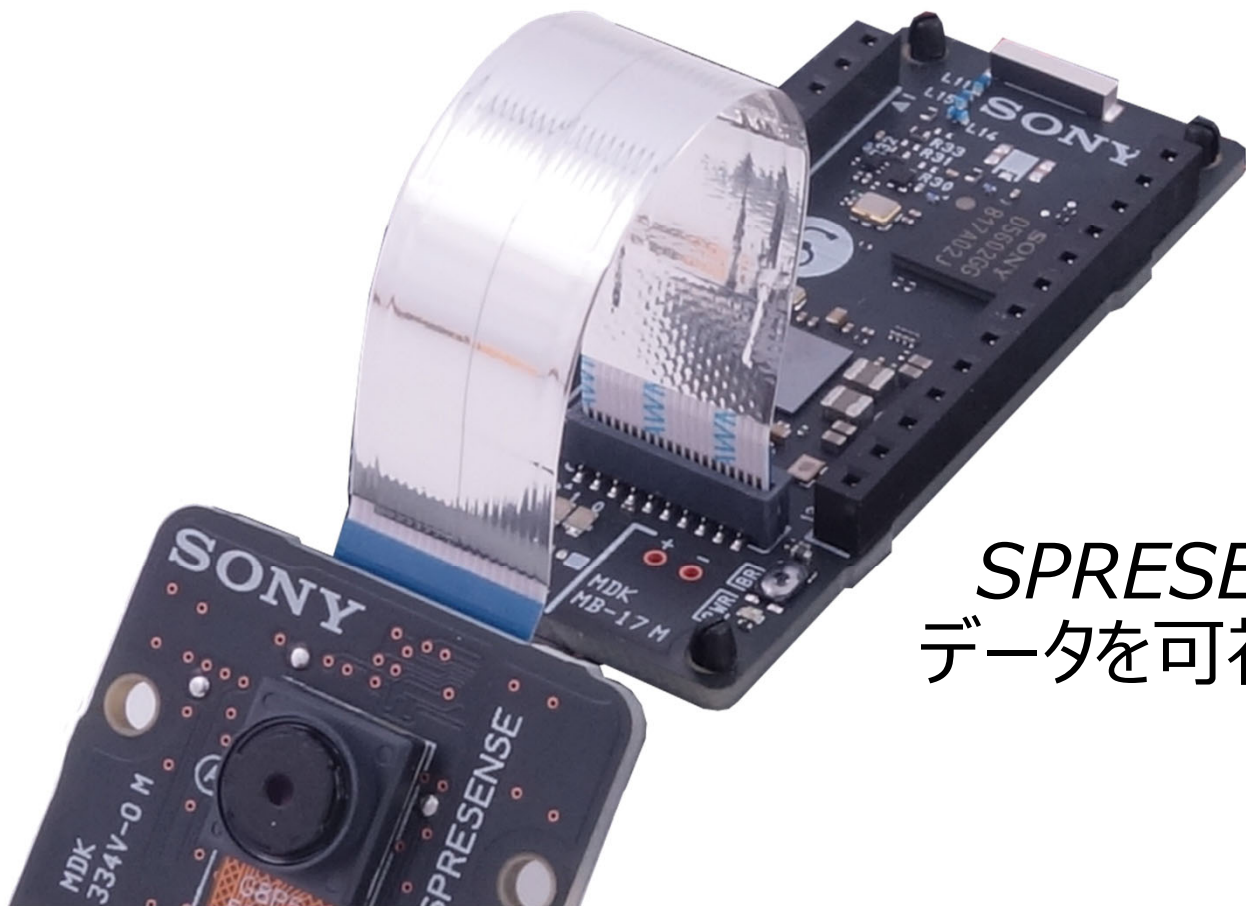
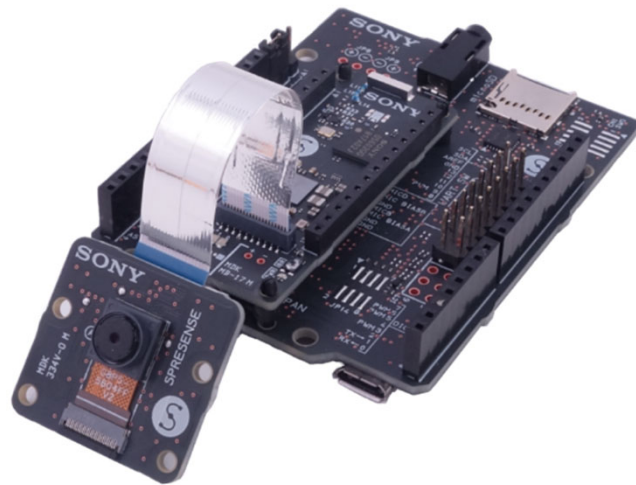




SPRESENSE™ からの
データを可視化してみよう！

Agenda

- Spresense Add-on・拡張ボードご紹介
- Spresense LTEを使ってみよう！
- WiFi Add-onを使ってみよう！
- BLE Add-onを使ってみよう！
- **Spresenseからのデータを可視化してみよう！**
 - Ambientを使ってみよう！
 - kintoneを使った可視化をしてみよう！
 - フリーブローカを使ってローカルPCで可視化しよう！
- SpresenseでいろんなLPWAを使ってみよう！



データの可視化の必要性

改めて言うまでもないですが、データは**人が理解する**ことで初めて**価値を持つ**ものであり、目に見えて分かる形にされていないデータは価値がありません。（のちに利用するとしてもその時点で可視化されます。）

データの可視化とは、単なる数値だけでは理解しにくい現象や関係性を**ひと目で分かる形**（グラフ・チャート・表）に表すことを指します。

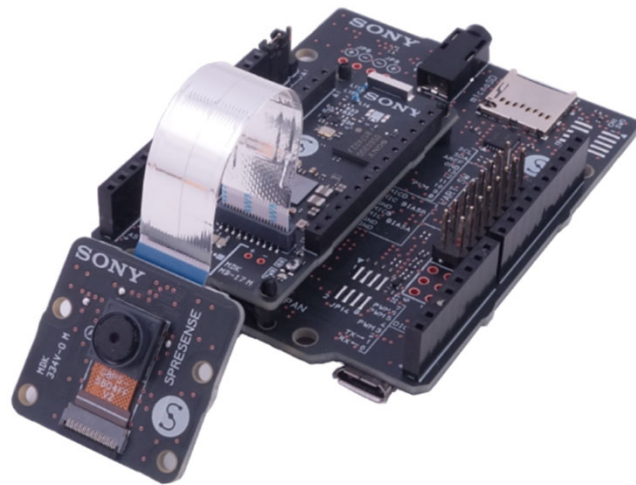
Spresenseで使える無料での可視化ツール

- **Ambient**
- AWS
- Azure
- **kintone**
- ~~Machinist @IIT~~
- ~~モノプラットフォーム @SAKURA Internet~~
- Harvest @SORACOM
- **フリーのMQTTブローカーを使った可視化**

etc...

Agenda

- Spresense Add-on・拡張ボードご紹介
- Spresense LTEを使ってみよう！
- WiFi Add-onを使ってみよう！
- BLE Add-onを使ってみよう！
- **Spresenseからのデータを可視化してみよう！**
 - **Ambientを使ってみよう！**
 - kintoneを使った可視化をしてみよう！
 - フリーブローカを使ってローカルPCで可視化しよう！
- SpresenseでいろんなLPWAを使ってみよう！





Ambientは無料で使いやすいツールです。

□ケーション情報も含め手軽に
可視化できます。

Ambient 使用の流れ

Ambient には、チュートリアルがあり、これを見ながら作業が可能です。

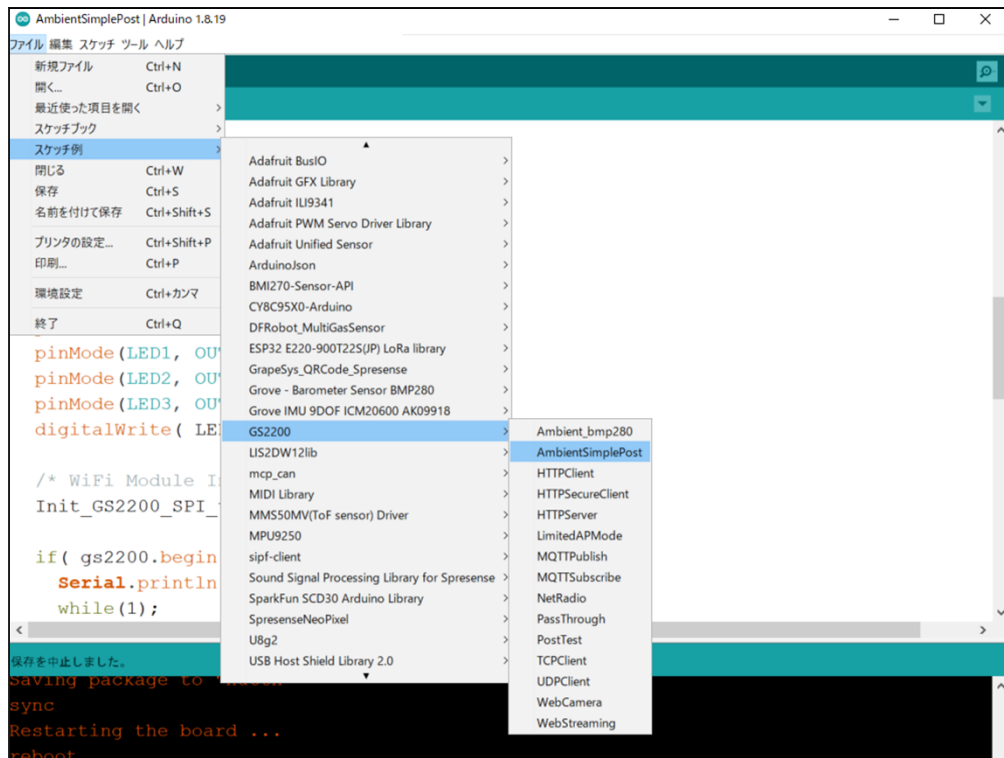
まずは、 **Ambientを使ってみる** を参考に進めます。(<https://ambidata.io/docs/gettingstarted/>)

Ambientを使う大まかな流れは次のようになります。

1. ユーザー登録(無料)
2. チャンネル生成
3. マイコン側プログラミング
4. データ送信
5. 可視化(グラフ化)

Ambientではマイコンからデータを送ってグラフ化するまでは非常に簡単です。最低限必要なのはAmbientサイトで「チャンネル」を作ること、マイコン側のプログラムでチャンネルIDとライトキーを指定してデータを送ることだけです。チャンネルの名前やどんなマイコンが繋がっているか、どんなセンサーのどんなデータを送るかといった属性情報は後から記述することができますが、複雑なことは後回しにして、とにかく最初は簡単に始められます。

SpresenseのデータをAmbientに送ってみよう！



Ambient は、Ambient が用意しているキーにJSON形式でデータを入れてHTTPでPOSTすることで可視化しています。そのため、HTTPでPOSTできれば、どのような通信プラットフォーム（LTE、WiFi、etc...）でも構いません。

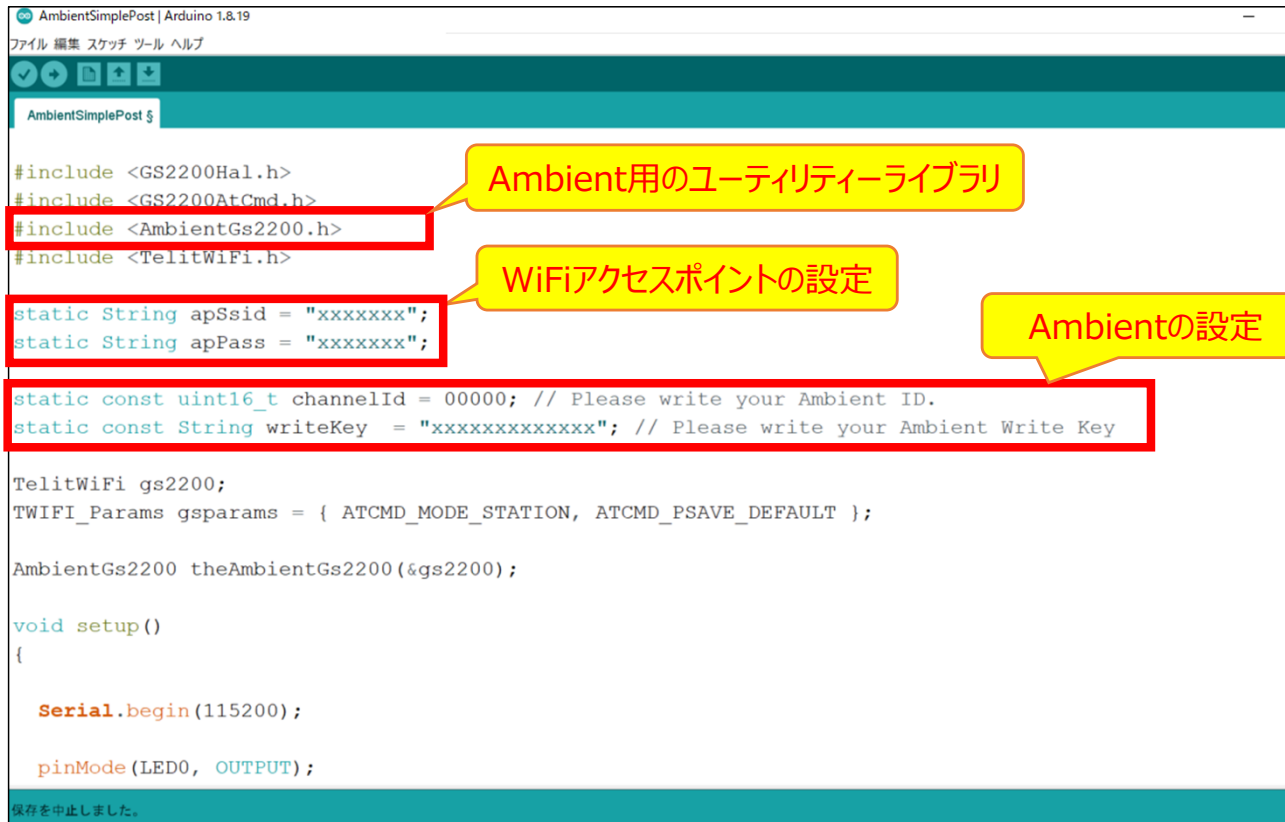
今回は、WiFiを使って試してみます。

GS2200ライブラリの

AmbientSimplePost.ino

を使います。

AmbientSimplePostのコード解説



The screenshot shows the Arduino IDE interface with the file 'AmbientSimplePost.ino' open. The code is as follows:

```
#include <GS2200Hal.h>
#include <GS2200AtCmd.h>
#include <AmbientGs2200.h>
#include <TelitWiFi.h>

static String apSsid = "xxxxxxx";
static String apPass = "xxxxxxx";

static const uint16_t channelId = 00000; // Please write your Ambient ID.
static const String writeKey = "xxxxxxxxxxxxx"; // Please write your Ambient Write Key

TelitWiFi gs2200;
TWIFI_Params gsparams = { ATCMD_MODE_STATION, ATCMD_PSAVE_DEFAULT };

AmbientGs2200 theAmbientGs2200(&gs2200);

void setup()
{
    Serial.begin(115200);

    pinMode(LED0, OUTPUT);
}
```

Annotations in the image:

- A yellow callout bubble points to the `#include <GS2200Hal.h>` and `#include <GS2200AtCmd.h>` lines, containing the text: **Ambient用のユーティリティライブラリ**
- A red box highlights the `#include <AmbientGs2200.h>` line.
- A yellow callout bubble points to the `static String apSsid = "xxxxxxx";` and `static String apPass = "xxxxxxx";` lines, containing the text: **WiFiアクセスポイントの設定**
- A yellow callout bubble points to the `static const uint16_t channelId = 00000;` and `static const String writeKey = "xxxxxxxxxxxxx";` lines, containing the text: **Ambientの設定**
- A red box highlights the `static const uint16_t channelId = 00000;` and `static const String writeKey = "xxxxxxxxxxxxx";` lines.

At the bottom of the IDE window, a status bar indicates: 保存を中止しました。

AmbientSimplePostのコード解説



The screenshot shows the Arduino IDE interface with the file 'AmbientSimplePost.ino' open. The code is for an Arduino Uno (1.8.19). The code is as follows:

```
/* WiFi Module Initialize */
Init GS2200 SPI type(iS110B TypeC)

if( gs2200.begin( gsparams ) ){
  Serial.println( "GS2200 Initilization Fails" );
  while(1);
}

/* GS2200 Association to AP */
if( gs2200.activate_station( apSsid, apPass ) ){
  Serial.println( "Association Fails" );
  while(1);
}

digitalWrite( LED0, LOW );
digitalWrite( LED1, HIGH );

Serial.println(F("GS2200 Initialized"));

theAmbientGs2200.begin(channelId, writeKey);

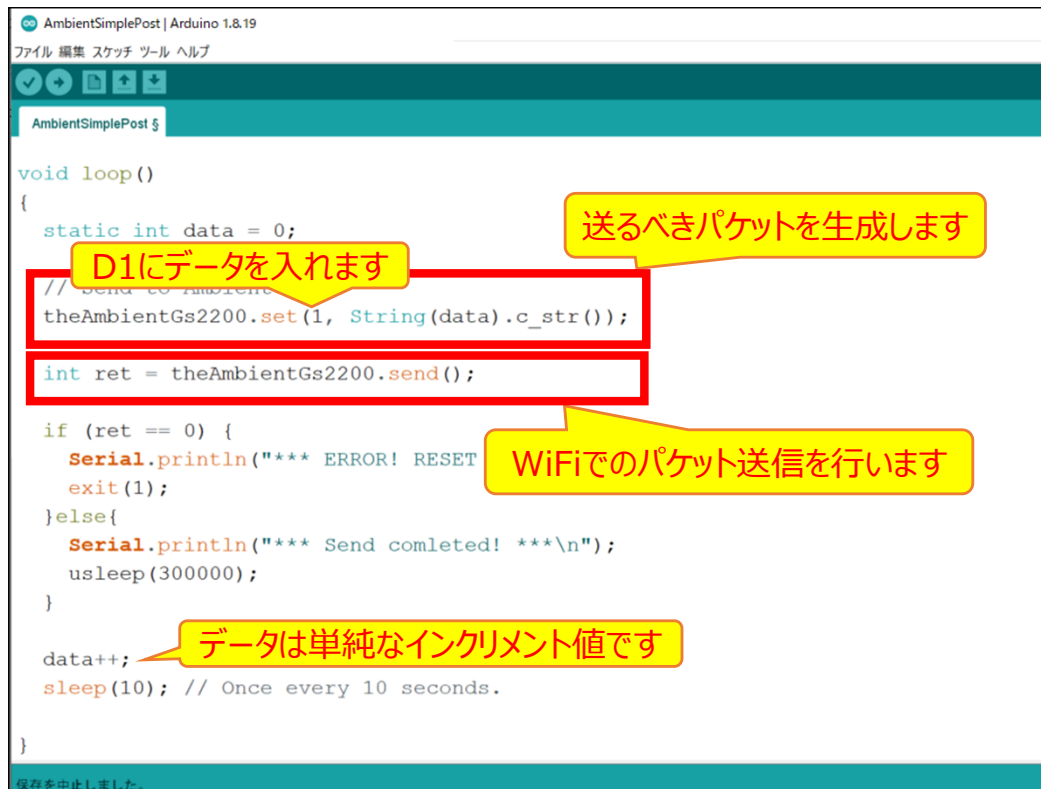
Serial.println(F("Ambient Initialized"));
}
```

Annotations (yellow callouts) are present:

- Type気を付けてください** (Please be careful of the type) points to the line `Init GS2200 SPI type(iS110B TypeC)`.
- HTTPクライアントの設定** (HTTP client settings) points to the `gs2200.activate_station` function call.
- Ambientライブラリの初期化** (Initialization of the Ambient library) points to the line `theAmbientGs2200.begin(channelId, writeKey);`.

The status bar at the bottom indicates '保存を中止しました。' (Save cancelled).

AmbientSimplePostのコード解説



The screenshot shows the Arduino IDE interface with the file 'AmbientSimplePost.ino' open. The code is for an Arduino Uno (ATmega328P) and is written in C++. The code is as follows:

```
void loop()
{
  static int data = 0;
  // Send to Ambient
  theAmbientGs2200.set(1, String(data).c_str());

  int ret = theAmbientGs2200.send();

  if (ret == 0) {
    Serial.println("*** ERROR! RESET");
    exit(1);
  }else{
    Serial.println("*** Send comleted! ***\n");
    usleep(300000);
  }

  data++;
  sleep(10); // Once every 10 seconds.
}
```

Annotations in the image:

- D1にデータを入れます** (D1 is where the data is entered) - points to the `String(data)` argument in the `set` method.
- 送るべきパケットを生成します** (Generate the packet to be sent) - points to the `set` method call.
- WiFiでのパケット送信を行います** (Perform packet transmission via WiFi) - points to the `send` method call.
- データは単純なインクリメント値です** (Data is a simple increment value) - points to the `data++` line.

The status bar at the bottom indicates '保存を中止しました。' (Save aborted).

データは、D1～D8まで使えます。
D9、D10は、緯度、経度になります。

入れるべきデータは、

`bool set(int field, const char * data);`
`bool set(int field, double data);`
`bool set(int field, int data);`

文字列、整数、少数が可能です。

うまく動かない時に！

- WiFiのタイプはありますか？
- WiFiのアクセスポイントはありますか？
- AmbientのチャンネルID、write keyはありますか？

LTEでも同様に可能です！

- 太田さん作成のLTE向けAmbientライブラリがあります。

https://github.com/TE-YoshinoriOota/Ambient_SpresenseLTEM

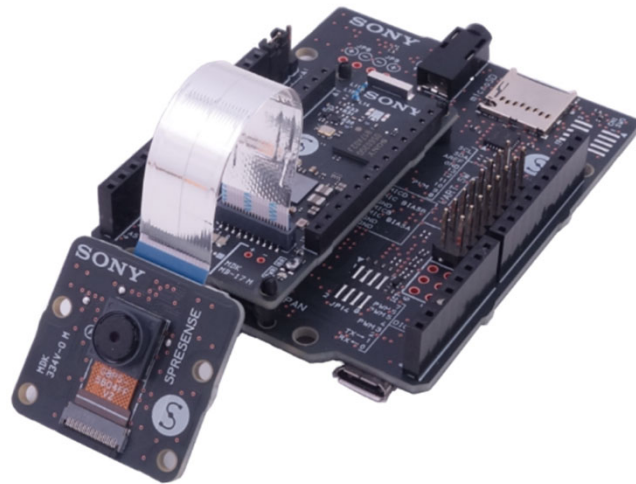
- LTEを使ったCO2監視サンプルがこちらにあります。

[CO2 sensing to Ambient.ino](#)

ご参考に。

Agenda

- Spresense Add-on・拡張ボードご紹介
- Spresense LTEを使ってみよう！
- WiFi Add-onを使ってみよう！
- BLE Add-onを使ってみよう！
- **Spresenseからのデータを可視化してみよう！**
 - Ambientを使ってみよう！
 - **kintoneを使った可視化をしてみよう！**
 - フリーブローカを使ってローカルPCで可視化しよう！
- SpresenseでいろんなLPWAを使ってみよう！



kintone をつかってデータ収集と可視化をしてみよう！

ご紹介していたキャリアの提供しているIoTプラットフォームですが、最近、低調気味です。

- ~~Machinist @IIJ~~：廃止
- ~~モノプラットフォーム @SAKURA Internet~~：停止
- Harvest @SORACOM

その中で、今回は、一般的には業務管理ツールのイメージのある kintone をIoTに向けて使ってみる方法をご紹介します。



◆ 現場の業務にフィットする

自分たちで業務 アプリがつかれる

キントーンは、プログラミングの知識がなくても、ノーコードで業務のシステム化や効率化を実現するアプリがつかれるクラウドサービスです。

散在するエクセルや、煩雑なメール、紙の書類の山、バラバラなシステムなど、業務を非効率にしている困りごとを解決できます。



kintone の開発ライセンス

 cybozu developer network

はじめにkintoneGarooncybozu.com共通管理チュートリアルイベントコミュニティ

kintone開発者ライセンス（開発環境）

kintone開発環境の必要性

kintoneでシステムを構築および運用する際に、設定変更の影響を試することができる開発環境の利用を推奨します。

JavaScriptカスタマイズなど、設定の変更を本番環境に直接適用する運用の場合、適用した変更が本番環境に予期せぬ影響を与える恐れがあります。

適用したい変更の動作確認を事前に開発環境で行うことで、より安全なシステム構築および運用を実現できます。

kintone開発者ライセンスとは

kintone開発者ライセンスは、kintone APIを使った開発を目的として、利用できる環境です。

本運用での利用はできません。

APIを使ったアプリの開発を目的とした場合、開発ライセンスが1年間無料で使えます。
（制限はあります。）

お試しであれば、こちらが使えます。

<https://cybozu.dev/ja/kintone/developer-license/>

kintone でのデータ収集（アプリ開発）

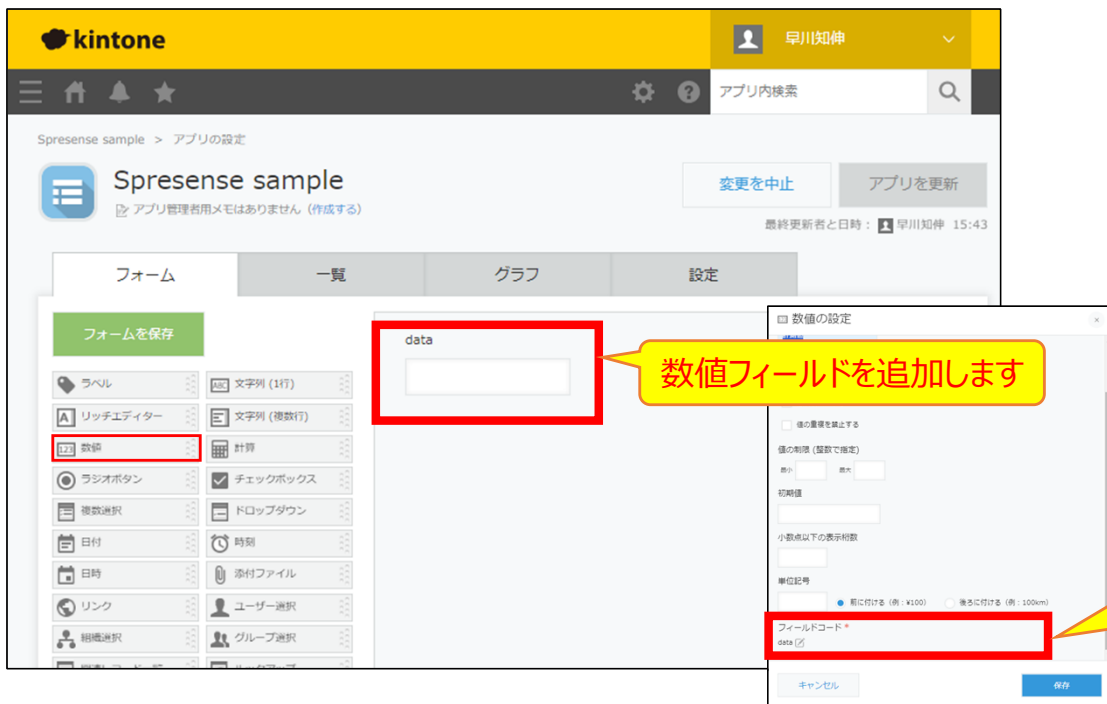
基本的には、kintoneを使ったアプリ開発は、こちらを参照に実施します。

<https://kintone.cybozu.co.jp/seminar/ondemand.html>



kintone でのアプリ開発

今回のサンプルとしては、単純な数値データフィールドを一つだけ作成します。

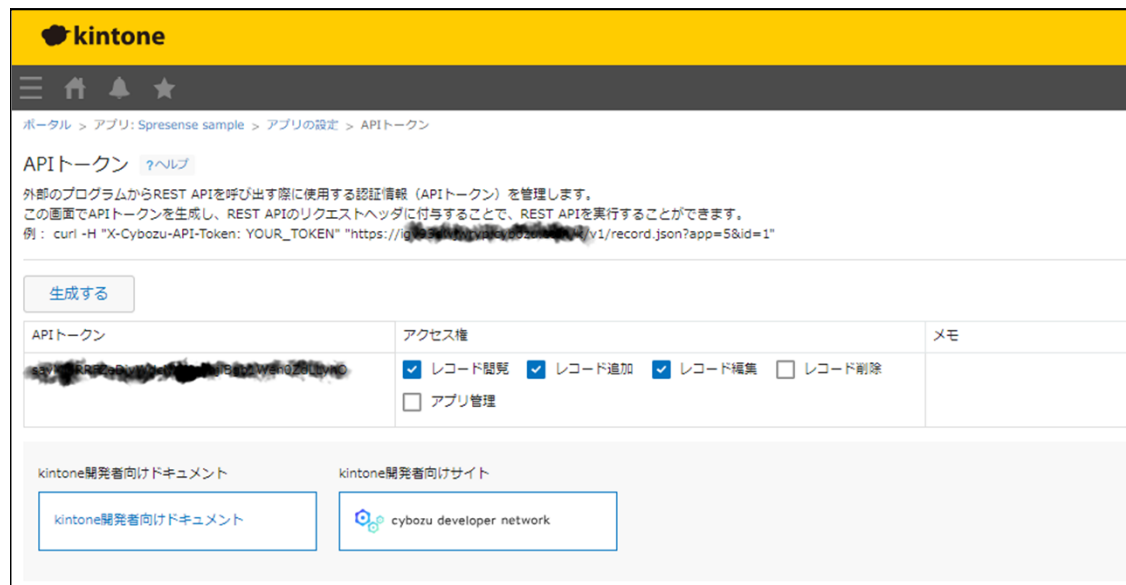


Spresenseを使ったkintoneアプリのブログはこちらにあります。

https://kintone-geeks.hatenablog.com/entry/SonySpresenseLTE_to_kintone

kintone のAPIトークンの準備

アプリを作成したら、APIトークンキーを生成します。



kintone

ポータル > アプリ: Spresense sample > アプリの設定 > APIトークン

APIトークン [ヘルプ](#)

外部のプログラムからREST APIを呼び出す際に使用する認証情報（APIトークン）を管理します。
この画面でAPIトークンを生成し、REST APIのリクエストヘッダに付与することで、REST APIを実行することができます。
例: `curl -H "X-Cybozu-API-Token: YOUR_TOKEN" "https://ip:port/kintone/v1/record.json?app=5&id=1"`

[生成する](#)

APIトークン	アクセス権	メモ
[Redacted Token]	☒ レコード閲覧 ☒ レコード追加 ☒ レコード編集 ☐ レコード削除 ☐ アプリ管理	

kintone開発者向けドキュメント kintone開発者向けサイト

[kintone開発者向けドキュメント](#) [cybozu developer network](#)

このAPIトークンは、クラウドアクセスに使いますが、誰でもアクセス可能になってしまいますので、キーの扱いには気を付けてください。

問題が発生した場合は、そのキーを破棄して、新たにキーを生成することで回避可能です。

SpresenseからLTE経由でデータを送ってみよう！

Kintoneの準備ができましたら、LTE経由でSpresenseからhttpsでデータを送るサンプルを動かしてみましよう。

このサンプルは、こちらにあります。

https://github.com/TomonobuHayakawa/Spresense-Playground/blob/master/ForHandsOn/DataVisualization/forKintone/viaLTE/sample_for_visualization/sample_for_visualization.ino

こちらも単純なインクリメント値を送信するサンプルになります。

1分ごとにデータを送ります。（kintone上での時刻がかぶらないようにしています。）

基本は、LTEの LteHttpSecureClient.ino になります。こちらの使い方に関しては、LTEのハンズオンを参照してください。

SIMは、MEEQのSIMを使用しています。お使いのSIMに合わせて修正してください。

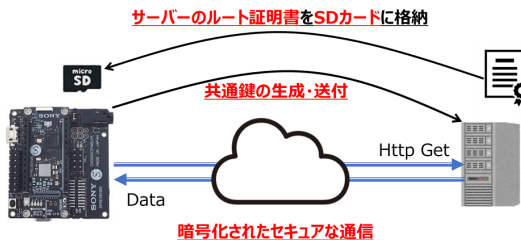
Cybozuサーバの証明書の取得

Kintoneのサーバには、セキュアなアクセスしかできませんので、証明書の取得が必要です。
こちらは、LteHttpSecureClient.ino の時と同じです。

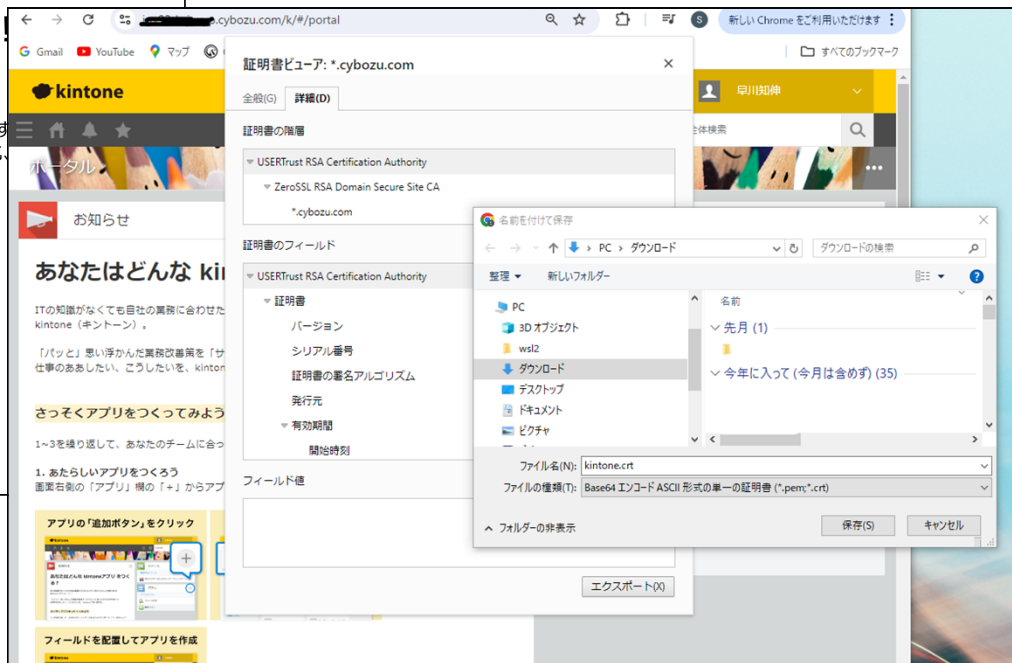
Sony Semiconductor Solutions Corporation © 2019 – 2023

セキュアなWebサーバからデータを取得してみよう (TLSクライアントでの接続)

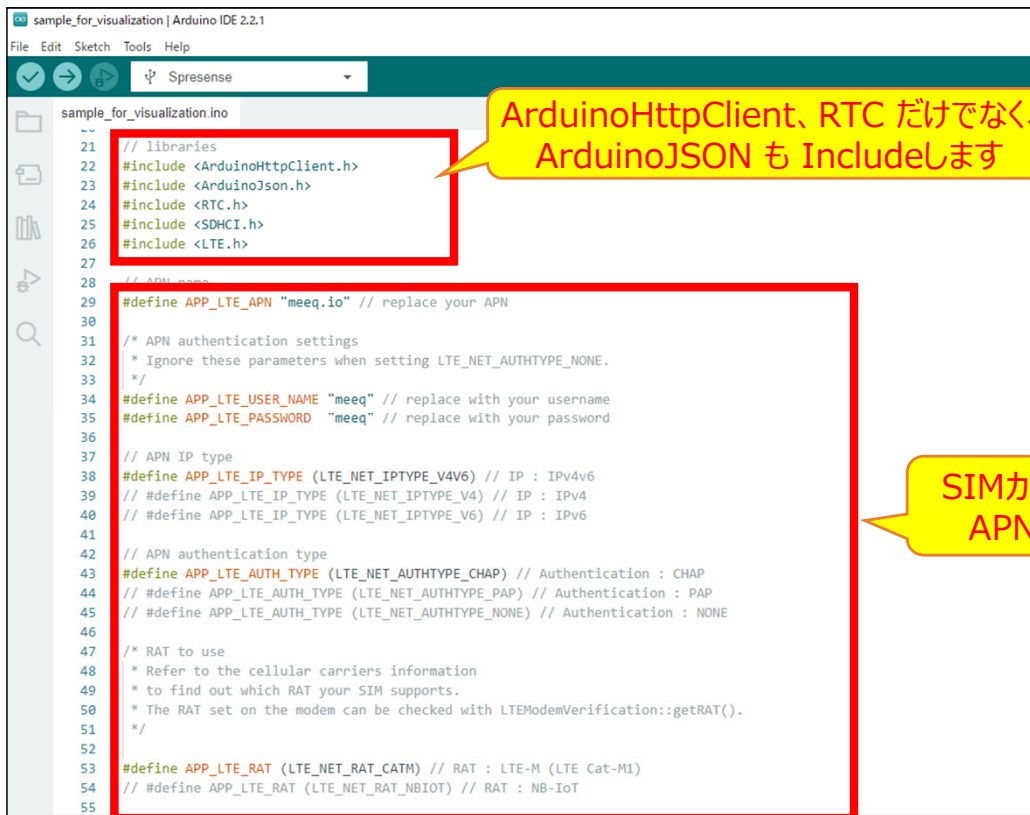
実際のインターネットサービスは、ほとんどの通信が暗号化によりセキュリティが施されています。ほとんどのサイトが <https://> で始まっており、[SSL/TLS](#) を利用した通信データの暗号化、止、なりすまし防止を行っています。



kintone.crt という名前でSDカードの
ルートに保存してください。



sample_for_visualizationのコード解説



```
sample_for_visualization | Arduino IDE 2.2.1
File Edit Sketch Tools Help

sample_for_visualization.ino
~
21 // libraries
22 #include <ArduinoHttpClient.h>
23 #include <ArduinoJson.h>
24 #include <RTC.h>
25 #include <SDHCI.h>
26 #include <LTE.h>
27
28 // APN name
29 #define APP_LTE_APN "meeq.io" // replace your APN
30
31 /* APN authentication settings
32  * Ignore these parameters when setting LTE_NET_AUTHTYPE_NONE.
33  */
34 #define APP_LTE_USER_NAME "meeq" // replace with your username
35 #define APP_LTE_PASSWORD "meeq" // replace with your password
36
37 // APN IP type
38 #define APP_LTE_IP_TYPE (LTE_NET_IPTYPE_V4V6) // IP : IPv4v6
39 // #define APP_LTE_IP_TYPE (LTE_NET_IPTYPE_V4) // IP : IPv4
40 // #define APP_LTE_IP_TYPE (LTE_NET_IPTYPE_V6) // IP : IPv6
41
42 // APN authentication type
43 #define APP_LTE_AUTH_TYPE (LTE_NET_AUTHTYPE_CHAP) // Authentication : CHAP
44 // #define APP_LTE_AUTH_TYPE (LTE_NET_AUTHTYPE_PAP) // Authentication : PAP
45 // #define APP_LTE_AUTH_TYPE (LTE_NET_AUTHTYPE_NONE) // Authentication : NONE
46
47 /* RAT to use
48  * Refer to the cellular carriers information
49  * to find out which RAT your SIM supports.
50  * The RAT set on the modem can be checked with LTEModemVerification::getRAT().
51  */
52
53 #define APP_LTE_RAT (LTE_NET_RAT_CATM) // RAT : LTE-M (LTE Cat-M1)
54 // #define APP_LTE_RAT (LTE_NET_RAT_NBIOT) // RAT : NB-IoT
55
```

ArduinoHttpClient、RTC だけでなく、
ArduinoJSON も Includeします

JSONデータを送る必要があ
りライブラリを使用しています

SIMカードに合わせた
APN情報を入力

sample_for_visualizationのコード解説

The screenshot shows the Arduino IDE interface with the file `sample_for_visualization.ino` open. The code is as follows:

```
55
56 // URL, path & port (for example: httpbin.org)
57 char server[] = "xxxxxxx.cybozu.com"; // Set your server.
58 char path[] = "/k/v1/record.json";
59 char token[] = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"; // Set API token.
60 int port = 443; // port 443 is the default for HTTPS
61
62 #define ROOTCA_FILE "kintone.der" // Define the path to a file containing CA
63 // certificates that are trusted.
64 #define CERT_FILE "path/to/certfile" // Define the path to a file containing certificate
65 // for this client, if required by the server.
66 #define KEY_FILE "path/to/keyfile" // Define the path to a file containing private key
67 // for this client, if required by the server.
68
69 // initialize the library instance
70 LTE lteAccess;
71 LTETLSClient tlsClient;
72 HttpClient client = HttpClient(tlsClient, server, port);
73 SDClass theSD;
74
```

Annotations (yellow callouts) explain the code:

- サーバのパスは共通** (Server path is common) points to the `HttpClient` constructor.
- あなたのkintoneサーバのアドレスを設定** (Set your Kintone server address) points to `server`.
- あなたのAPIトークンを設定** (Set your API token) points to `token`.
- 証明書のパス情報** (Certificate path information) points to `ROOTCA_FILE`.
- 以下の部分は、LteHttpSecureClient.inoと一緒に** (The following part is used together with LteHttpSecureClient.ino) points to the initialization of `lteAccess`, `tlsClient`, and `client`.

sample_for_visualizationのコード解説

```
sample_for_visualization | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Spsense
sample_for_visualization.ino
234
235 void loop()
236 {
237   String contentType = "application/json";
238
239   JsonDocument doc;
240   String output;
241   static float dmy_value = 0.25;
242
243   dmy_value = dmy_value + 0.11;
244   dmy_value = (dmy_value > 2.5) ? (dmy_value - 2.5) : dmy_value;
245
246   doc["app"].set(5);
247   JsonObject doc1 = doc["record"].to<JsonObject>();
248   JsonObject doc2 = doc1["data"].to<JsonObject>();
249   doc2["value"].set(dmy_value);
250
251   serializeJson(doc, output);
252
253   Serial.print("Send: ");
254   Serial.println(output);
255
256   client.beginRequest();
257   client.post(path);
258   client.sendHeader("X-Cybozu-API-Token", token);
259   client.sendHeader("Content-Type", contentType);
260   client.sendHeader(HTTP_HEADER_CONTENT_LENGTH, output.length());
261   client.endRequest();
262   client.write((const byte*)output.c_str(), output.length());
263   Serial.println("Request has ended");
264
265   // read the status code and body of the response
266   int statusCode = client.responseStatusCode();
267   String response = client.responseBody();
268
269   Serial.print("Status code: ");
270   Serial.println(statusCode);
271   Serial.print("Response: ");
272   Serial.println(response);
273
274   sleep(60);
275
276 }
```

ダミーデータの更新

JSONフォーマットの詳細
はkintoneサイトを参照

データをJSONに変換

appはそのアプリに合わせた番号にする

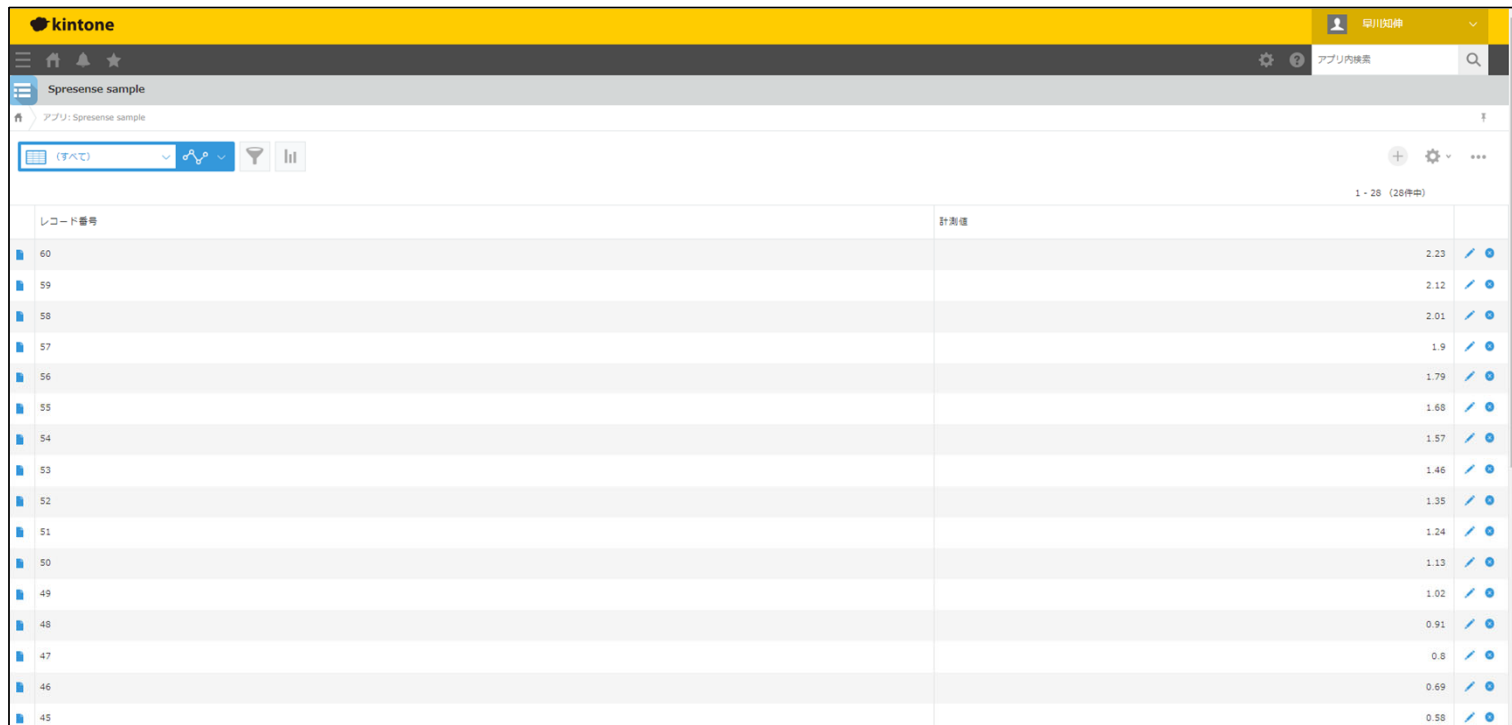
Kintoneにデータを打ち上げる

API tokenの部分で、ユニークなhttp
ヘッダーが必要なので、ArduinoHttpライ
ブラリのCustomヘッダーを使用

結果を表示

データの収集結果

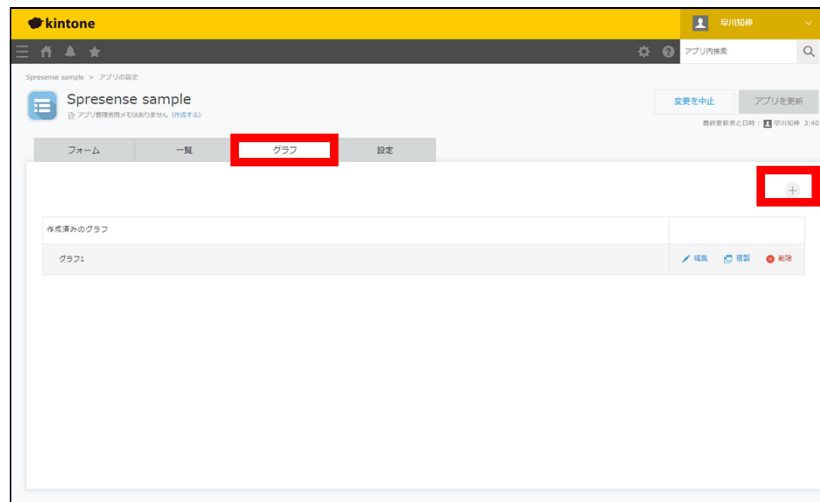
上手くいくとこのようにレコード情報が追記されていきます。



kintone		早川 陽伸	
Spresense sample		アプリ内検索	
アプリ: Spresense sample			
(すべて)		1 - 20 (28件中)	
レコード番号	計測値		
60	2.23	/	
59	2.12	/	
58	2.01	/	
57	1.9	/	
56	1.79	/	
55	1.68	/	
54	1.57	/	
53	1.46	/	
52	1.35	/	
51	1.24	/	
50	1.13	/	
49	1.02	/	
48	0.91	/	
47	0.8	/	
46	0.69	/	
45	0.58	/	

データのグラフ化

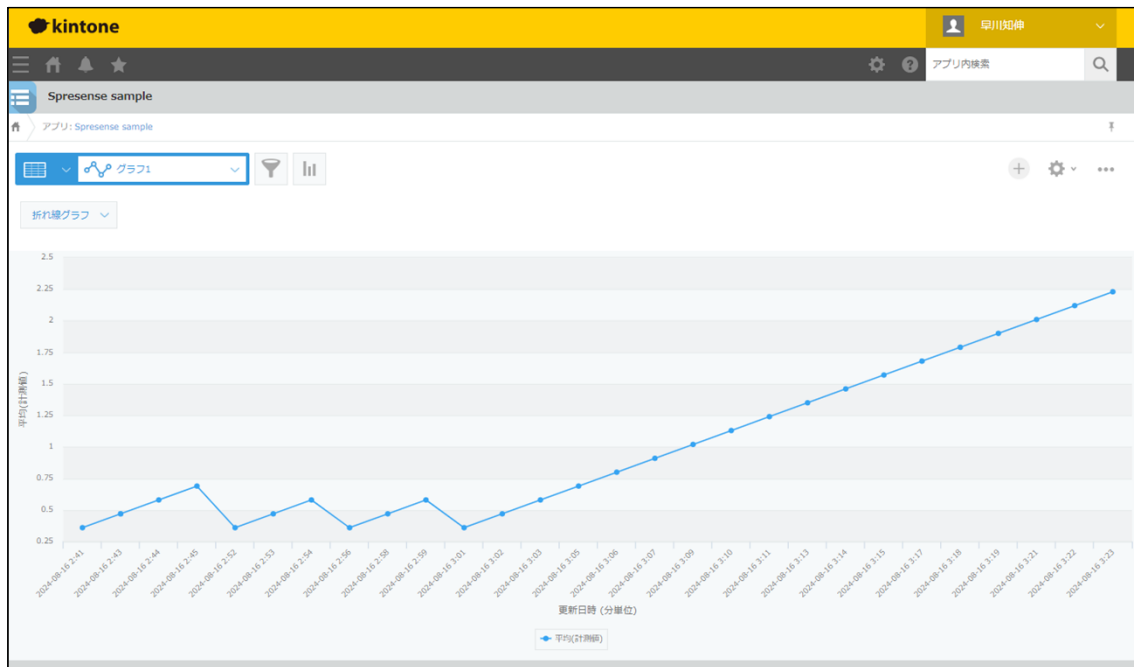
データはグラフなどを利用して可視化し、解析することで意味が出ます。
グラフ化する方法は以下になります。



設定タブの中のグラフタブを指定します。
そこで、グラフを追加します。

データのグラフ化

このような形でグラフ化できました。

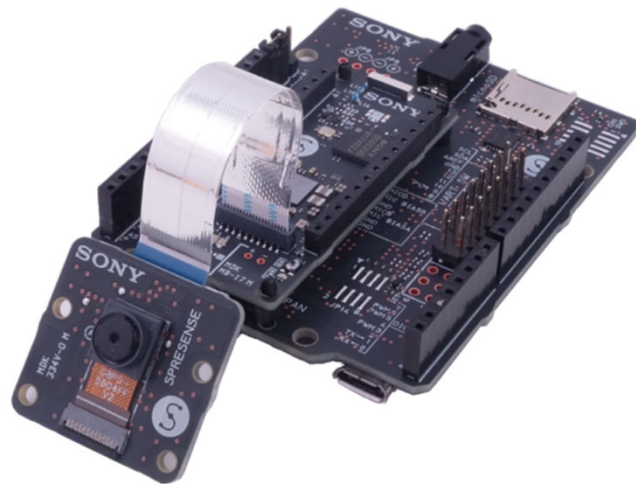


kintoneを使うことで、各種の業務アプリとの連携も可能になります。

業務のIoT/DX化に大きく貢献
できることになります。

Agenda

- Spresense Add-on・拡張ボードご紹介
- Spresense LTEを使ってみよう！
- WiFi Add-onを使ってみよう！
- BLE Add-onを使ってみよう！
- **Spresenseからのデータを可視化してみよう！**
 - Ambientを使ってみよう！
 - kintoneを使った可視化を試みよう！
 - **フリーブローカを使ってローカルPCで可視化しよう！**
- SpresenseでいろんなLPWAを使ってみよう！



フリーブローカを使ってもっと手軽にローカルPCで可視化

- 外部のサービスを使うと、
 - 契約が必要
 - フリー期間が限られている
 - 有償
- など、制約が発生する。

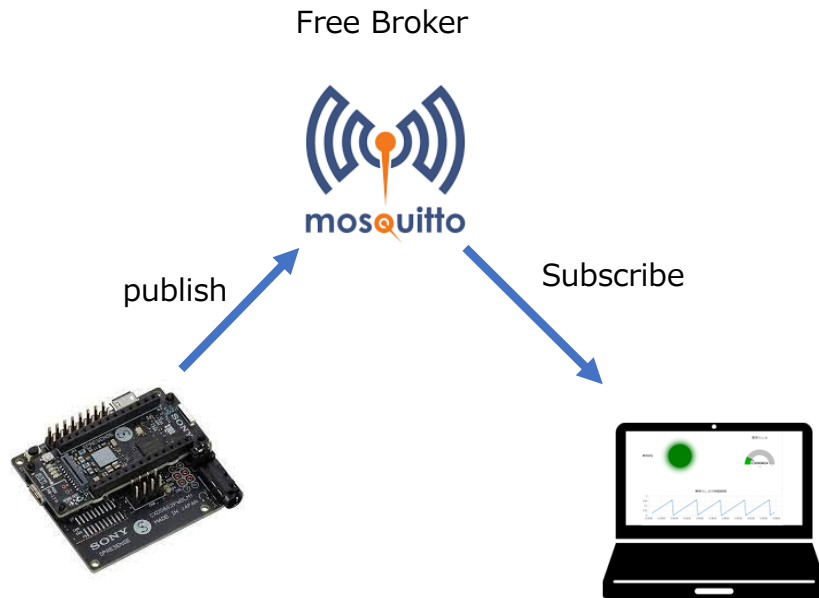


もっと手軽に見える可したい。



ローカルPCでの

フリーのMQTTブローカーを使った可視化
を説明します。



Node-REDをローカルPCにインストール



The screenshot shows the 'Node-RED User Group Japan' website. The navigation bar includes links for 'ホーム', '概要', 'ブログ', 'ドキュメント' (selected), 'フォーラム', 'フロー', and 'github'. The breadcrumb trail is 'ドキュメント > getting started > windows'. The main content area is titled 'Windowsで実行する'. It includes a 'クイックスタート' sidebar with links to 'Node.jsのインストール', 'Node-REDのインストール', and 'Node-REDの実行'. The main text explains that the page provides specific instructions for installing Node-RED on Microsoft Windows. It notes that Windows 7 and Windows Server 2008R2 are not supported. A yellow box contains a note about command prompts and PowerShell. The 'クイックスタート' section lists '1. Node.jsのインストール', followed by instructions to download Node.js from the official website and run the MSI file as an administrator.

Node-RED User Group Japan

ホーム 概要 ブログ **ドキュメント** フォーラム フロー github

ドキュメント > getting started > windows

Windowsで実行する

このページでは、Microsoft Windows環境におけるNode-REDのセットアップについて具体的な手順を紹介しています。この手順はWindows 10固有です。Windows 7およびWindows Server 2008R2以降でも利用できますが、現在サポートされていないことから推奨しません。

Note: 以下の説明の中では、「コマンドプロンプト」についての説明があります。コマンドプロンプトと書かれている場合、Windows cmdまたはPowerShellターミナルシェルを指します。PowerShellを使うことで、Windowsのすべての新しいバージョンでLinux/Macに似たコマンドやフォルダ名を利用できるため、PowerShellの利用を推奨します。

クイックスタート

1. Node.jsのインストール

Node.jsの最新版LTSをNode.js公式ホームページからダウンロードします。このサイトはあなたのシステムに最適なバージョンを提供します。

ダウンロードしたMSIファイルを実行します。Node.jsのインストールにはローカル管理者権限が必要です。つまり、ローカル管理者ではない場合、インストール時に管理者パスワードの入力を求められます。デフォルト設定でインストールをおこなってください。インストールが完了したら、開いているコマンドプロンプトを一旦閉じ、再度開いて新しい環境変数が登録されていることを確認してください。

インストール後、Node.jsとnpmが正しくインストールされていることを確認するため、コマンドプロンプトを開き、以下のコマンドを実行してください。

PowerShellを利用する場合: `node --version; npm --version`

Node-REDの公式ページにあるインストールの手順に従って、

Node.js (18.15.0以降)
npm (9.5.0以降)

をインストール。

その後、Node-REDをインストールしてください。

<https://nodered.jp/docs/getting-started/windows>

Node-REDの起動

Node-REDを起動します。

※ただし、なぜかPowerShellでの起動には失敗します。

コマンドプロンプトでの実行は成功するのでコマンドプロンプトで起動します。

```
node-red
Microsoft Windows [Version 10.0.19044.2965]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Y0000106216>node-red
1 Aug 10:34:20 - [info]

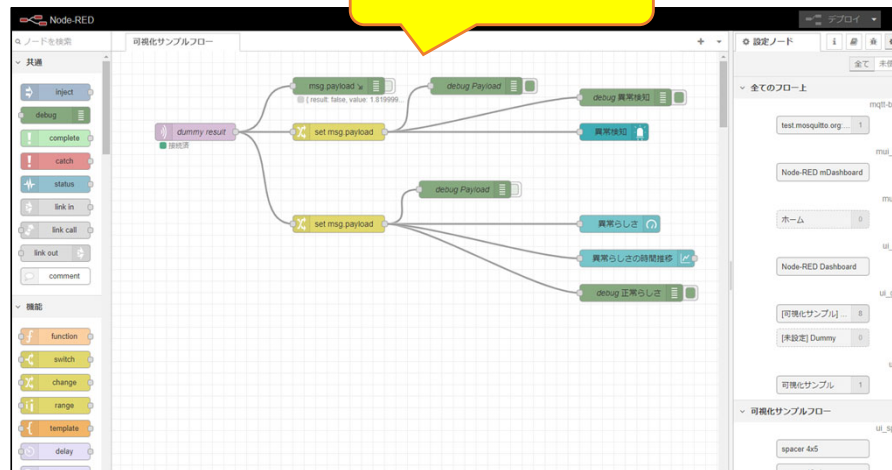
Welcome to Node-RED
=====
1 Aug 10:34:20 - [info] Node-RED version: v3.0.2
1 Aug 10:34:20 - [info] Node.js version: v20.3.0
1 Aug 10:34:20 - [info] Windows_NT 10.0.19044 x64 LE
1 Aug 10:34:22 - [info] Loading palette nodes
1 Aug 10:34:27 - [info] mDashboard version 2.19.4-beta started at /mui
1 Aug 10:34:27 - [info] Dashboard version 3.5.0 started at /ui
1 Aug 10:34:28 - [info] Settings file : C:\Users\Y0000106216\AppData\Local\node-red\settings.js
1 Aug 10:34:28 - [info] Context store : 'default' [module=memory]
1 Aug 10:34:28 - [info] User directory : Y0000106216\AppData\Local\node-red
1 Aug 10:34:28 - [warn] Projects disabled : editorTheme.projects.enabled=false
1 Aug 10:34:28 - [info] Flows file : Y0000106216\AppData\Local\node-red\flows.json
1 Aug 10:34:28 - [info] Server now running at http://127.0.0.1:1880/
1 Aug 10:34:28 - [warn]

Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.

1 Aug 10:34:28 - [info] Starting flows
```

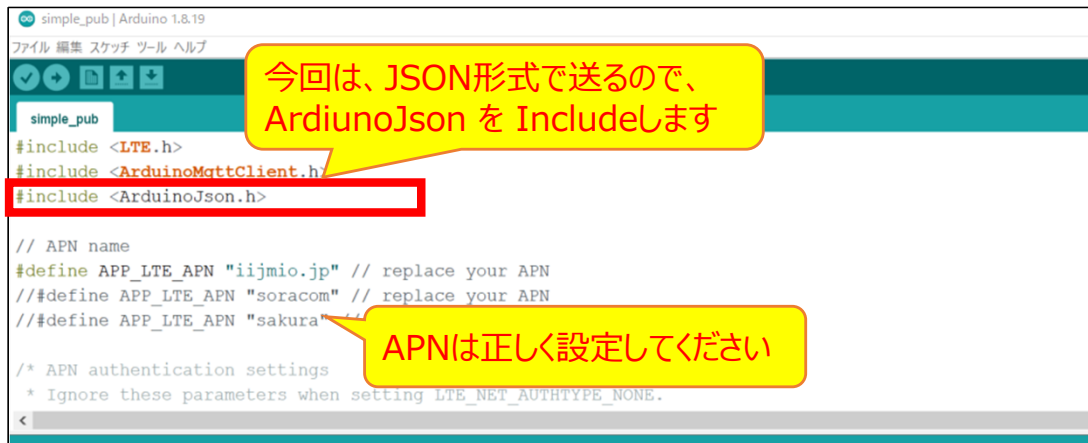


起動すると、

<http://localhost:1880>

でNode-REDが立ち上がります！

LTE経由でデータを送信するサンプルを作成してみよう！



```
simple_pub | Arduino 1.8.19
ファイル 編集 スケッチ ツール ヘルプ

simple_pub
#include <LTE.h>
#include <ArduinoMqttClient.h>
#include <ArduinoJson.h>

// APN name
#define APP_LTE_APN "iijmio.jp" // replace your APN
// #define APP_LTE_APN "soracom" // replace your APN
// #define APP_LTE_APN "sakura"

/* APN authentication settings
 * Ignore these parameters when setting LTE_NET_AUTHTYPE_NONE.
 */
```

ArduinoJsonは、Spresense向けのパッチが必要なので、こちらからダウンロードしてください。

<https://github.com/YoshinoTaro/ArduinoJson>

LTEのハンズオンで使った
[simple_pub.ino](#)
をベースに変更していきます。

基本的な使い方は、LTEの時と同じです。

ここに作ったものを置きました。

https://github.com/TomonobuHayakawa/Spresense-Playground/blob/master/ForHandsOn/DataVisualization/viaLTE/dummy_publish_for_local_visualization/dummy_publish_for_local_visualization.ino

LTE経由でデータを送信するサンプルを作成してみよう！

The screenshot shows the Arduino IDE interface with the 'simple_pub' sketch loaded. The code is as follows:

```
void loop()
{
  unsigned long currentTime = lteAccess.get
  if (currentTime >= lastPubSec + PUBLISH_I

  DynamicJsonDocument doc(1024);
  String output;
  static bool dmy_resule = true;
  static float dmy_value = 0.25;

  dmy_resule = dmy_resule ? false : true;
  dmy_value = dmy_value + 0.11;
  dmy_value = (dmy_value > 2.5) ? (dmy_value - 2.5) : dmy_value;

  doc["result"].set(dmy_resule);
  doc["value"].set(dmy_value);

  serializeJson(doc, output);

  // Publish to broker
  Serial.print("Sending message to topic: ");
  Serial.println(topic);
}
```

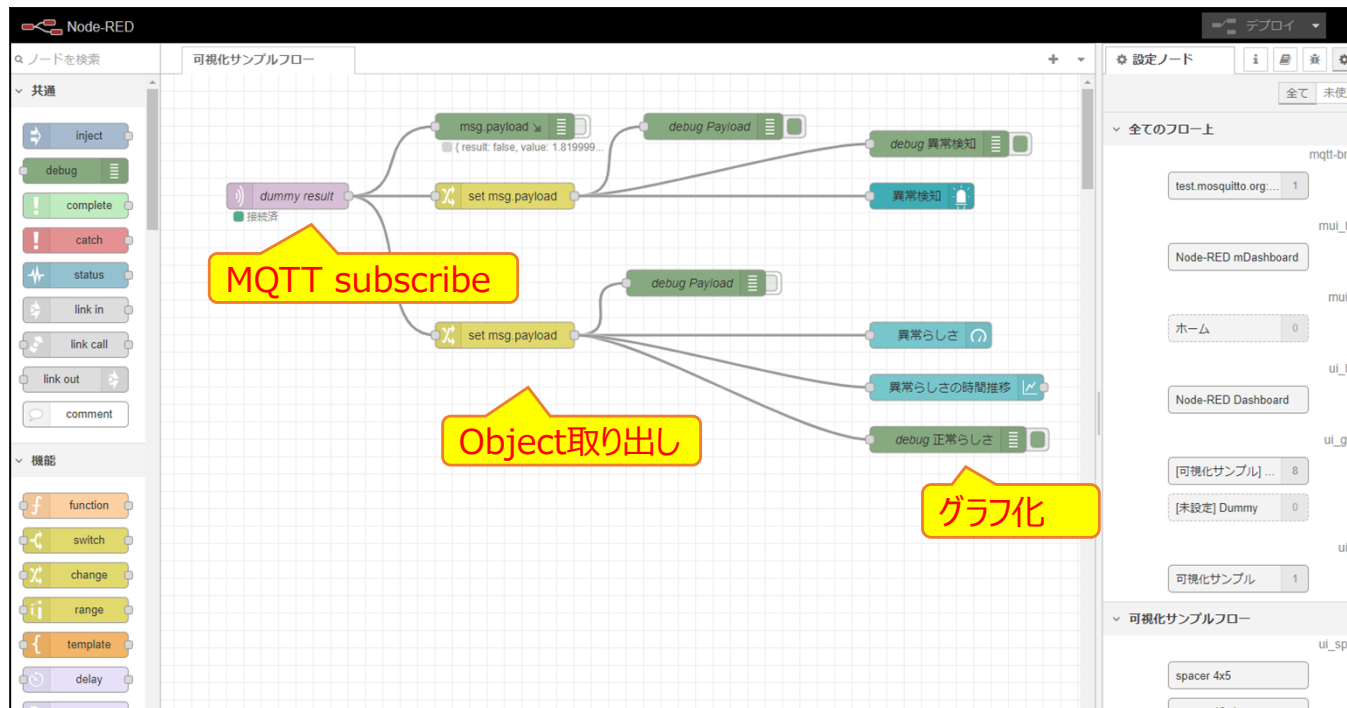
Annotations (yellow callouts) point to specific parts of the code:

- JSONインスタンスの定義** (JSON instance definition) points to `DynamicJsonDocument doc(1024);`
- シリアライズ後の文字列定義** (String definition after serialization) points to `String output;`
- ダミーデータ生成** (Dummy data generation) points to the logic for updating `dmy_resule` and `dmy_value`.
- インスタンスに格納** (Store in instance) points to `doc["result"].set(dmy_resule);` and `doc["value"].set(dmy_value);`
- シリアライズ** (Serialize) points to `serializeJson(doc, output);`
- メッセージ送信** (Message sending) points to the final `Serial.println(output);` line in the bottom section of the code.

LTEのハンズオンで使った
[simple_pub.ino](https://github.com/Sony-Semiconductor-Solutions/simple_pub.ino)
をベースに変更していきます。

基本的な使い方は、LTEの時と同じです。

Node-RED flow

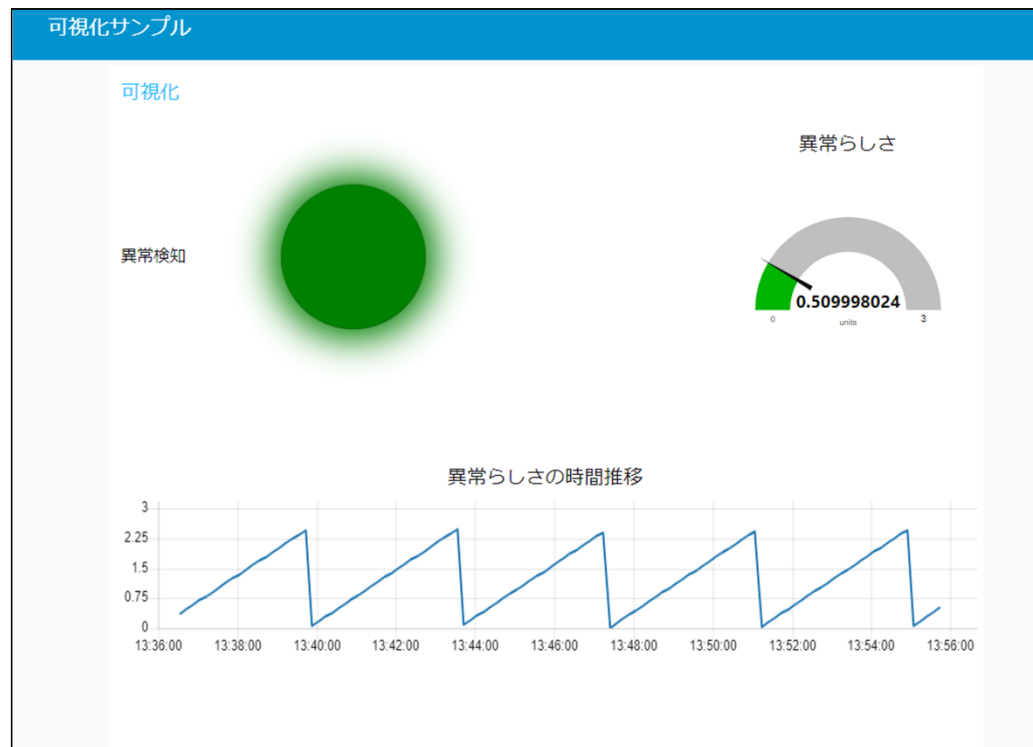


今回はJSON
そのものを送って
いるので簡単です。

ここに作ったものを置きました。

https://github.com/TomonobuHayakawa/Spresense-Playground/blob/master/ForHandsOn/DataVisualization/viaLTE/Node-RED/dummy_publish_for_local_visualization.json

可視化結果



<http://localhost/ui:1880>

で可視化結果が表示されます！

ここで上げたものは様々な**可視化ツール**の一部です。

ご自分で最適なツールを見つけて、
是非、**Spresense**を**IoT**の**POC**に活用してみてください！