を参照してください



The screenshot displays a Windows File Explorer window titled 'HTTPSRedirect-main'. The left sidebar shows the navigation pane with 'ドキュメント' (Documents) selected. The main pane lists files and folders. The file 'config.cpp' is highlighted with a red rectangular box. A tooltip is visible over this file, providing the following information:

- 種類: CPP ファイル
- サイズ: 401 バイト
- 更新日時: 2022/04/06 13:59

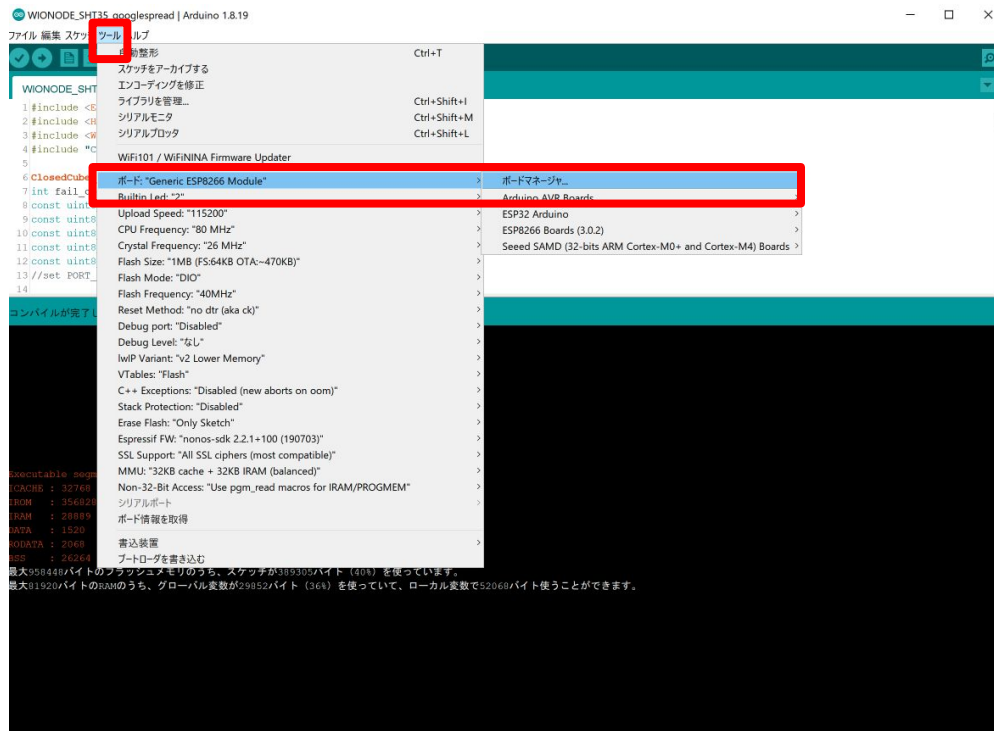
名前	更新日時	種類	サイズ
Extra	2022/04/06 13:27	ファイル フォルダー	
GoogleDocs.ino	2022/04/06 13:27	Arduino file	7 KB
HTTPSRedirect.ino	2022/04/06 13:27	Arduino file	1 KB
config.cpp	2022/04/06 13:59	CPP ファイル	1 KB
HTTPSRedirect.cpp	2022/04/06 13:27	CPP ファイル	13 KB
GoogleScript.gs	2022/04/06 13:27	GS ファイル	6 KB
DebugMacros.h	2022/04/06 13:27	H ファイル	1 KB
HTTPSRedirect.h	2022/04/06 13:27	H ファイル	3 KB
library.json	2022/04/06 13:27	JSON ファイル	1 KB
README.md	2022/04/06 13:27	MD ファイル	18 KB
library.properties	2022/04/06 13:27	PROPERTIES ファイル	1 KB
keywords.txt	2022/04/06 13:27	テキスト ドキュメント	1 KB
LICENSE	2022/04/06 13:27	ファイル	34 KB

13 個の項目

ESP8266ボードマネージャの更新

- ・講座で使用していた2.3.0というバージョンではスプレッドシートへ接続できません
- * Azure IoT Hubへのライブラリは2.3.0でしか動きませんので、ご注意ください。

1.Arduino IDEのツール>ボード>ボードマネージャーをクリック



```
lude <HTTPSRedirect.h>
lude <Wire.h>
lude "ClosedCube_SHT31D.h"
```

edCube ボードマネージャ

タイプ 全て

esp8266

esp8266

by ESP8266 Community バージョン3.0.2 INSTALLED

このパッケージに含まれているボード:

Generic ESP8266 Module, Generic ESP8285 Module, Lifyly Agrumino Lemon v4, ESPduino (ESP-13 Module), Adafruit Feather HUZZAH ESP8266, Invent One, XinaBox CW01, ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Olimex MOD-WIFI-ESP8266(-DEV), SparkFun ESP8266 Thing, SparkFun ESP8266 Thing Dev, SparkFun Blynk Board, SweetPea ESP-210, LOLIN(WEMOS) D1 R2 & mini, LOLIN(WEMOS) D1 mini (clone), LOLIN(WEMOS) D1 mini Pro, LOLIN(WEMOS) D1 mini Lite, LOLIN(WeMos) D1 R1, ESPino (ESP-12 Module), ThaiEasyElec's ESPino, WifiInfo, Arduino, 4D Systems gen4 IoT Range, Digistump Oak, WIFIduino, Amperka WiFi Slot, Seeed Wio Link, ESPectro Core, Schirmilabs Eduino WiFi, ITEAD Sonoff, DOIT ESP-Mx DevKit (ESP8285).

[Online Help](#)

[More Info](#)

バージョンを選択

インストール

削除

3.0.1
3.0.0
2.7.4
2.7.3
2.7.2
2.7.1
2.7.0

2. 検索窓に esp8266 と入力
3. バージョンを選択 をクリックし、3.0.2 をインストール
4. 終了したら閉じる

```
2060 // ...
26264 } - zeroed variables (global, static) in RAM/HEAP
8バイトのフラッシュメモリのうち、スケッチが389305バイト (40%) を使っています。
バイトのRAMのうち、グローバル変数が29852バイト (36%) を使っていて、ローカル変数で52068バイト使うことができます。
```



3. Arduino スケッチの設定

- WiFiのSSID / PASSの書き込み
- GoogleのIDの書き込み

1.下記に保存しているスケッチを開く

2.スケッチの24行目から29行目の部分を書き換える
SSID / PASS / Google ID

注意1. ” ” の中に書き込むこと

注意2.GoogleIDは、[スプレッドシートのデプロイ](#)の際に出てきたモノ

3.**SHT31の場合はコメントアウト部分の設定**を入れ替えること！

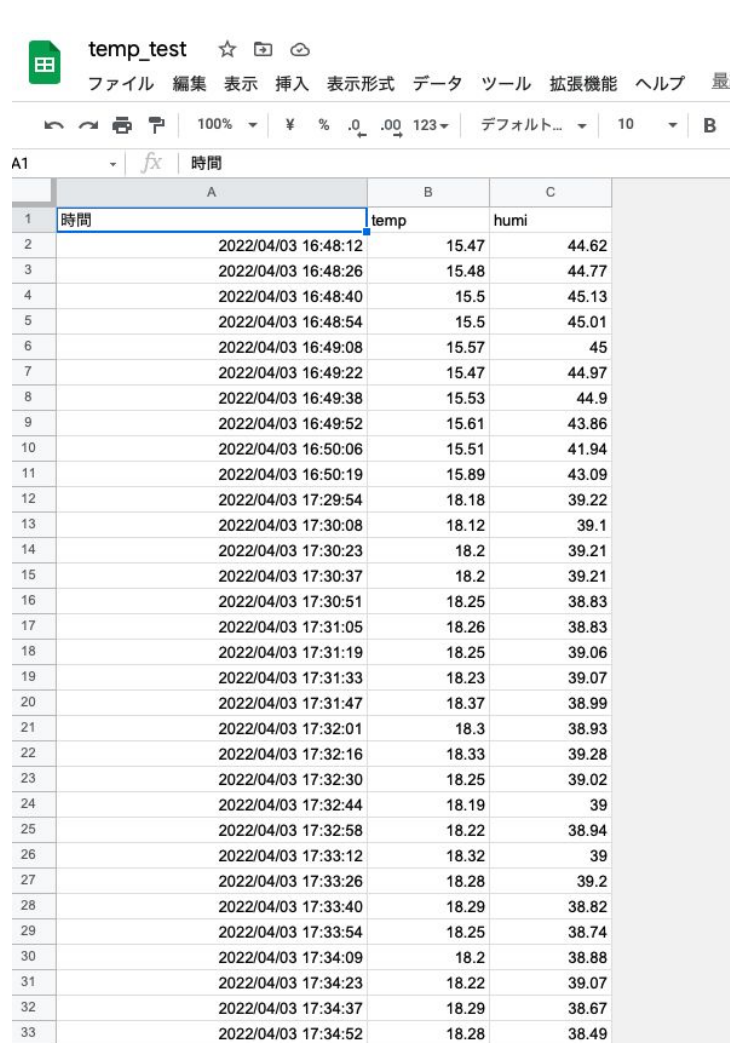
```
// wifi設定
const char* ssid = "SSID";
const char* password = "PASS";

//Googleスプレッドシートの設定
const char *GScriptId = "Google ID";
const char* host = "script.google.com";
const int httpsPort = 443;
String url = String("/macros/s/") + GScriptId + "/exec";
const String payload_base = "{\"command\": \"appendRow\", \"sheet_name\": \"Sheet1\", \"values\": \"\"";
String payload = "";
HTTPSRedirect* client = nullptr;
```


4.Wio Nodeへ書き込みを行う

* FUNC ボタン押しながら、RSTで書き込みモードにするのを忘れない

5.スプレッドシートへ書き込まれているか確認



temp_test ☆ 📄 ☁

ファイル 編集 表示 挿入 表示形式 データ ツール 拡張機能 ヘルプ 最

100% ▼ ¥ % .0_ .00 123 ▼ デフォルト... ▼ 10 ▼ B

A1 fx 時間

	A	B	C
1	時間	temp	humi
2	2022/04/03 16:48:12	15.47	44.62
3	2022/04/03 16:48:26	15.48	44.77
4	2022/04/03 16:48:40	15.5	45.13
5	2022/04/03 16:48:54	15.5	45.01
6	2022/04/03 16:49:08	15.57	45
7	2022/04/03 16:49:22	15.47	44.97
8	2022/04/03 16:49:38	15.53	44.9
9	2022/04/03 16:49:52	15.61	43.86
10	2022/04/03 16:50:06	15.51	41.94
11	2022/04/03 16:50:19	15.89	43.09
12	2022/04/03 17:29:54	18.18	39.22
13	2022/04/03 17:30:08	18.12	39.1
14	2022/04/03 17:30:23	18.2	39.21
15	2022/04/03 17:30:37	18.2	39.21
16	2022/04/03 17:30:51	18.25	38.83
17	2022/04/03 17:31:05	18.26	38.83
18	2022/04/03 17:31:19	18.25	39.06
19	2022/04/03 17:31:33	18.23	39.07
20	2022/04/03 17:31:47	18.37	38.99
21	2022/04/03 17:32:01	18.3	38.93
22	2022/04/03 17:32:16	18.33	39.28
23	2022/04/03 17:32:30	18.25	39.02
24	2022/04/03 17:32:44	18.19	39
25	2022/04/03 17:32:58	18.22	38.94
26	2022/04/03 17:33:12	18.32	39
27	2022/04/03 17:33:26	18.28	39.2
28	2022/04/03 17:33:40	18.29	38.82
29	2022/04/03 17:33:54	18.25	38.74
30	2022/04/03 17:34:09	18.2	38.88
31	2022/04/03 17:34:23	18.22	39.07
32	2022/04/03 17:34:37	18.29	38.67
33	2022/04/03 17:34:52	18.28	38.49



4. その他備考



最大保存数について

Google スプレッドシートは最大1000万セルまでとなっています。(2022/03現在)

データ数がサンプルのように3列(A - C列)の場合は、D列以降を削除することで最大3,333,333行書き込めます。

ただし、行が増えれば増えるほどブラウザが重くなる＆表示に時間がかかるので、

適宜別にコピーするなり、スプレッドシートを再発行するなりしてあげるほうが良いかもしれません。

タイムラグについて

スプレッドシートへの接続と保存のラグがあるようで、

10秒サイクルで送信していても受信時間は10秒サイクルとはなりませんのでご注意ください。

config.cppの削除について

今回はArduinoのスケッチの中でSSID / PASS / GoogleIDを記載しているため、削除しています。

* 残していると重複しているというエラーが返ります。

スケッチのSSID / PASS / GoogleID箇所を削除し、config.cppへ書き込むことでも動作するはずですので、興味があれば試してみてください。

DeepSleepあり・なし

Arduinoのスケッチを2種類準備しています。

無印: 4回以上Googleへ接続エラーをした場合は、WioNodeをスリープ状態にする

nondeepsleep: エラーが何度起きてもWioNode自体は稼働し続ける

Googleアカウントが漏れてしまえば計測値も全て漏れる可能性

インターネット上のサービスに関してはすべて言えることですが、管理を間違えると外に全て見えてしまうということが起きてしまいます。スプレッドシートの公開範囲や編集権限等くれぐれも気をつけてください。

各自、自己責任・自己管理でのご利用をお願いします。