

USB 温度計 TEMP_{er} 使用法

Ubuntu への TEMP_{er}ed インストール及び、GAS 使用法

目次

hidapi/TEMPered のインストール	3
GAS の設定・使用	7
crontab と ShellScript	16
USB 温度計複数時の変更.....	18

hidapi/TEMPPered のインストール

Temper は公式が Linux で使用できると銘打っているが、公式の Linux ソフトウェアは存在しない。よって、有志による Temper プログラムを使用する。

Temper を Linux で使用するために hidapi のインストールを行う。方法はこの[ページ](#)に従う。

hidapi のインストールに必要なパッケージを事前に以下のコマンドでインストールしておく。

```
$ sudo apt install libusb-dev libudev-dev libusb-1.0.0-dev libfox-1.6-dev autotools-dev autoconf automake libtool
```

git から hidapi のライブラリを持ってくるので入ってなければ git もインストールしておく。

```
$ sudo apt install git
```

hidapi ライブラリを git からコピーしてくる。

```
$ git clone https://github.com/signal11/hidapi
```

```
$ git clone https://github.com/signal11/hidapi
Cloning into 'hidapi'...
remote: Enumerating objects: 2006, done.
remote: Total 2006 (delta 0), reused 0 (delta 0), pack-reused 2006
Receiving objects: 100% (2006/2006), 2.73 MiB | 1.14 MiB/s, done.
Resolving deltas: 100% (1172/1172), done.
$
```

makefile を指定して make を行う。

ディレクトリ hidapi/linux/ に移動し、

```
$ make -f Makefile-manual
```

```
$ cd hidapi/linux/
$ make -f Makefile-manual
cc -Wall -g -fpic -c -I../hidapi `pkg-config libusb-1.0 --cflags` hid.c -o hid.o
g++ -Wall -g -fpic -c -I../hidapi `pkg-config libusb-1.0 --cflags` ../hidtest/hidtest.cpp -o ../hidtest/hidtest.o
g++ -Wall -g hid.o ../hidtest/hidtest.o `pkg-config libudev --libs` -lrt -o hidtest-hidraw
cc -Wall -g `pkg-config libudev --libs` -lrt -shared -fpic -Wl,-soname,libhidapi-hidraw.so.0 hid.o -o libhidapi-hidraw.so
$
```

cc コマンドで libhidapi-hidraw.so を作成する。

```
$ cc -Wall -g -lrt -shared -fpic -Wl,-soname,libhidapi-hidraw.so.0 hid.o -o libhidapi-hidraw.so `pkg-config libudev --libs`
```

コピーを /usr/local/lib/ 下に作成し、シンボリックリンクを作成する。

```
$ sudo cp libhidapi-hidraw.so /usr/local/lib
$ sudo ln -s libhidapi-hidraw.so libhidapi-hidraw.so.0
```

```
$ cc -Wall -g -lrt -shared -fpic -Wl,-soname,libhidapi-hidraw.so.0 hid.o
-o libhidapi-hidraw.so `pkg-config libudev --libs`
$ sudo cp libhidapi-hidraw.so /usr/local/lib/
$ sudo ln -s libhidapi-hidraw.so libhidapi-hidraw.so.0
```

ここまで来ると hidapi のインストールが可能になる。

```
$ sudo apt install libhidapi-hidraw0
```

yes/no を聞かれるので y を入力しインストール。

```
$ sudo apt install libhidapi-hidraw0
パッケージリストを読み込んでいます... 完了
依存関係ツリーを作成しています
状態情報を読み取っています... 完了
以下のパッケージが自動でインストールされましたが、もう必要とされていません:
  libllvm7
これを削除するには 'sudo apt autoremove' を利用してください。
以下のパッケージが新たにインストールされます:
  libhidapi-hidraw0
アップグレード: 0 個、新規インストール: 1 個、削除: 0 個、保留: 246 個。
10.4 kB のアーカイブを取得する必要があります。
この操作後に追加で 36.9 kB のディスク容量が消費されます。
取得:1 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 libhidapi-hidraw0 amd64 0.8.0~rc1+git20140818.d17db57+dfsg-2 [10.4 kB]
10.4 kB を 1秒 で取得しました (16.3 kB/s)
以前に未選択のパッケージ libhidapi-hidraw0:amd64 を選択しています。
(データベースを読み込んでいます ... 現在 177434 個のファイルとディレクトリがインストールされています。)
.../libhidapi-hidraw0_0.8.0~rc1+git20140818.d17db57+dfsg-2_amd64.deb を展開する準備をしています ...
libhidapi-hidraw0:amd64 (0.8.0~rc1+git20140818.d17db57+dfsg-2) を展開しています...
libhidapi-hidraw0:amd64 (0.8.0~rc1+git20140818.d17db57+dfsg-2) を設定しています ...
libc-bin (2.27-3ubuntu1) のトリガを処理しています ...
$ █
```

cmake でエラーが起きたので libhidapi-dev もインストール ※2020/06/29 追記

```
$ sudo apt install libhidapi-dev
```

できれば sudo apt update を行ってから行う。

今度は TEMPered をインストールする。/hidapi/linux/内にいるので、親の親ディレクトリに移動する。今回は TEMPered インストール用に temper というディレクトリを用意しておいたので、そこにインストールする。

hidapi/TEMPered のインストール

```
$ cd ../../  
$ ls  
hidapi  temper  
$ cd temper/  
$
```

TEMPered のライブラリを git からコピーする。

```
$ git clone https://github.com/hughesr/TEMPered
```

```
$ git clone https://github.com/hughesr/TEMPered  
Cloning into 'TEMPered'...  
remote: Enumerating objects: 527, done.  
remote: Total 527 (delta 0), reused 0 (delta 0), pack-reused 527  
Receiving objects: 100% (527/527), 133.29 KiB | 362.00 KiB/s, done.  
Resolving deltas: 100% (293/293), done.  
$ ls  
TEMPered  
$
```

git の branch を master から hack-413d-2107 に変更する。

```
$ git checkout hack-413d-2107
```

```
$ git checkout hack-413d-2107  
Branch 'hack-413d-2107' set up to track remote branch 'hack-413d-2107' from 'ori  
gin'.  
Switched to a new branch 'hack-413d-2107'  
$
```

checkout した後で reset をかける。

```
$ git reset --hard 75aa1e2
```

```
$ git reset --hard 75aa1e2  
HEAD is now at 75aa1e2 Apply hacks from issue edorfaus/TEMPered#51  
$
```

TEMPered をコンパイルする。

```
$ cmake .
```

```
$ make
```

```
$ cmake .
CMake Warning (dev) at CMakeLists.txt:57:
  Syntax Warning in cmake code at column 54

  Argument not separated from preceding token by whitespace.
This warning is for project developers.  Use -Wno-dev to suppress it.

-- The C compiler identification is GNU 7.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/atuzirou/workspace/package/temper/TEMPered
$ make
Scanning dependencies of target tempered-shared
```

error が出なければこれで temper を扱う準備は完了。

テストとして

```
$ sudo utils/tempered
```

もしくは utils ディレクトリに移動し、

```
$ sudo ./tempered
```

と入力し温度が返ってこれば正常。

```
tsuru07@tsuru07:~/package/temper/TEMPered/utils$ sudo ./tempered
/dev/hidraw1 0: temperature 28.00 °C
/dev/hidraw1 1: Failed to get the temperature: Not enough data was read from the sensor.
/dev/hidraw1 1: no sensor data available
tsuru07@tsuru07:~/package/temper/TEMPered/utils$
```

プログラムは両側読み出しの温度計に対応しているのに対し、使用する温度計が片側読み出しなので、もう片側でエラーが出るが問題ない

GAS の設定・使用

GoogleSpreadSheet に書き込むためのプログラムを GAS で用意する。

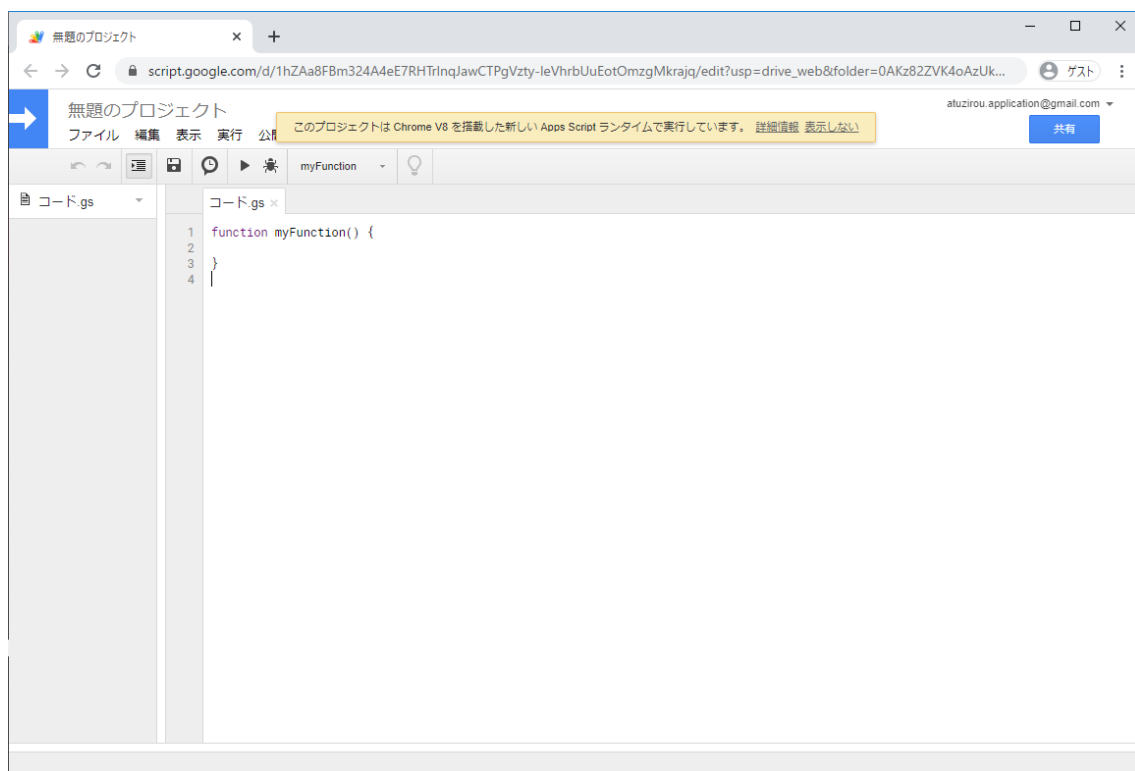
GAS(GoogleAppsScript)とは Google のサーバーで動かすことができるサーバーサイド言語で、JavaScript 言語に Google のアプリ (gmail や drive など)を使用するための関数が追加された言語である。目に見える制限としては 5 分以内に処理が終わるプログラムという点がある。

chrome を開いて GAS を追加する。



無題のプロジェクトが立ち上がる。名前を妥当なものにしておく。

GAS の設定・使用



今回は Linux から GAS に数値を送り、GAS 側で SpreadSheet に書き込む処理を行わせる。
コードは以下のようにした。

```
var sheet_url = PropertiesService.getScriptProperties().getProperty('SHEET_URL');

function doPost(e){
  SheetAppend(e);
  return 0;
}

function SheetAppend(e){
  var ss = SpreadsheetApp.openByUrl(sheet_url);
  var sheet = ss.getSheets()[0];
  var PostData = JSON.parse(e.postData.contents);
  var spl = PostData.value.split(" ") //multi USB value

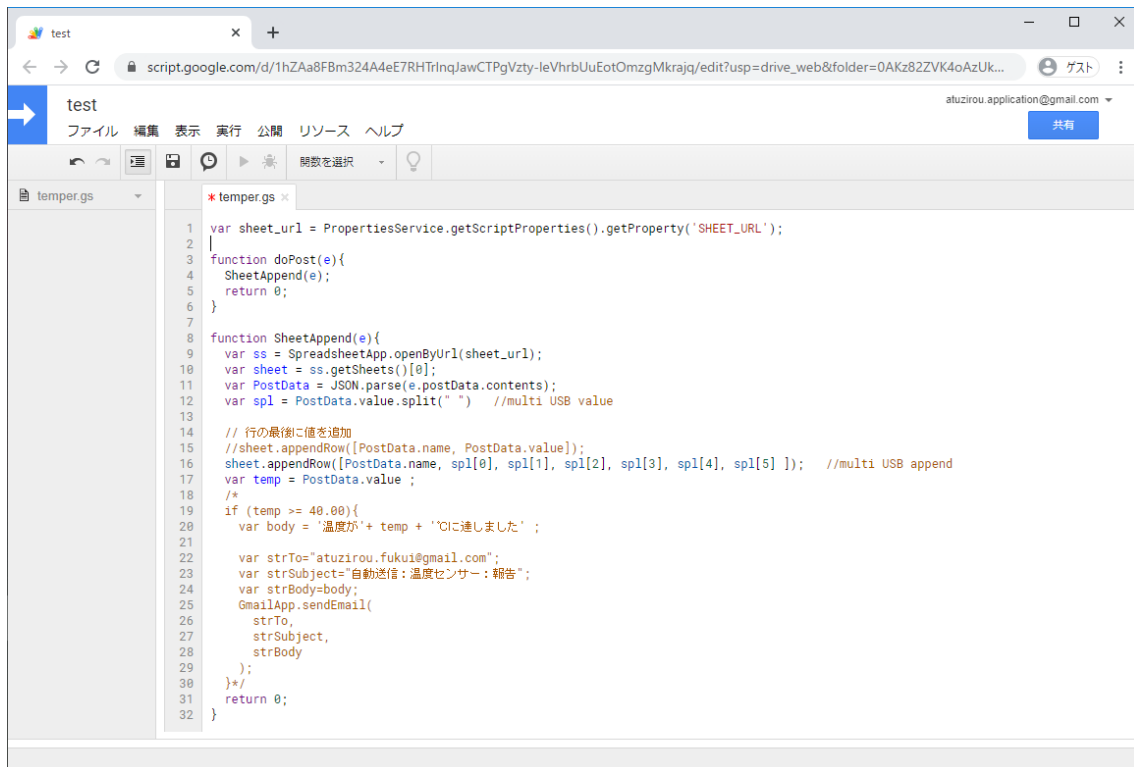
  // 行の最後に値を追加
  sheet.appendRow([PostData.name, PostData.value]);
  //sheet.appendRow([PostData.name, spl[0], spl[1], spl[2], spl[3], spl[4], spl[5] ]);
  //multi USB append
  var temp = PostData.value ;
  /*
  if (temp >= 40.00){
    var body = '温度が'+ temp + "°Cに達しました';

    var strTo="XXX@gmail.com";    ←アドレス
    var strSubject="自動送信：温度センサー：報告";
    var strBody=body;
    GmailApp.sendEmail(
      strTo,
      strSubject,
      strBody
    );
  }*/
  return 0;
}
```

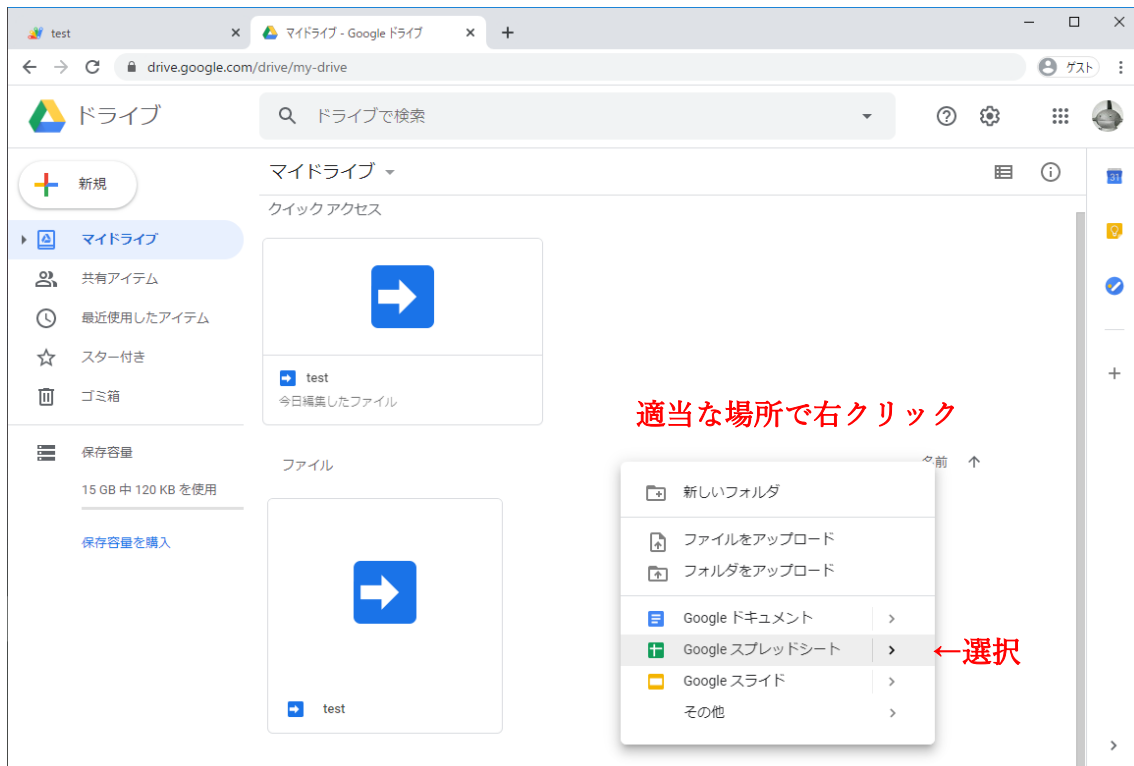
温度計が
←一つするとき
←複数のとき

温度が一定以上の
ときに連絡する

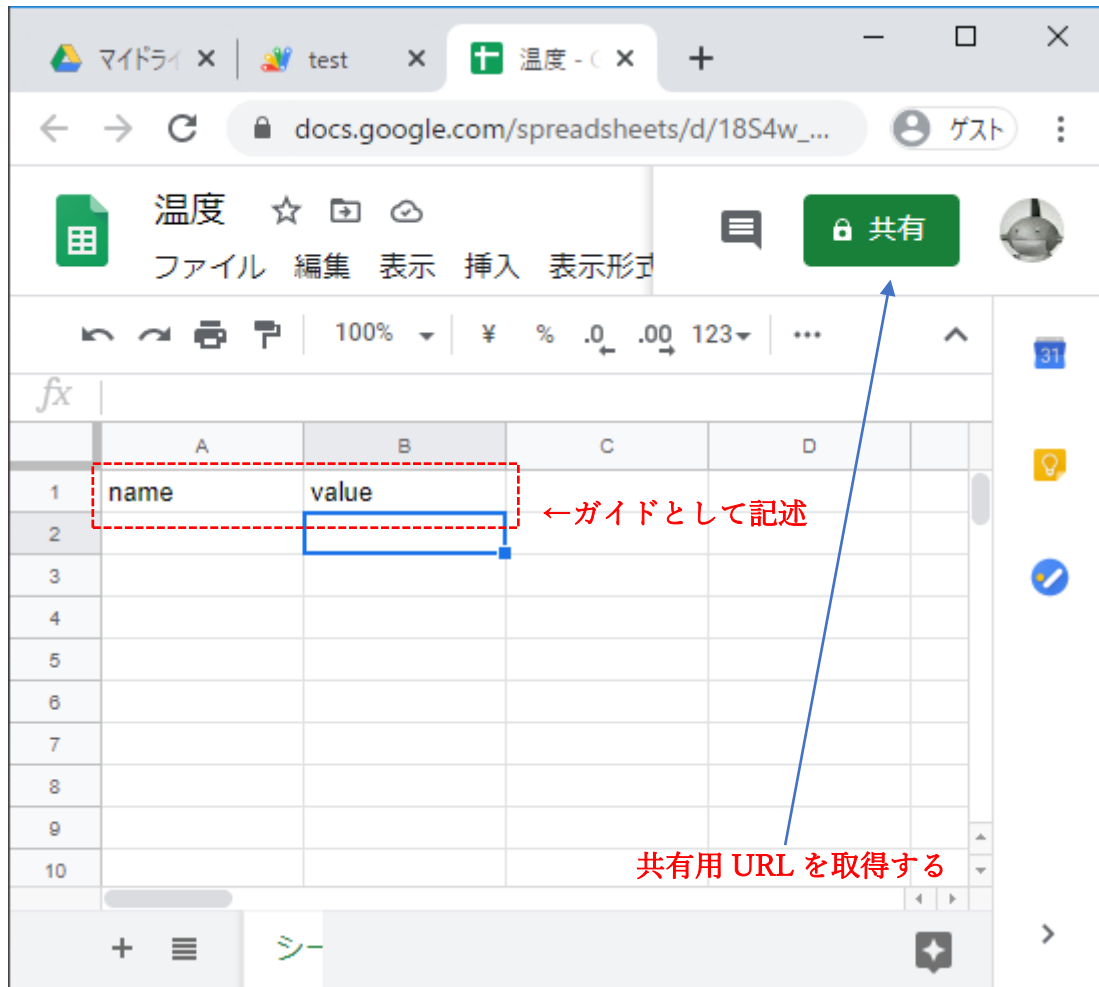
GAS の設定・使用



SpreadSheet を用意する。



温度だけなので以下のように記述しておく。



GAS がアクセスするための URL を取得する。共有を選択。





リンクを取得



↓ URL を取得

https://docs.google.com/spreadsheets/d/18S4w_MST6dxrsRVfPNbD...

リンクをコピー



リンクを知っている全員 ▼

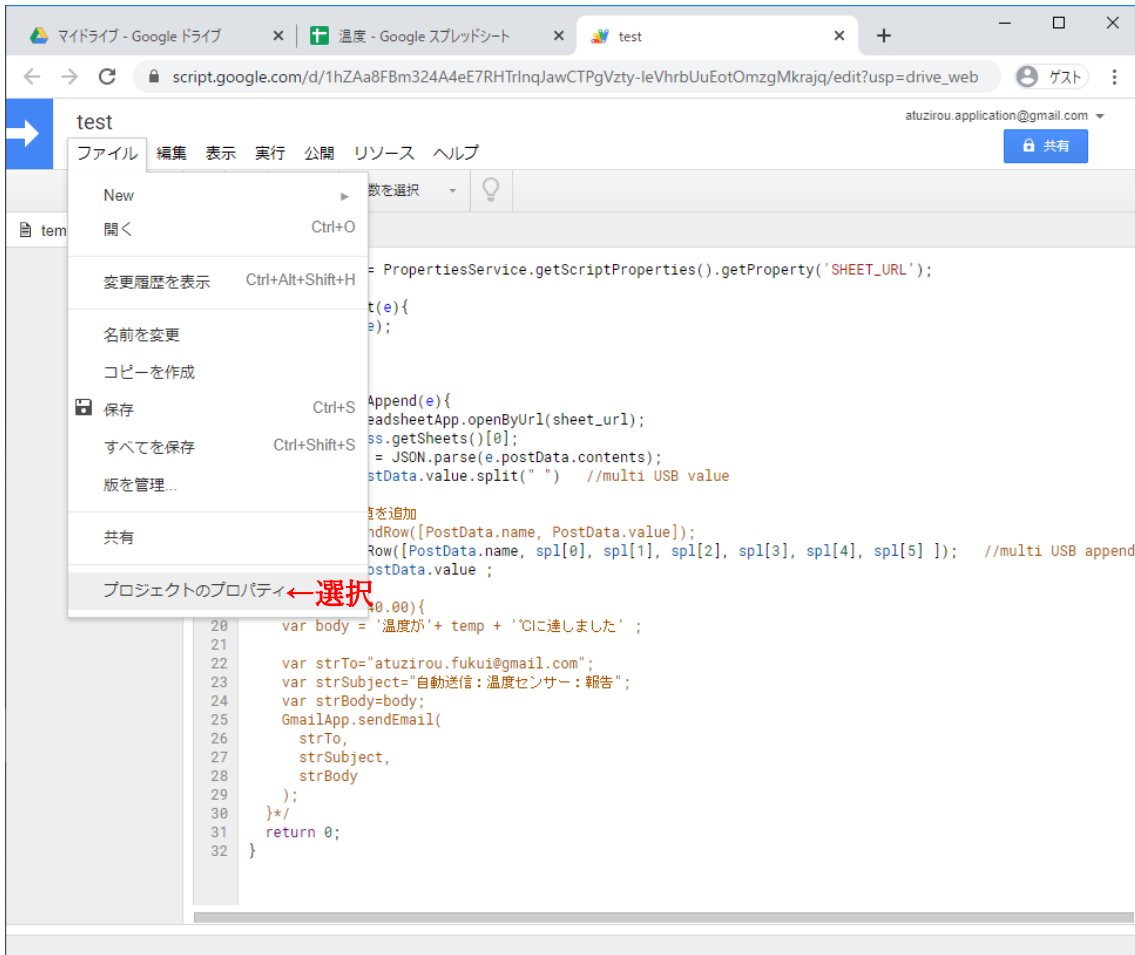
このリンクを知っているインターネット上の全員が閲覧できます

閲覧者 ▼

[ご意見](#)

完了

GAS 側で Sheet の URL をライブラリに記述しておく。ファイルからプロジェクトのプロパティを選択。



The screenshot shows the Google Apps Script editor interface. The 'File' menu is open, displaying options like 'New', 'Open', 'Save', and 'Project Properties'. A red arrow points to the 'Project Properties' option, with the text 'プロジェクトのプロパティ ← 選択' (Project Properties ← Select) next to it. The background shows a script for a temperature sensor project, with variables like 'temp' and 'body' being used to send data via email.

行を追加し、プロパティ名と値を入力。プロパティ名はコードの一行目にある

`var sheet_url = PropertiesService.getScriptProperties().getProperty('SHEET_URL');`
の最後の部分。

プロジェクトのプロパティ

プロパティ	値	
SHEET_URL	https://docs.google.com/spreadsheets/d/18S4w_MST6dxrsRVfPNbDVEKV-MCH50AWs_rlyj4fygY/edit?usp=sharing	削除

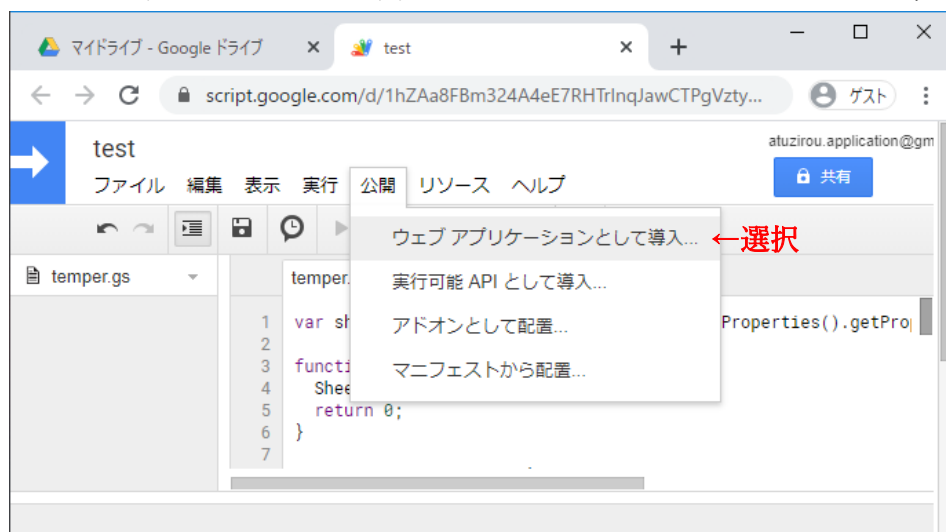
+ 行を追加

保存 キャンセル

プログラムの test を Linux からデータを送って行く。curl がインストールされていなければ以下のコマンドでインストールしておく。

```
$ sudo apt install curl
```

GAS の URL を取得する。GAS で公開タブから Web アプリケーションとして導入を選択。



Linux から実行するので誰でも実行できるように設定。

Deploy as web app

Project version:

New ▼

←更新するときは New にする

Describe what has changed...

Execute the app as:

Me (個人情報 @gmail.com) ▼

You need to authorize the script before distributing the URL.

Who has access to the app:

Anyone, even anonymous ▼

←匿名の誰でもに設定

Deploy

キャンセル

ヘルプ

Deploy すると許可の確認が出るので、権限も許可しておく。

Authorization required

test needs your permission to access your data on Google.

許可を確認

キャンセル

アカウントを選択し、詳細を表示してページに飛ぶ。



このアプリは確認されていません

このアプリは、Google による確認が済んでいません。よく知っている信頼できるデベロッパーの場合に限り続行してください。

詳細を非表示 ←表示

安全なページ



Google ではまだこのアプリを確認していないため、アプリの信頼性を保てません。未確認のアプリは、あなたの個人データを脅かす可能性があります。
[詳細](#)

[test \(安全ではないページ\) に移動](#) ←移動

test が Google アカウントへのアクセスをリクエストしています

 個人情報 @gmail.com

test に以下を許可します:

-  Gmail のすべてのメールの閲覧、作成、送信、完全な削除 ⓘ
-  Google ドライブのスプレッドシートの表示、編集、作成、削除 ⓘ

test を信頼できることを確認

機密情報をこのサイトやアプリと共有する場合があります。test の利用規約とプライバシー ポリシーで、ユーザーのデータがどのように取り扱われるかをご確認ください。アクセス権の確認、削除は、Google アカウントで行えます。

[リンクの詳細](#)

[キャンセル](#)

許可

←許可

URL が出るのでコピー

Deploy as web app

This project is now deployed as a web app.

Current web app URL:

<https://script.google.com/macros/s/AKfycbxYFEAqr455vvg7II>

←URL をコピー

Test web app for your latest code.

OK

Linux 側から curl コマンドを使って GAS に数値を送る。

```
$ curl -X POST -H "Content-Type: application/json" -d '{"name":"test", "value":"25.00"}' -L  
"https://ゆーあーるえる" -s
```

↑ 名前、 ↑ 値

↑ここに GAS の URL

動作が正常ならば、SpreadSheet の一列目に test、二列目に 25.00 と表記される。

temper で温度データを取得し、GAS に数値を飛ばすために ShellScript を使用する。

仕組みとしては、

- ① `temper` で温度読み取り & ファイルに温度書き込み。
- ② ターミナルで温度を変数として読み込み。
- ③ `curl` で変数を `GAS` に送信。

となる。実際のコードは以下のようになる。

```
#!/bin/sh ←シェバング
```

```
echo -n "TEMP=\"" > /home/tsuru02/hirota/temp.dat
```

```
/home/tsuru02/package/hidapi/linux/TEMPered/Utils/tempered 2>> /home/tsuru02/hirota/temp.dat |
```

```
/usr/bin/awk 'NR==1' | /usr/bin/cut -b 29-33
```

```
/usr/bin/truncate -s 11 /home/tsuru02/hirota/temp.dat
```

↑ temper コマンドの出力から数値の

`echo -n "\" >> /home/tsuru02/hirota/temp.dat` ↑改行の削除 部分だけをファイルに書き込む処理

```
./home/tsuru02/hirota/temp.dat
```

```
curl -X POST -H "Content-Type: application/json" -d '{"name":"test", "value":"$TEMP"}' -L
```

```
"https://XXXXXXXXXXXXXXXXXXXXXXXXXXXXX" -s
```

←GAS の URL ↑ 温度の変数

コードの内容としては、まずファイル内を「TEMP=」で上書きしその後、温度データ「25.00(例)」を追記し、最後に「」を書き込む。その結果、ファイル内には「TEMP="25.00"」などと記述される。この「変数名=値」で書き込まれたファイルは、ソースコマンド「.」で読み込むことでターミナル内でも変数が使えるようになる。この変数を、curl コマンドにエスケープを使って組み込むことで数値を送れるようになる。

curl コマンドの log は以下のようになる。

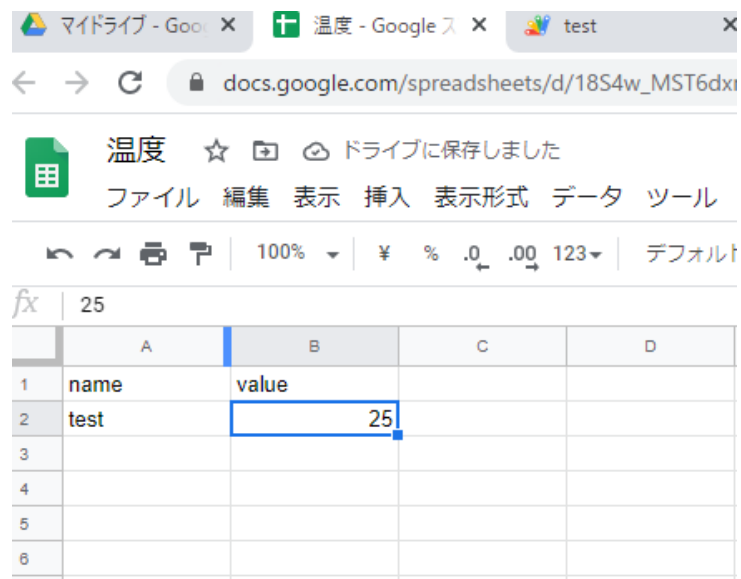
```

" href="//ssl.gstatic.com/docs/script/images/favicon.ico"><title>エラー</title><style type="text/css">
  @font-face {font-family: Arial,sans-serif; font-size: 12pt; font-weight: bold; line-height: 150%; padding: 10px 0 0 0;}
  @font-face {font-family: "Google Apps Script" src="//ssl.gstatic.com/docs/script/images/logo.png"></div><div style="text-align: center; padding: 10px 0 0 0;">
    <div style="font-size: 24pt; font-weight: bold; margin-bottom: 10px;">エラー</div>
    <div style="font-size: 12pt; font-weight: normal; margin-bottom: 10px;">スクリプトが完了しましたが、返された値はサポートされている戻り値の型ではありませんでした。

```

スクリプトが完了すれば問題なし。Linux と GAS 間の環境の違いにより戻り値でエラーが出るがプログラムは問題なく動く。

書き込まれたことが確認出来たら完了。



The screenshot shows a Google Sheet interface. The title bar says '温度' (Temperature). The menu bar includes 'ファイル' (File), '編集' (Edit), '表示' (View), '挿入' (Insert), '表示形式' (Format), 'データ' (Data), and 'ツール' (Tools). The toolbar shows a zoom level of 100%. The spreadsheet has columns A, B, C, and D, and rows 1 through 6. The data is as follows:

	A	B	C	D
1	name	value		
2	test	25		
3				
4				
5				
6				

ShellScript を定期的に実行するために Crontab コマンドを利用する。

crontab -I

すると cron の編集に移れる。行末に ShellScript 実行用のコマンドを記述する。今回は、/home/tsuru02/hirota/shell/に保存した ShellScript(temp.sh)を使用する。以下のように記述。

```
***** ./home/tsuru02/hirota/shell/temp.sh
```

先頭の文字列の 5 つの「*」は毎分、毎時、毎日、毎月、毎曜日を表す。設定し、SpreadSheet に毎分温度が記述されるようになれば成功。

※2020/07/01 記述 PC が変わっているので path が少し異なります。

USB 温度計が複数のとき、temper プログラムの実行結果は以下になる。

```
tsuru07@tsuru07:~/package/temper/TEMPered/utls$ sudo ./tempered
/dev/hidraw1 0: temperature 31.18 °C
/dev/hidraw1 1: Failed to get the temperature: Not enough data was
/dev/hidraw1 1: no sensor data available
/dev/hidraw3 0: temperature 32.00 °C
/dev/hidraw3 1: Failed to get the temperature: Not enough data was
/dev/hidraw3 1: no sensor data available
/dev/hidraw5 0: temperature 30.12 °C
/dev/hidraw5 1: Failed to get the temperature: Not enough data was
/dev/hidraw5 1: no sensor data available
/dev/hidraw7 0: temperature 31.87 °C
/dev/hidraw7 1: Failed to get the temperature: Not enough data was
/dev/hidraw7 1: no sensor data available
/dev/hidraw9 0: temperature 29.56 °C
/dev/hidraw9 1: Failed to get the temperature: Not enough data was
/dev/hidraw9 1: no sensor data available
tsuru07@tsuru07:~/package/temper/TEMPered/utls$
```

この出力結果からから温度の部分のみ取り出す。温度計が一つのときのコードに手を加え、

```
#!/bin/sh
```

```
echo -n "TEMP=\" > /home/tsuru02/hirota/temp.dat
/home/tsuru02/package/hidapi/linux/TEMPered/utls/tempered 2>> /home/tsuru02/hirota/temp.dat |
/usr/bin/awk 'NR==1 || NR==3 || NR==5 || NR==7 || NR==9' | /usr/bin/cut -b 29-33
/usr/bin/truncate -s 11 /home/tsuru02/hirota/temp.dat      ↑ 行数の処理だけ追加
echo -n "\" >> /home/tsuru02/hirota/temp.dat
. /home/tsuru02/hirota/temp.dat
curl -X POST -H "Content-Type: application/json" -d '{"name":"test", "value":"'$TEMP'" }' -L
"https://XXXXXXXXXXXXXXXXXXXXXXXXXXXXX" -s
```

とした場合、temp.dat には

```
tsuru07@tsuru07:~/hirota/shell/prog$ cat ../../temp.dat
TEMP="31.18
32.75
30.75
32.56
30.18"
```

といった形で記述される。これは Linux 上では問題ないが GAS 上で JSON として扱うときに改行部分でエラーが出る。そこで、改行部分を無くして TEMP="31.18 32.75"のようにスペース区切りのデータに修正する必要がある。

ファイル内の文字列を改行無しの一行にするために以下の C++プログラムを用意した。

USB 温度計複数時の変更

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string.h>
using namespace std;

int main(int argc, char** argv){
    string fin_name = "/home/tsuru07/hirota/temp.dat";
    if(argc != 2){
        cerr << "SynErr" << endl;
        return 1;
    }
    string fout_name = argv[1];

    ifstream fin(fin_name.data());
    ofstream fout(fout_name.data());
    if (!fin) {
        cerr << "Error: don't exist such a data file, " << fin_name << " !" <<
endl;
        return -1;
    }
    string line;
    double X;
    fout << "TEMP=\n";
    while(getline(fin, line)) {
        if (line.empty() || (line[0] == '#')) {
            continue;
        }
        istringstream(line) >> X;
        fout << X << " ";
        ←行を読み、スペース
        区切りで書き込み。
    }
    fout << "\n";
    fin.close();
    fout.close();
    return 0;
}
```

USB 温度計複数時の変更

ShllScript の出力部分の「TEMP=」の部分削除するため以下のように変更する。

```
echo -n "TEMP=\"\" > /home/tsuru02/hirota/temp.dat
/home/tsuru02/package/hidapi/linux/TEMPred/utils/tempered 2>> /home/tsuru02/hirota/temp.dat |
/usr/bin/awk 'NR==1 || NR==3 || NR==5 || NR==7 || NR==9' | /usr/bin/cut -b 29-33
/usr/bin/truncate -s 11 /home/tsuru02/hirota/temp.dat
echo -n "\" >> /home/tsuru02/hirota/temp.dat
```



```
/home/tsuru02/package/hidapi/linux/TEMPred/utils/tempered | /usr/bin/awk 'NR==1 || NR==3 ||
NR==5 || NR==7 || NR==9' | /usr/bin/cut -b 29-33 > /home/tsuru02/hirota/temp.dat
```

Shell で作成される温度データは数値のみの以下ようになる。

```
1 32.87
2 33.62
3 31.43
4 33.81
5 31.06
6
```

このファイルを先ほど作った C++プログラムで読み込むと以下のように変換される。(実行ファイル名は temp_to_TEMP とした。)

```
$ ./temp_to_TEMP temp.dat
```

```
1 TEMP="32.87 33.62 31.43 33.81 31.06 "
```

以上の出力部分の変更および、temp_to_TEMP の実行を組み込んだ最終的な ShellScript は以下になる。

```
#!/bin/bash

/home/tsuru07/package/temper/TEMPred/utils/tempered | /usr/bin/awk 'NR==1 || NR==3 || NR==5 ||
NR==7 || NR==9' | /usr/bin/cut -b 29-33 >/home/tsuru07/hirota/temp.dat
/home/tsuru07/hirota/shell/prog/temp_to_TEMP /home/tsuru07/hirota/shell/prog/TEMP.dat
./home/tsuru07/hirota/shell/prog/TEMP.dat
curl -X POST -H "Content-Type: application/json" -d '{"name":"tsuru07","value":"'$TEMP'"}' -L
"https://XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX" -s
echo -n "\n"
```

GAS 側でも複数の温度計に対応するように変更する必要がある。

```
var sheet_url = PropertiesService.getScriptProperties().getProperty('SHEET_URL');

function doPost(e){
  SheetAppend(e);
  return 0;
}

function SheetAppend(e){
  var ss = SpreadsheetApp.openByUrl(sheet_url);
  var sheet = ss.getSheets()[0];
  var PostData = JSON.parse(e.postData.contents);
  var spl = PostData.value.split(" ") //multi USB value ←スペース区切りのデータを配列に変換

  // 行の最後に値を追加
  // sheet.appendRow([PostData.name, PostData.value]);
  sheet.appendRow([PostData.name, spl[0], spl[1], spl[2], spl[3], spl[4] ]);

  var temp = spl[0] ;
  /*
  if (temp >= 40.00){
    var body = '温度が'+ temp + '°Cに達しました';

    var strTo="XXX@gmail.com";
    var strSubject="自動送信：温度センサー：報告";
    var strBody=body;
    GmailApp.sendEmail(
      strTo,
      strSubject,
      strBody
    );
  }*/
  return 0;
}
```

USB 温度計複数時の変更

ShellScript が実行され温度が書き込まれれば正常。



The image shows a screenshot of a spreadsheet application. The menu bar includes options like 'ファイル' (File), '編集' (Edit), '表示' (View), '挿入' (Insert), '表示形式' (Format), 'データ' (Data), 'ツール' (Tools), 'アドイン' (Add-ins), and 'ヘルプ' (Help). The toolbar contains icons for undo, redo, print, and zoom, along with a zoom percentage of 100%. The spreadsheet has columns labeled A through F. Row 1 contains headers 'name' and 'value'. Row 2 contains the data 'tsuru07' and '32.87', with other columns showing values like 33.62, 31.43, 33.81, and 31.06.

	A	B	C	D	E	F
1	name	value				
2	tsuru07	32.87	33.62	31.43	33.81	31.06