

Wi-Fi 接続のご案内

SSID : Buffalo-G-AE22

PW : 12345678

本日の資料

https://github.com/IoT-ALGYAN-Hiroshima/IoT_Handson_20181223

STM開発ボードIoTプロトタイプ製作発表会meetup !

～ #2【ボード編】@ひろしまCamps ～

時間	内容
12:30-12:55	受付
13:00-13:05	オープニング
13:05-16:40	ハンズオン本編
16:40-17:30	懇親会(無料)

主催 : IoT ALGYAN(あるじyan)

協賛 : STマイクロエレクトロニクス

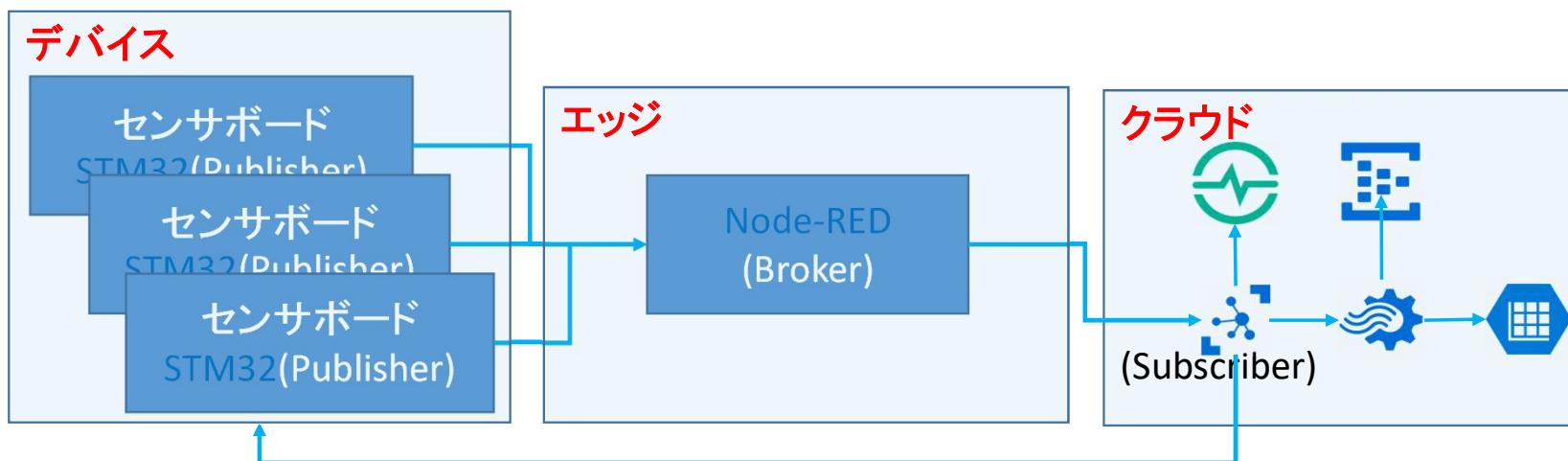
協力 : ST日本マイクロソフト株式会社, 広島県

<https://algyan.connpass.com/event/105130/>

<https://www.pref.hiroshima.lg.jp/site/innovation/algyan.html>

ハンズオンでやりたいこと

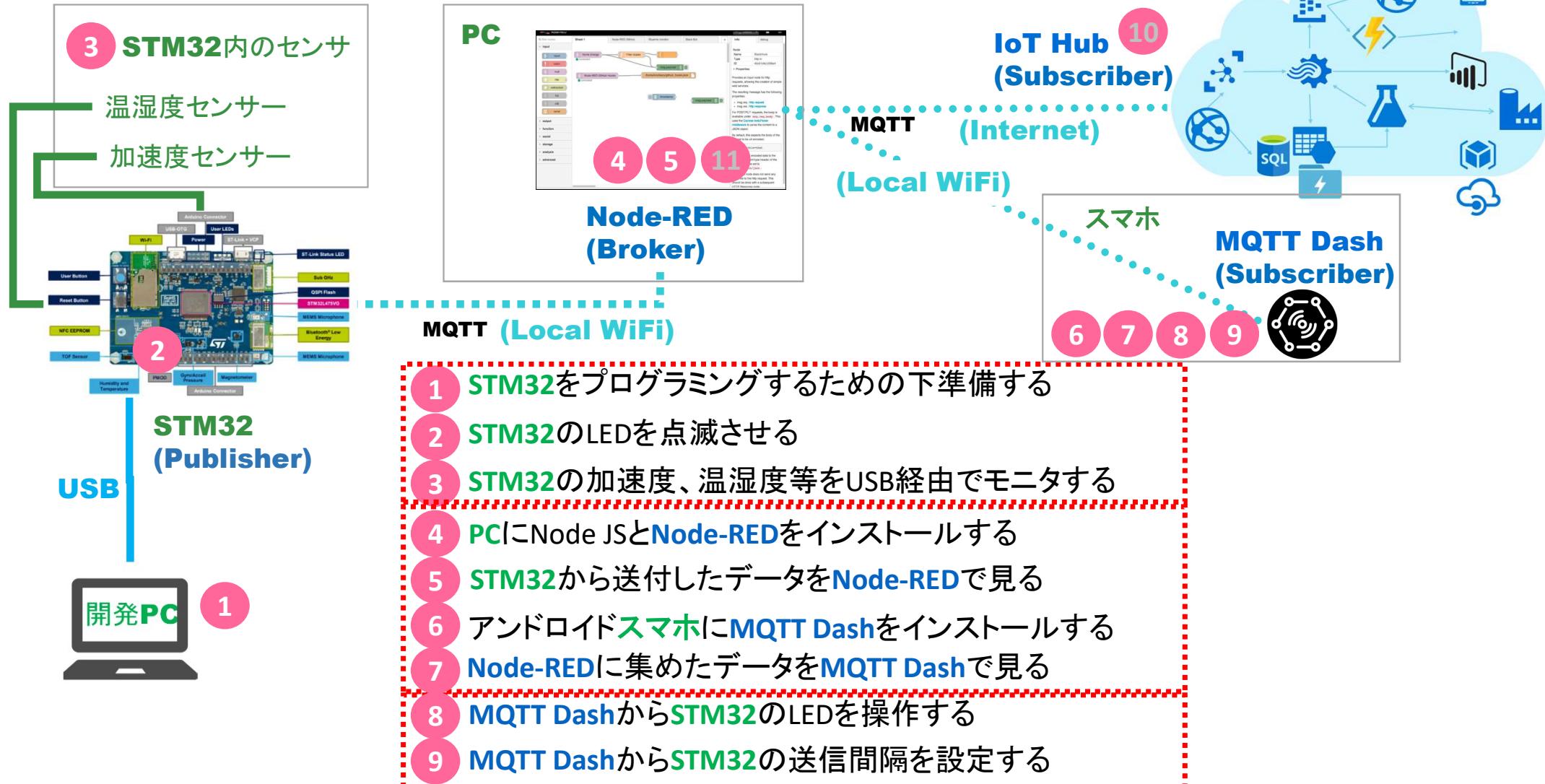
- ▶ 無数の管理された**デバイス**情報を
→ 無数のセンサーボードを一括設定する
(**STM32** の発信間隔をクラウドから一括設定する)
- ▶ 一時的に**エッジ**に集めて加工し
→ 集めたセンサ情報をクラウドに送る
(**Node-RED**を使って集めたセンサ情報を加工し送る)
- ▶ **クラウド**でデータを解析する
→ クラウド上に貯まったデータを見える化する
(**IoT Hub** と **Time Series Insights**等でグラフ化する)



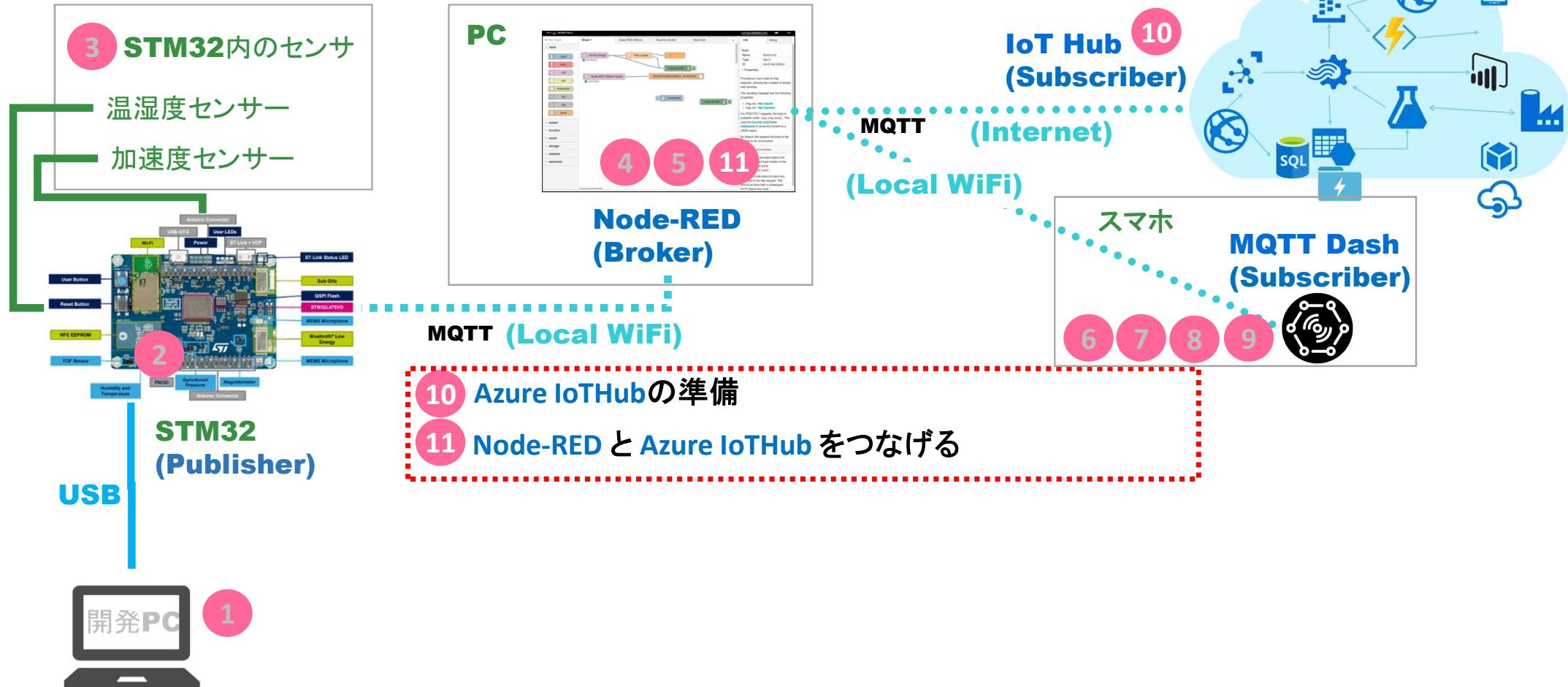
凡例：Microsoft Azureのサービス

- IoT Hub(DeviceExploreより設定)
- Time Series Insight
- Stream Analytics
- Event Hubs
- Storage

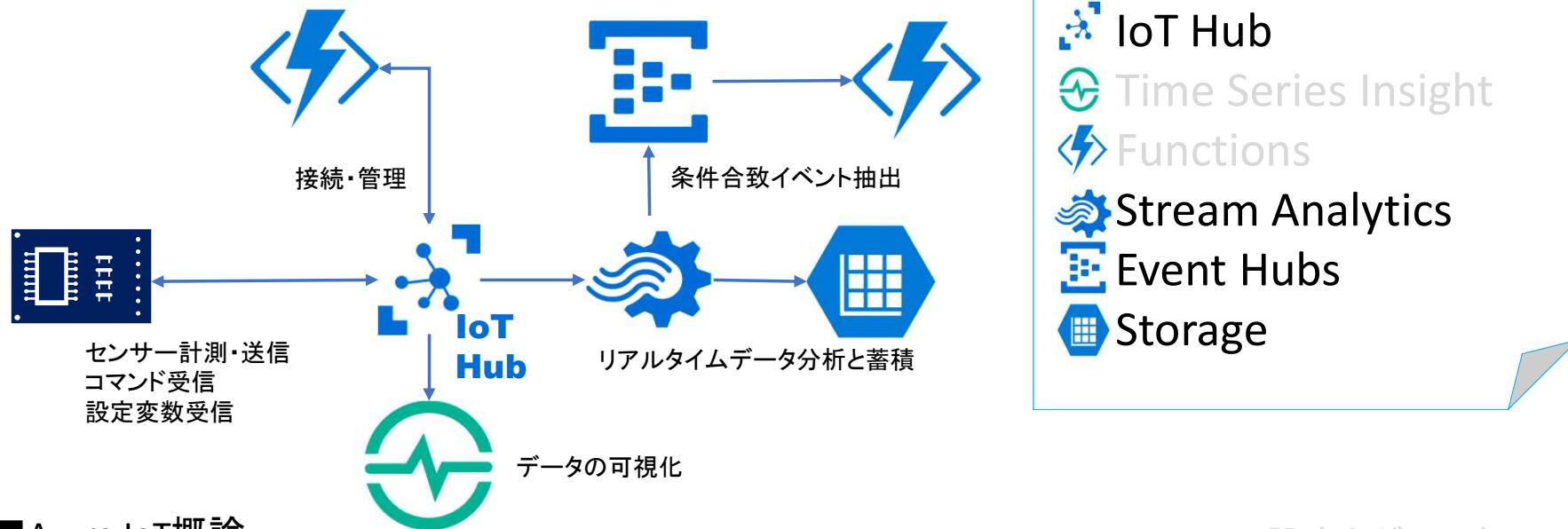
センサー・ボード・エッジ編 [1] (12/23)



センサー・ボード・エッジ編 [2] (12/23)



クラウド編(12/02振り返り)



- Azure IoT概論
- Azureの使い方概論
 ポータル、ダッシュボード、リソースグループ
- Azureハンズオン本編
 - IoTデバイスの登録
 - Node-RED Connectの設定
 - Storage Account作成
 - Stream Analyticsの作成
 - Stream Analytics実行 & ハードウェア送信開始
 - ブロブにデータが入ったか確認

- Time Series Insightsの設定とグラフ表示
(ここで実際にSTM32と接続)
- Option データの可視化(測定データの記録)
- BLOBストレージの作成
- Stream Analyticsの作成
- デバイス管理パラメタ設定
- Stream Analytics止めて
Event Hub作成とStream Analyticsクエリ/出力更新
- Time Series Insightsの作成・設定
- Stream Analytics再実行 & TSIでグラフ表示

1 センサーをボードをプログラミングするための下準備

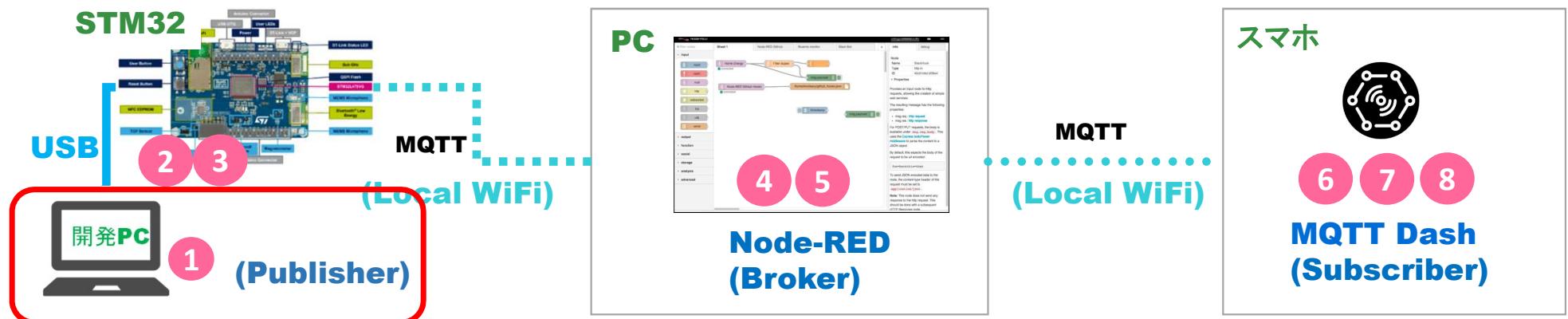
1-1 開発PCにArduino IDEをインストールします

Arduino IDE とは標準のArduino開発環境

1-2 STM32にstm32duino/Arduino_Core_STM32をインストールします

stm32duino/Arduino_Core_STM32 とは

STM32をArduino IDE でプログラミングするための基本的なライブラリ群



1-1 開発PCにArduino IDEをインストールする

以下のサイトからダウンロードしてインストールします
又は別途配布の”arduino-1.8.8-windows.exe”をデスクトップ等にコピーし
インストールします

■Arduino IDEのダウンロード先

<https://www.arduino.cc/en/Main/Software>

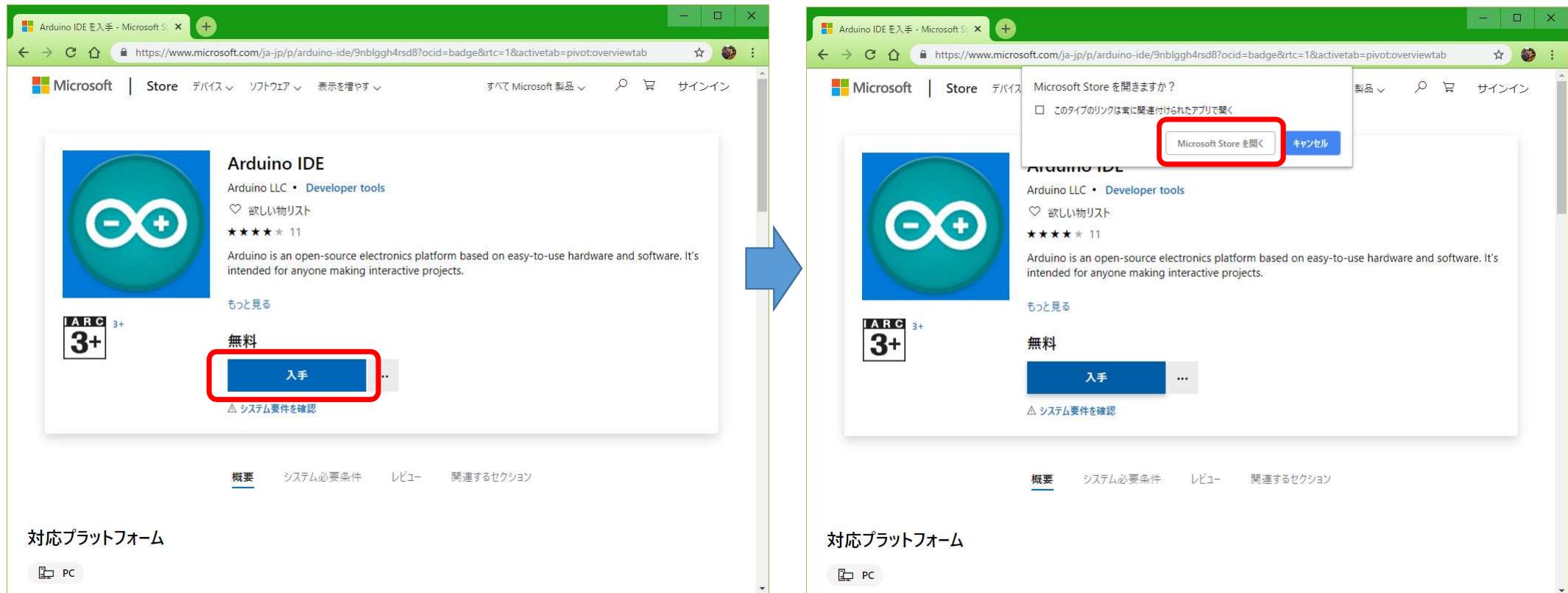
Arduino IDEをダウンロードしインストールする(1)

The screenshot shows the Arduino Software download page. At the top, there's a navigation bar with links for HOME, STORE, SOFTWARE, EDU, RESOURCES, COMMUNITY, and HELP. Below the navigation bar, the main content area is titled "Download the Arduino IDE". It features a large circular logo with the Arduino infinity symbol. To the right of the logo, there's a brief description of the software: "The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the Getting Started page for installation instructions." Below this text, there are download links for different operating systems: "Windows Installer, for Windows XP and up", "Windows ZIP file for non admin install", "Windows app Requires Win 8.1 or 10" (which is highlighted with a red box), "Mac OS X 10.8 Mountain Lion or newer", "Linux 32 bits", "Linux 64 bits", and "Linux ARM". At the bottom of the main content area, there are sections for "HOURLY BUILDS" and "BETA BUILDS".

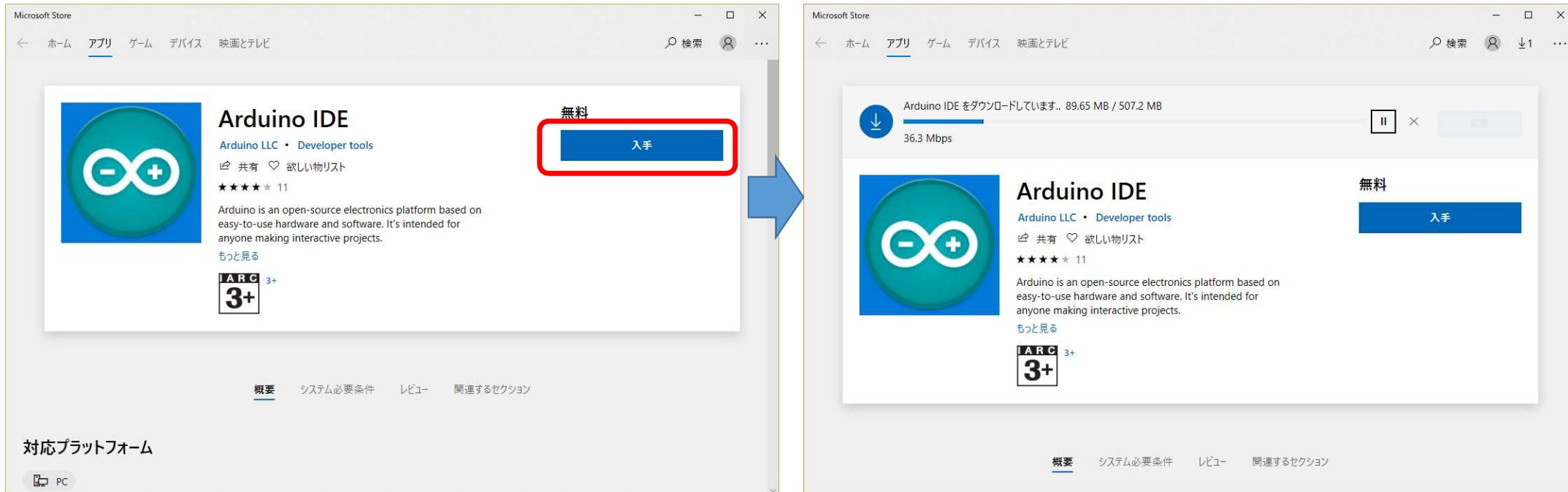


The screenshot shows the Arduino Software contribution page. The top navigation bar is identical to the download page. The main content area is titled "Contribute to the Arduino Software". It contains a message encouraging users to contribute to the development of the software: "Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). Learn more on how your contribution will be used." Below this message, there's a graphic of three stylized Arduino boards (orange, grey, and green) with legs and arms, standing together. To the right of the graphic, there's a block of text: "SINCE MARCH 2015, THE ARDUINO IDE HAS BEEN DOWNLOADED 28,652,170 TIMES. (IMPRESSIVE!) NO LONGER JUST FOR ARDUINO AND GENUINO BOARDS, HUNDREDS OF COMPANIES AROUND THE WORLD ARE USING THE IDE TO PROGRAM THEIR DEVICES, INCLUDING COMPATIBLES, CLONES, AND EVEN COUNTERFEITS. HELP ACCELERATE ITS DEVELOPMENT WITH A SMALL CONTRIBUTION! REMEMBER: OPEN SOURCE IS LOVE!" Below this text, there are several circular buttons with contribution amounts: "\$3", "\$5", "\$10", "\$25", "\$50", and "OTHER". At the bottom of the page, there are two buttons: "JUST DOWNLOAD" (which is highlighted with a red box) and "CONTRIBUTE & DOWNLOAD".

Arduino IDEをダウンロードしインストールする(2)



Arduino IDEをダウンロードしインストールする(3)



Arduino IDEをダウンロードしインストールする(4)



インストール場所は、c:¥program files¥

1-2 STM32にstm32duino/Arduino_Core_STM32をインストールする

Arduino IDE のボードマネージャーからインストールします
下記の7ステップの順に進めます

■インストール方法

<https://github.com/stm32duino/wiki/wiki/Getting-Started>

Step1 Arduino IDE のボードマネージャーのダウンロード先を設定する

Step2 Arduino IDE のボードマネージャーでstm32duino/Arduino_Core_STM32をダウンロードする

Step3 Arduino IDEで使うセンサボードをSTM32に設定する

Step4 開発PCのどのCOMポートにSTM32に繋がっているか確認する

Step5 Arduino IDEのシリアルポートをSTM32に繋がっているポートに設定する

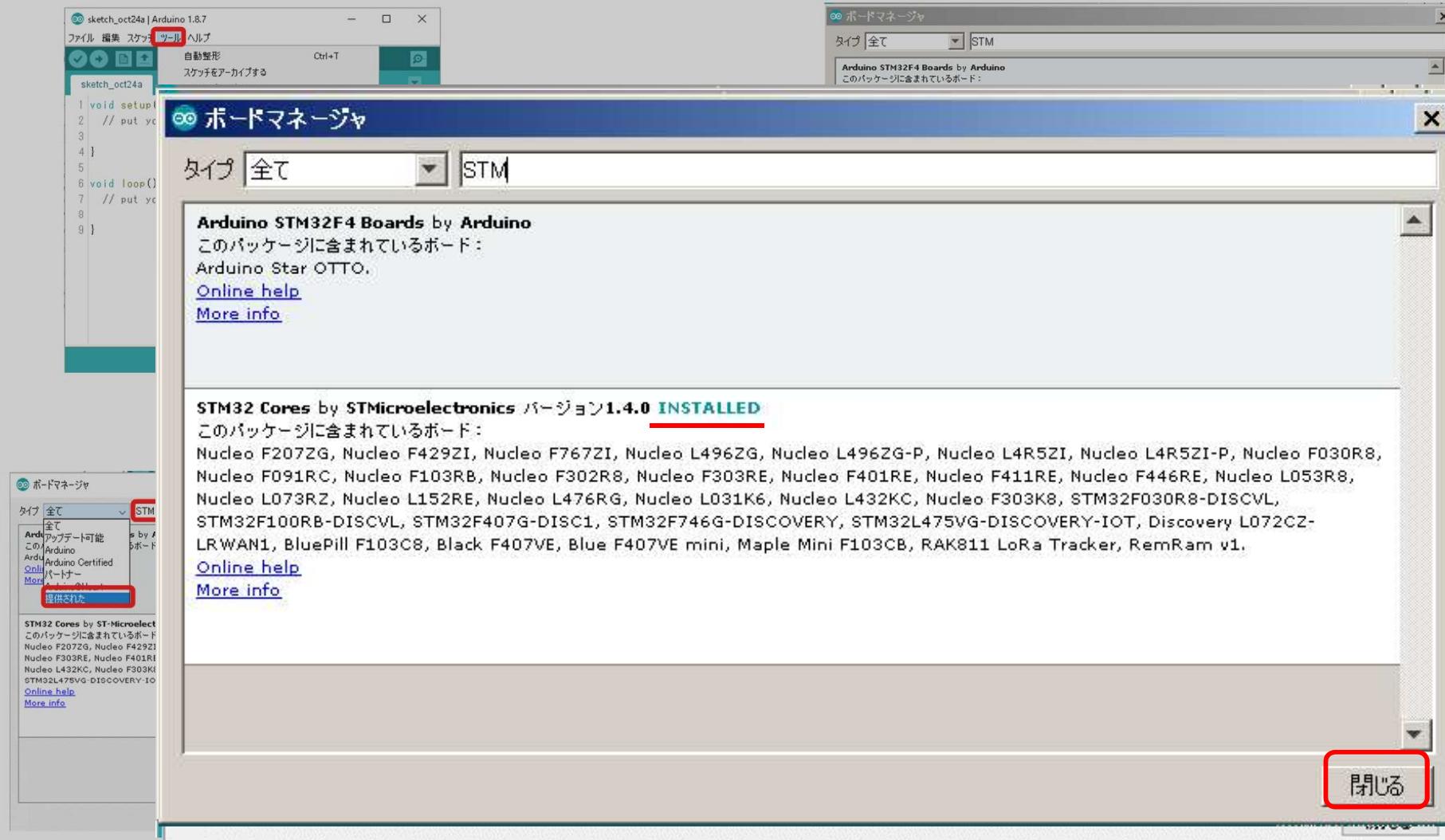
Step6 書き込み装置を ArduinoISPに設定する

Step7 STM32が正しくArduino IDEに認識されているか確認する

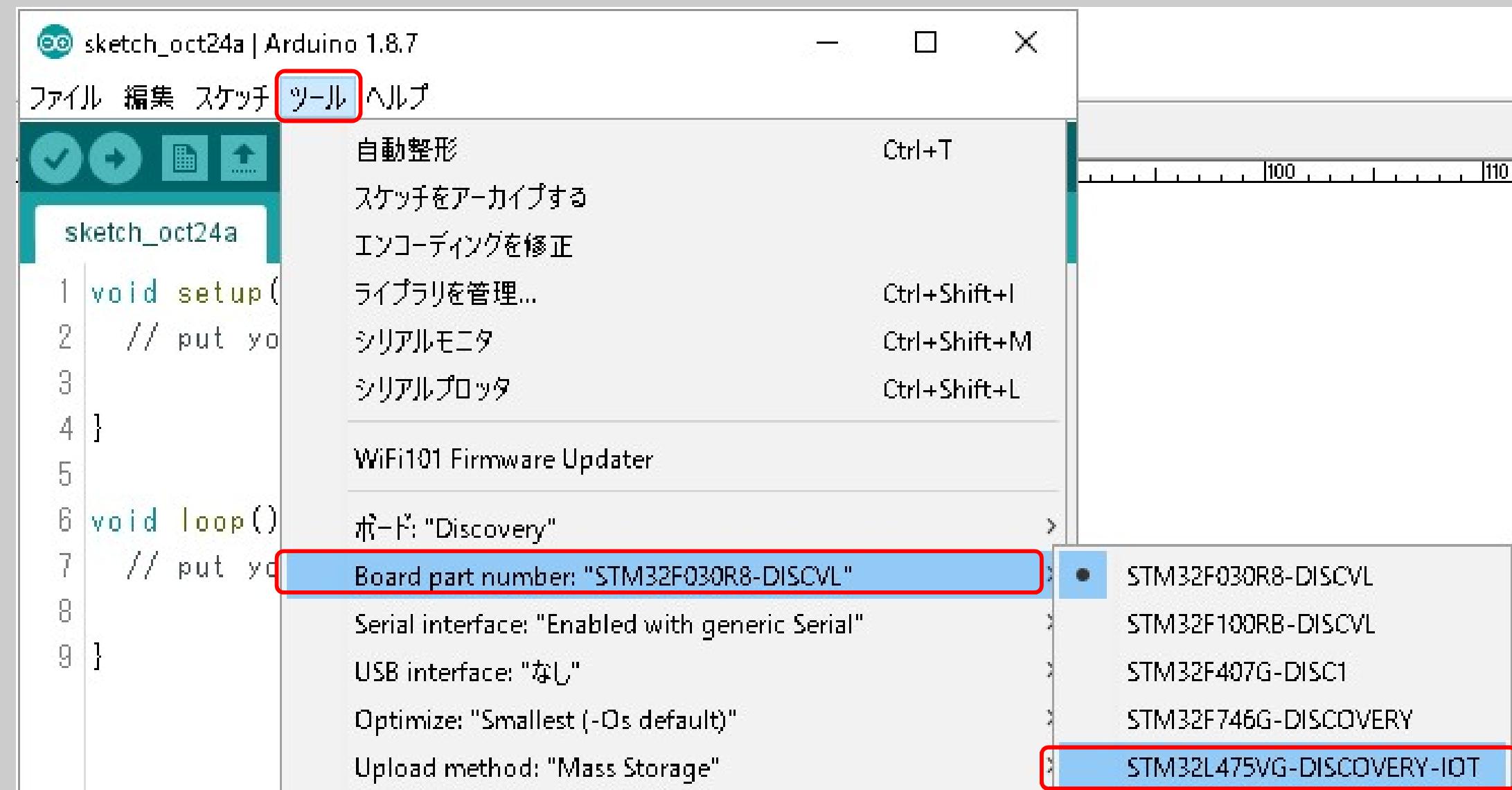
Step1 Arduino IDE のボードマネージャーのダウンロード先を設定します



Step2 Arduino IDE のボードマネージャーでstm32duino/Arduino_Core_STM32をダウンロードします



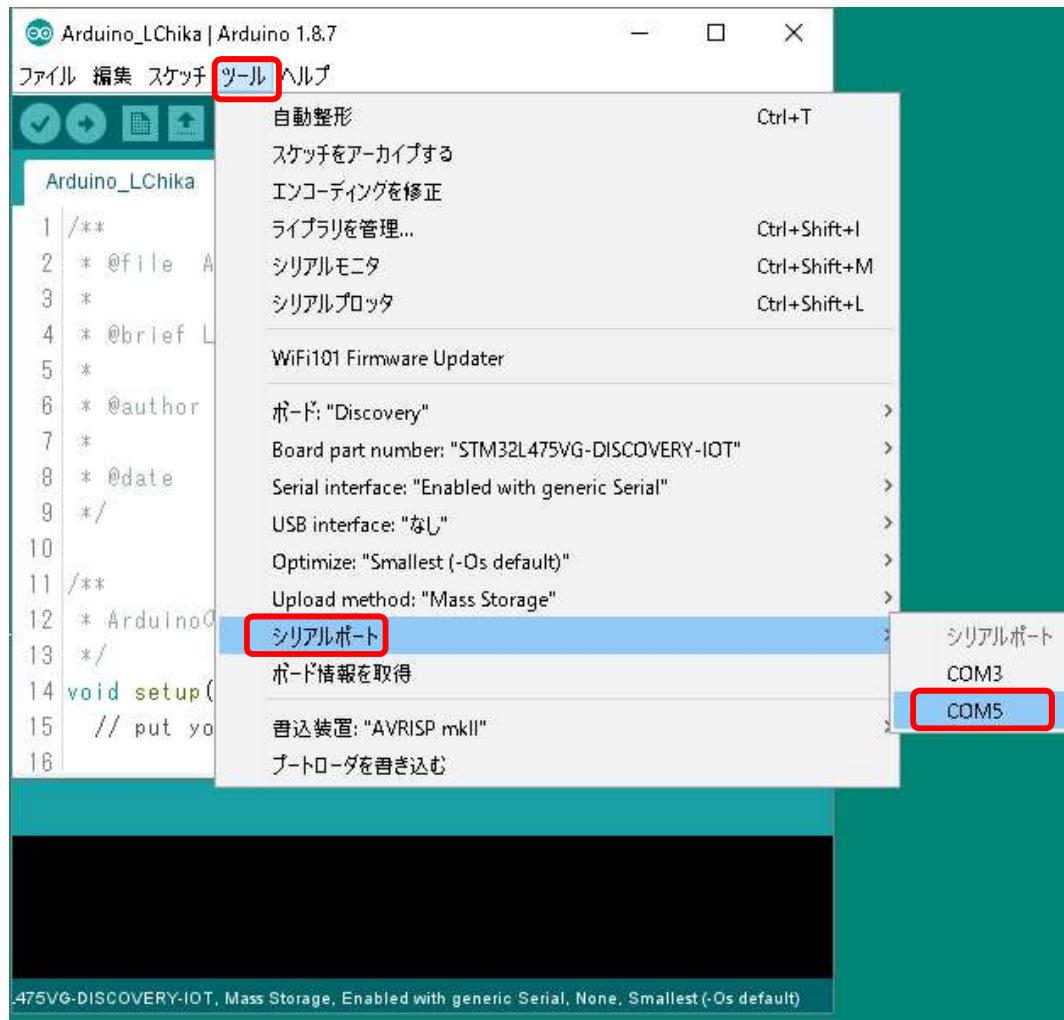
Step3 Arduino IDEで使うセンサボードをSTM32に設定します



Step4 開発PCのどのCOMポートにSTM32に繋がっているか確認します

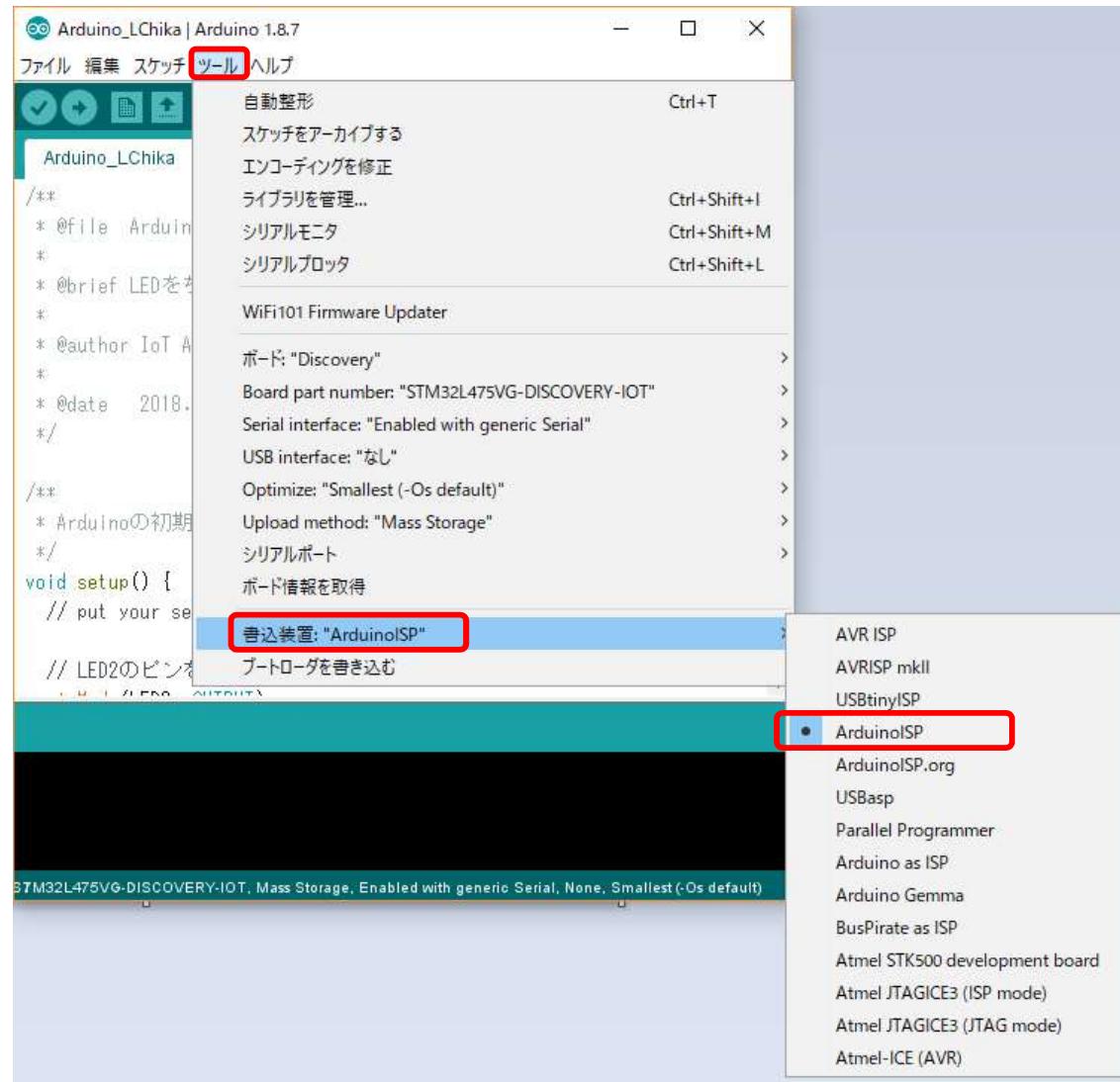


Step5 Arduino IDEのシリアルポートをSTM32に繋がっているポートに設定します

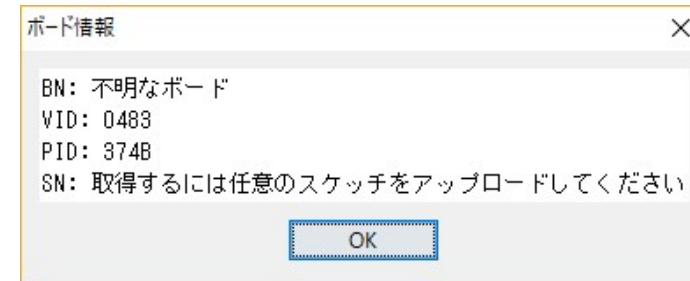
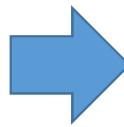
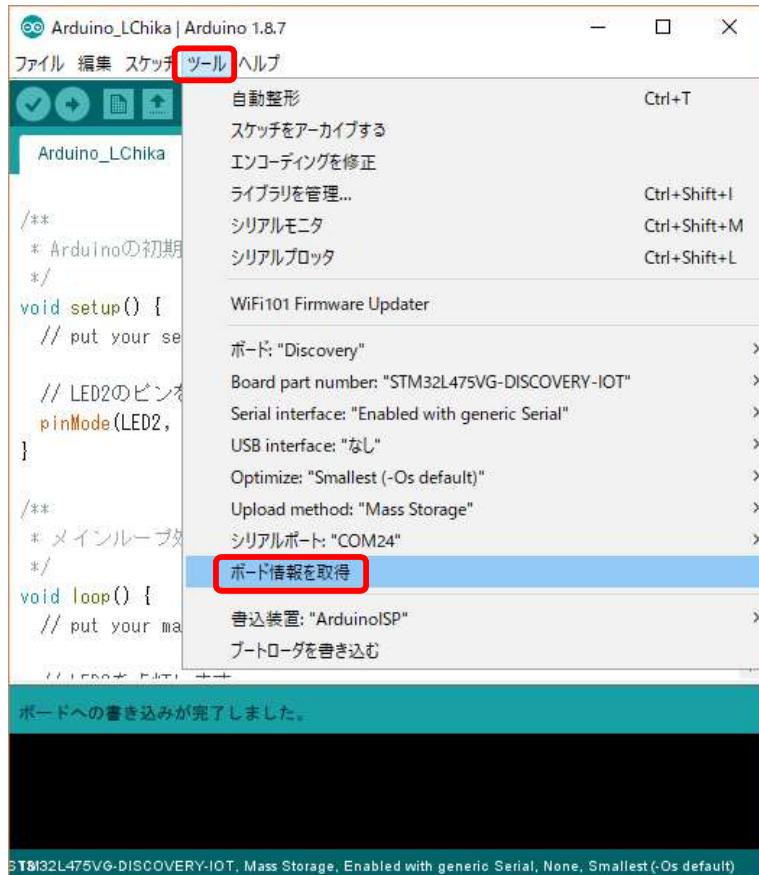


Step4で確認したSTM32に繋がっているポート
に設定します

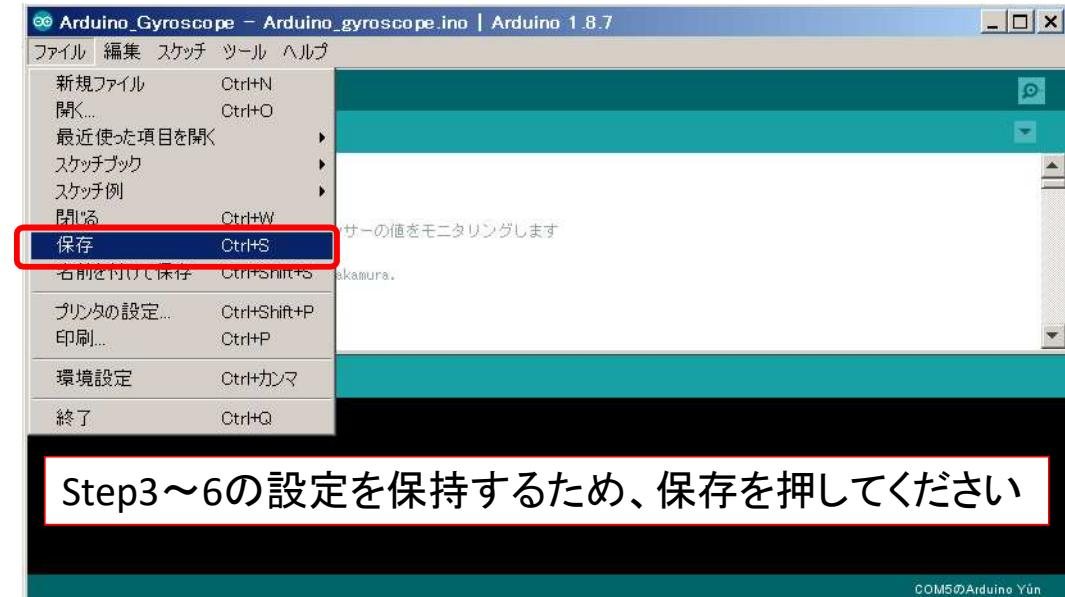
Step6 書き込み装置を ArduinoISPに設定します



Step7 STM32が正しくArduino IDEに認識されているか確認します



VID(ベンダーID)/ PID(プロダクトID)が表示されていれば
正しく認識されています

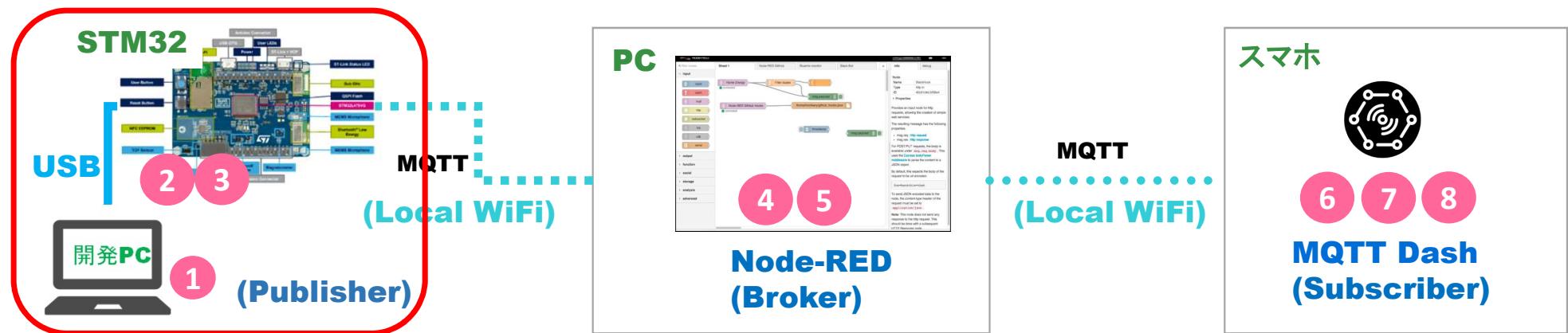


2 STM32のLEDを点滅させる

次の順で正常に開発PCからSTM32にプログラミングできるか確認します

2-1 サンプルコードをダウンロードする

2-2 STM32にサンプルコードを書き込む



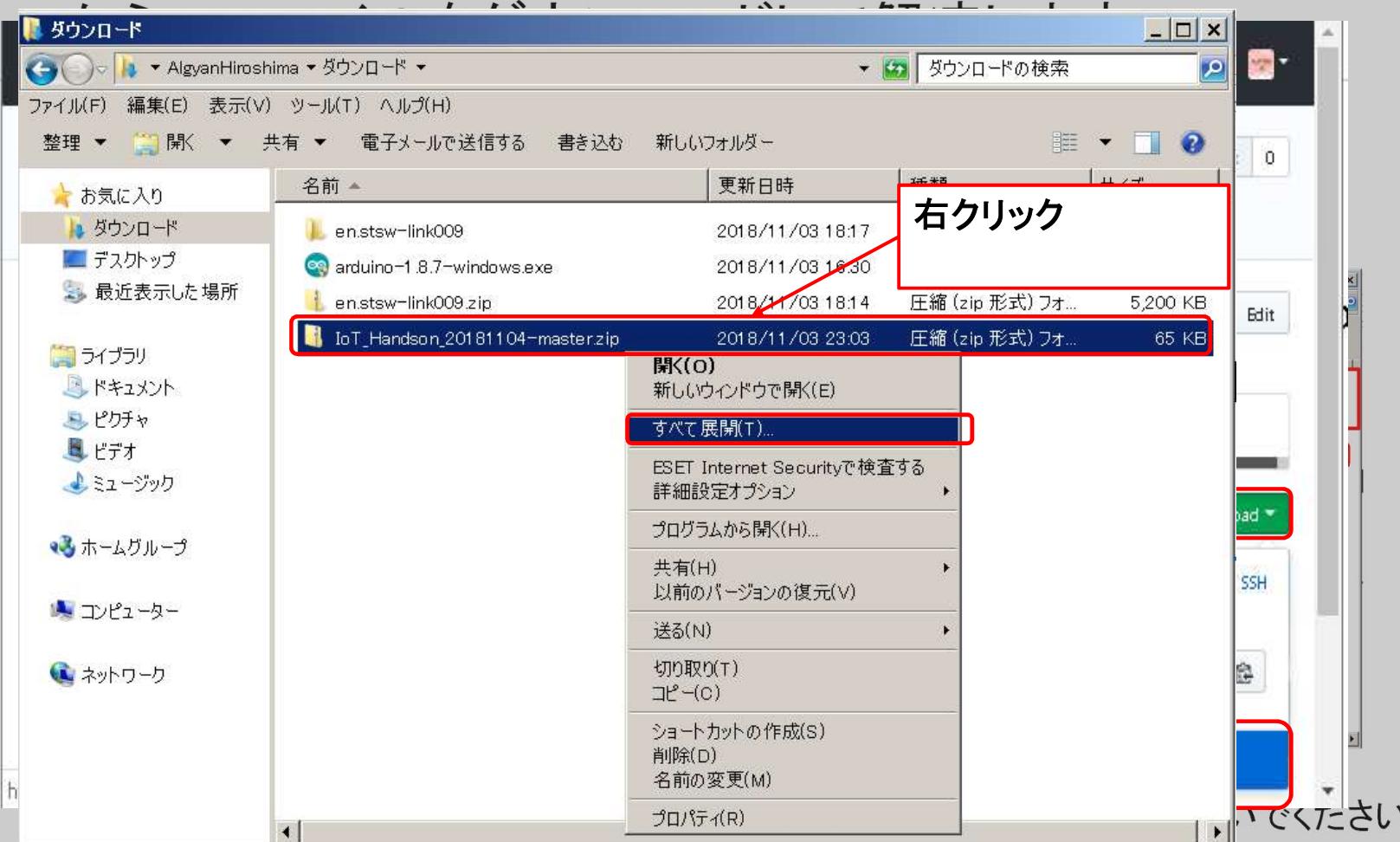
2-1 サンプルコードをダウンロードする

下記のGitHub

■ダウンロード

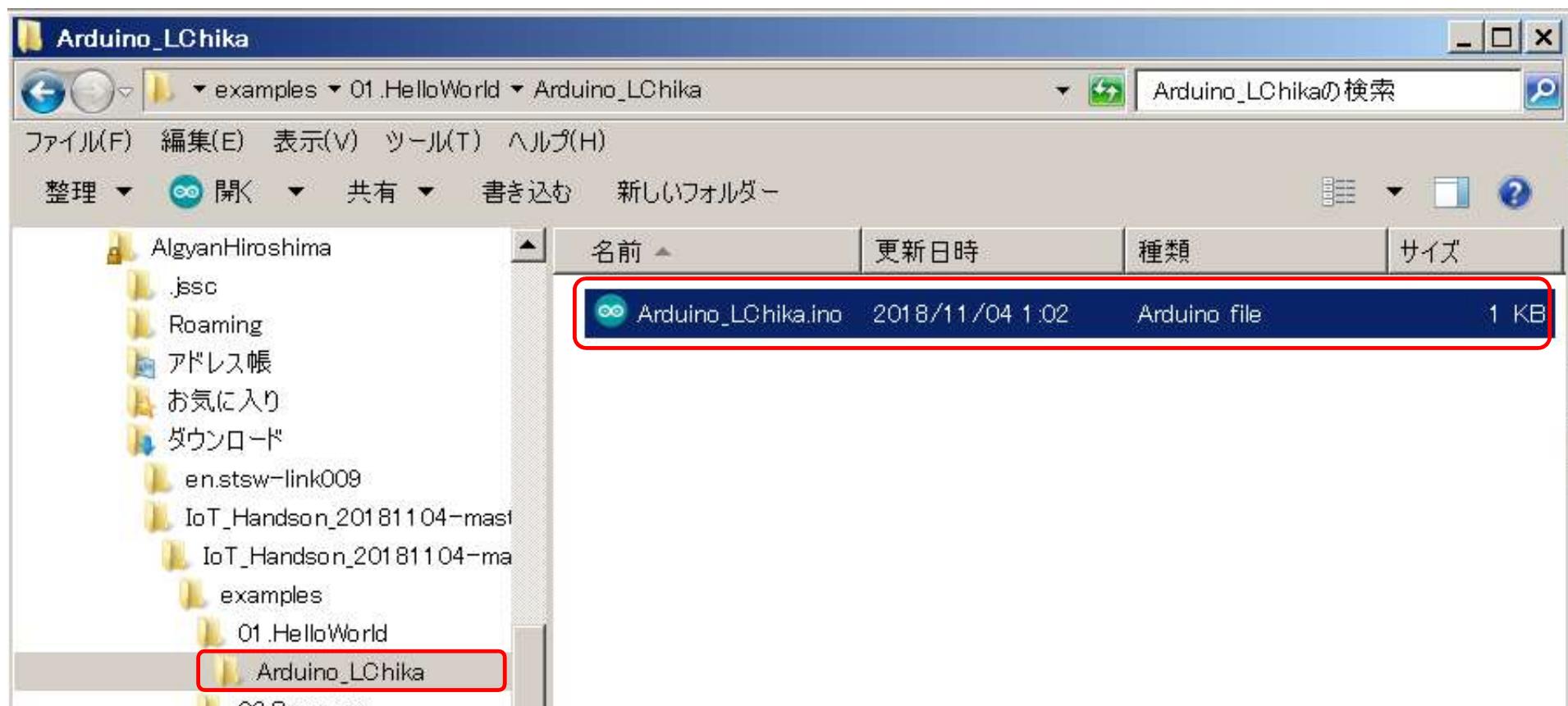
https://github.com/IoT-ALGYAN-Hiroshima/IoT_Handson

The screenshot shows the GitHub interface for the repository 'IoT_Handson'. On the left, there's a sidebar with links for 'Code', 'Issues', and 'Pull requests'. The main area displays a list of files, including 'enstsw-link009', 'arduino-1.8.7-windows.exe', 'enstsw-link009.zip', and 'IoT_Handson_20181104-master.zip'. The file 'IoT_Handson_20181104-master.zip' is highlighted with a red box.

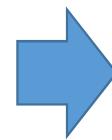


2-2 STM32にサンプルコードを書き込む

解凍してできたフォルダ内のArduino_LChika.inoをダブルクリックします
すると Arduino IDE を使ってサンプルコードが開きます



PCとSTM32とUSBケーブルでつなぎ繋ぎ
赤と緑のランプが点灯することを確認します



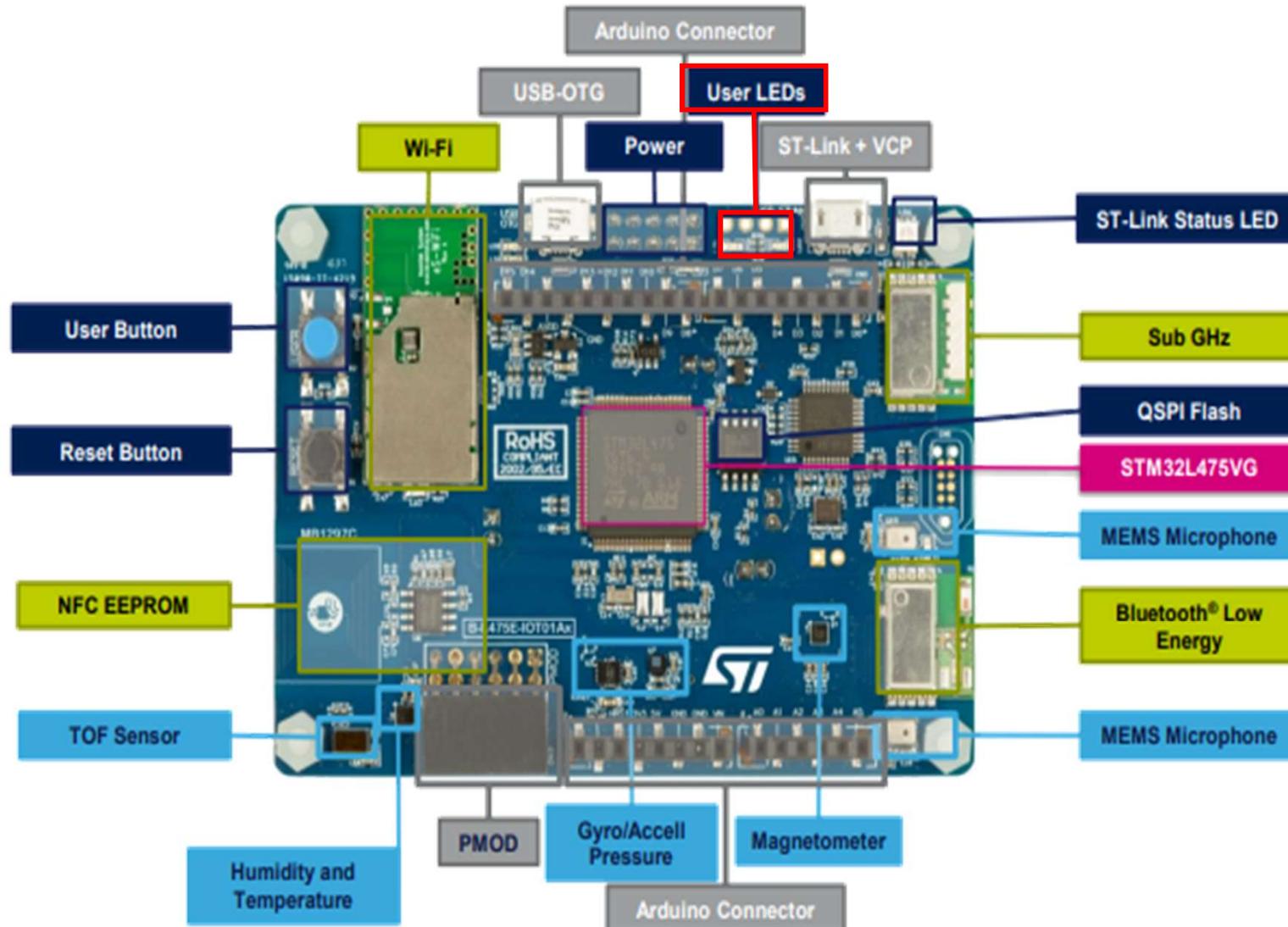
コンパイル&書き込みボタンを押して書き込みます

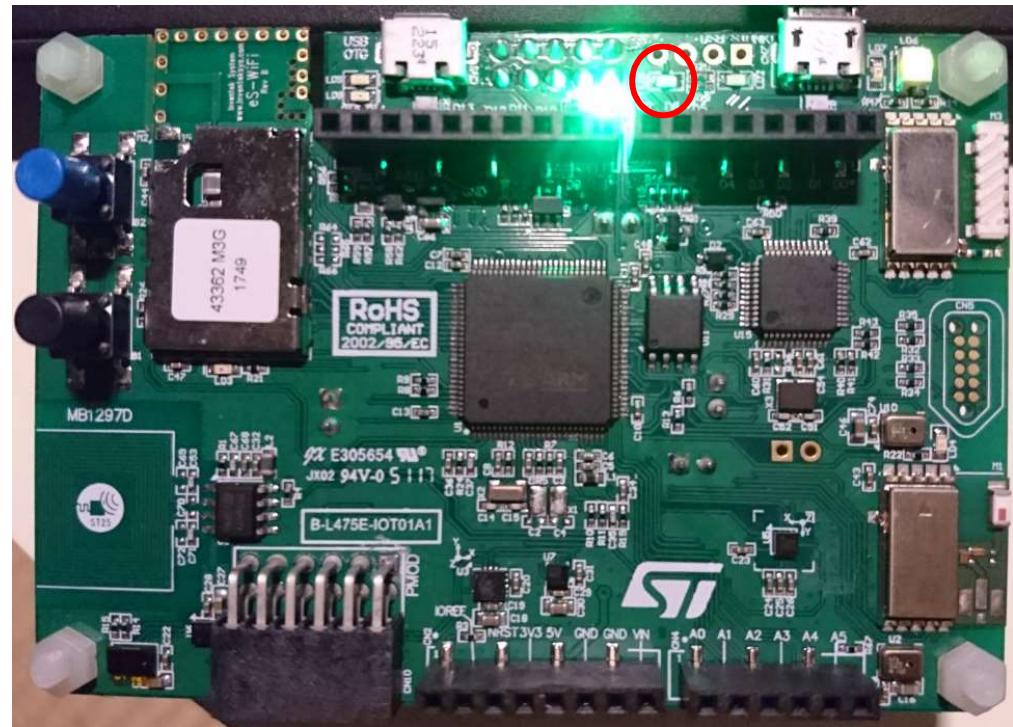
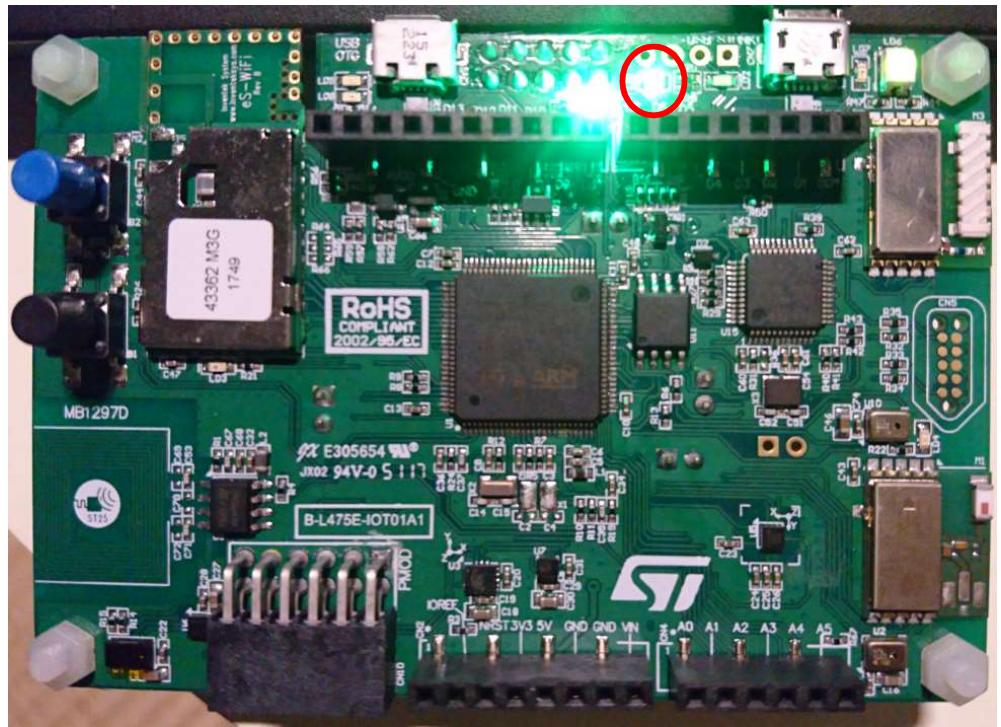
```
Arduino_LChika | Arduino 1.8.7
ファイル ファイル ページ ツール ヘルプ
マイコンボードに書き込む
Arduino_LChika
1 /**
2 * @file Arduino_LChika.ino
3 *
4 * @brief LEDをちかちかさせます
5 *
6 * @author IoT Algyan in Hiroshima. K.Nakamura.
7 *
8 * @date 2018.09.09
9 */
10
11 /**
12 * Arduinoの初期化を行います
13 */
14 void setup() {
15 // put your setup code here, to run once:
16 }
```

ボードへの書き込みが完了しました。

1 個のファイルをコピーしました
Upload complete on DIS_L4IOT (1:)

赤枠で囲った箇所のLEDが点滅していることを確認します





赤い枠で囲ったLEDが点滅すれば、動作確認は成功です

(参考) 書き込んだサンプルプログラムの内容

初期化関数

```
void setup() {  
    // put your setup code here, to run once:  
  
    pinMode(LED2, OUTPUT);  
}
```

LEDのピンの 設定

メインループ処理関数

```
void loop() {  
    // put your main code here, to run repeatedly:  
  
    digitalWrite(LED2, HIGH);  
    delay(1000);  
  
    digitalWrite(LED2, LOW);  
    delay(1000);  
}
```

LEDを点灯させる

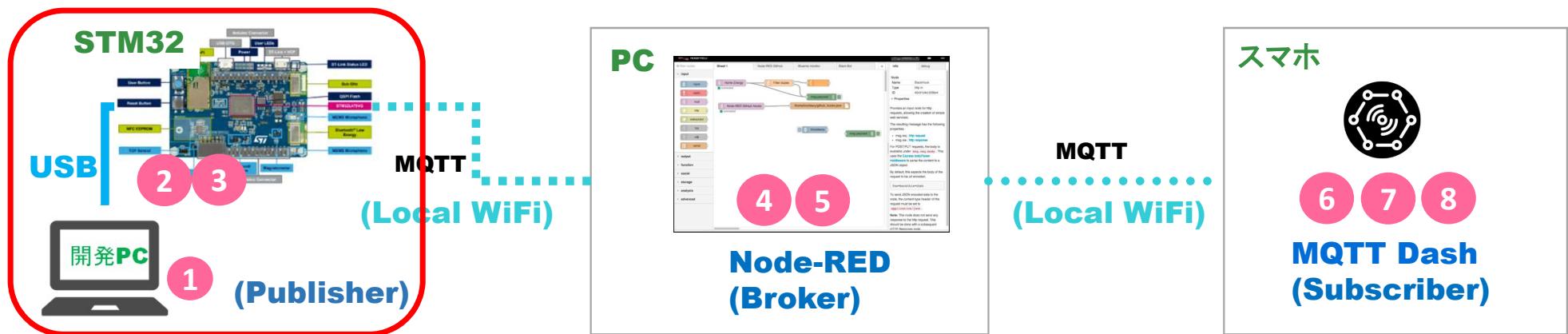
LEDを消灯させる

delay(1000)は、
1000msec =(1sec)待つ

3. STM32の加速度、温湿度等をUSB経由でモニタする

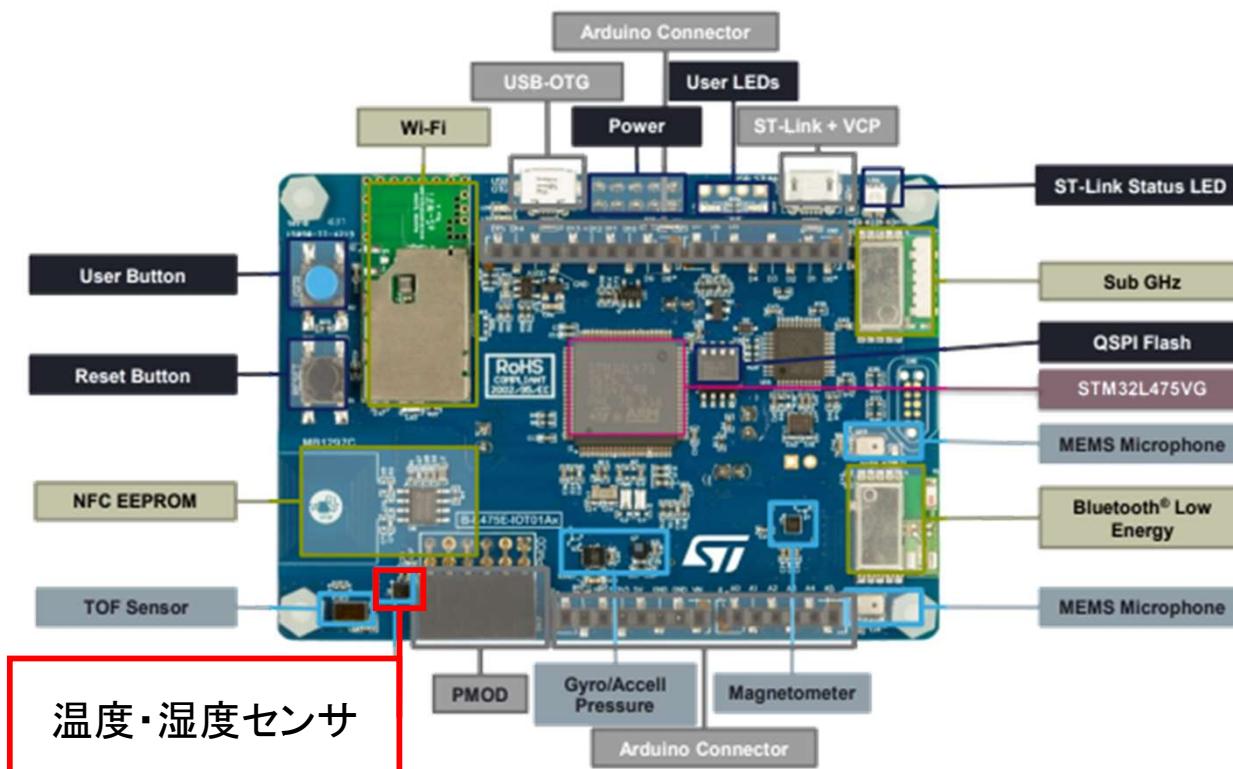
STM32に載っているセンサのライブラリをインストールしてサンプルコードを書き込みます。USBケーブル経由でセンサデータをモニタしセンサが正常に動作していることを確認します。

- 3-1 STM32の温度・湿度センサデータをUSB経由でモニタする
- 3-2 STM32の気圧計センサデータをUSB経由でモニタする
- 3-3 STM32のジェスチャセンサデータをUSB経由でモニタする
- 3-4 STM32の3軸地磁気センサデータをUSB経由でモニタする
- 3-5 STM32の3DジャイロセンサデータをUSB経由でモニタする



3-1 温度・湿度センサをUSB経由でモニタ

温度・湿度センサのライブラリをインストールし、その後STM32にサンプルコードを書き込み
センサからのデータをUSB経由でモニタします。

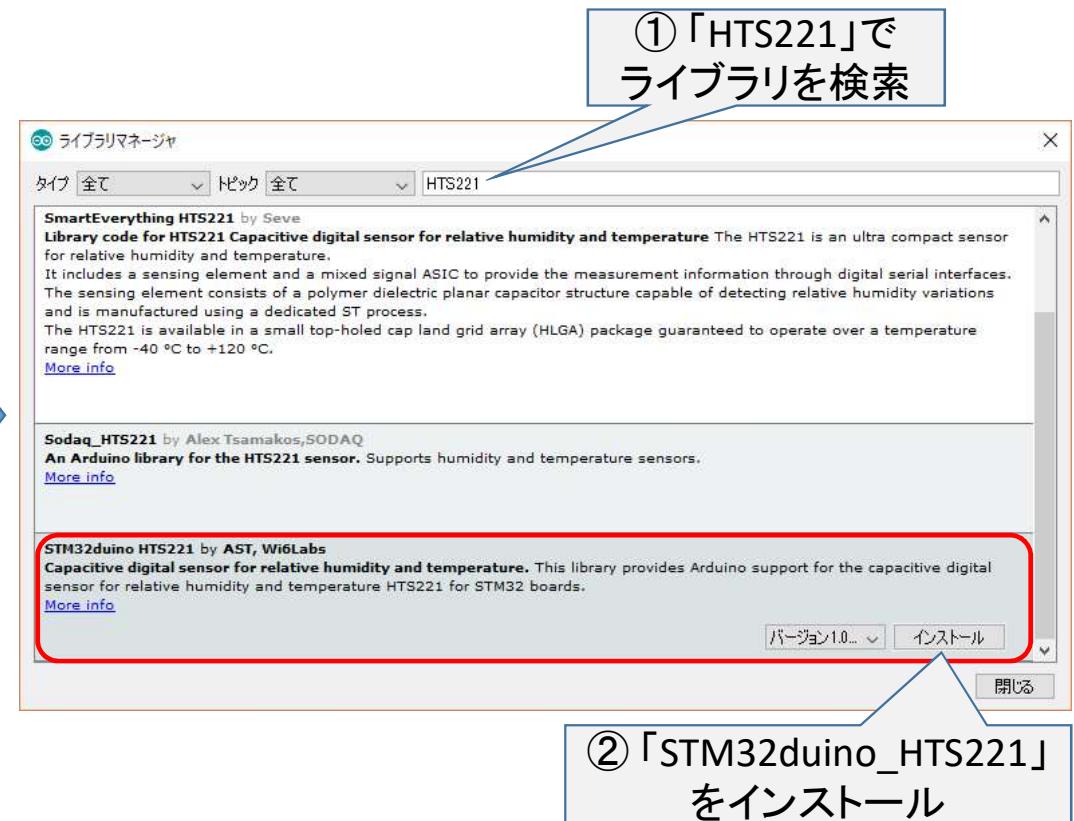
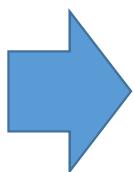


■温度・湿度センサのライブラリのインストール

ライブラリマネージャで「STM32duino-HTS221」をインストールします。

温度・湿度センサ名 : HTS221

ライブラリ名 : STMduino-HTS221

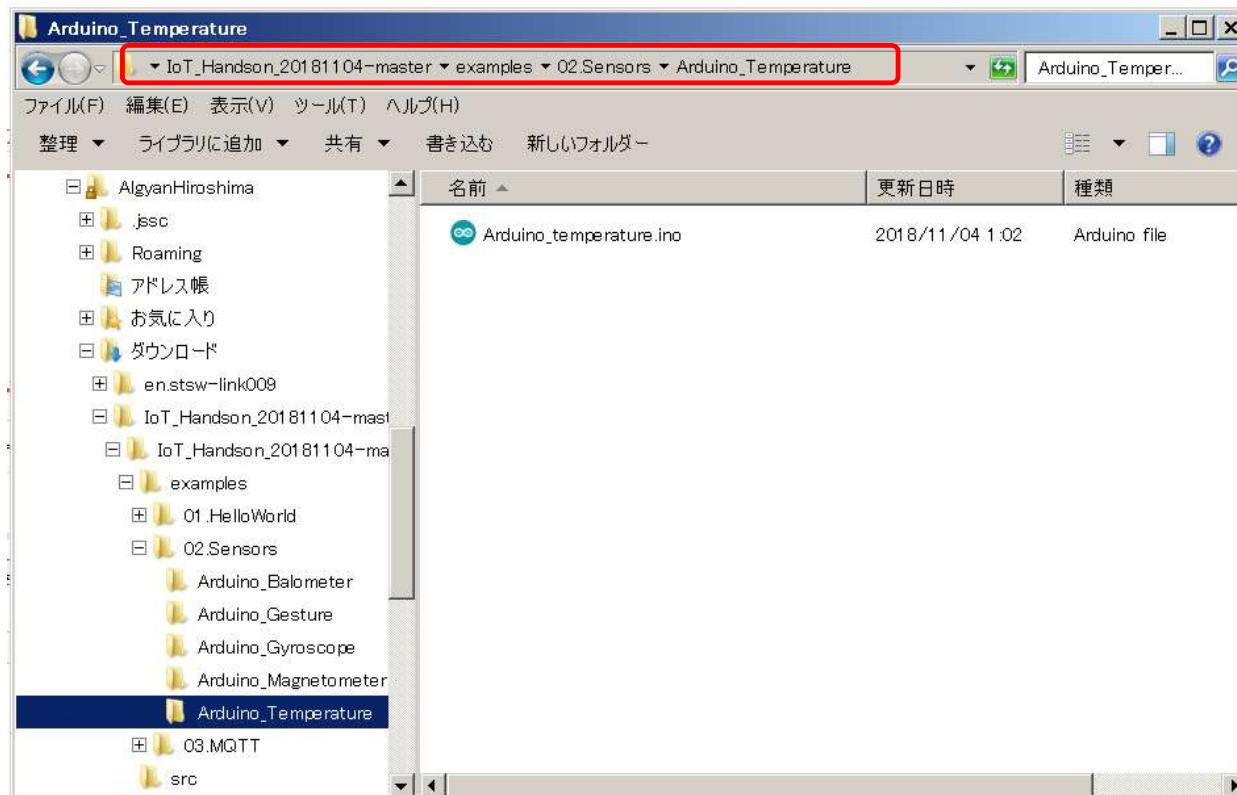


■サンプルコードを書込む

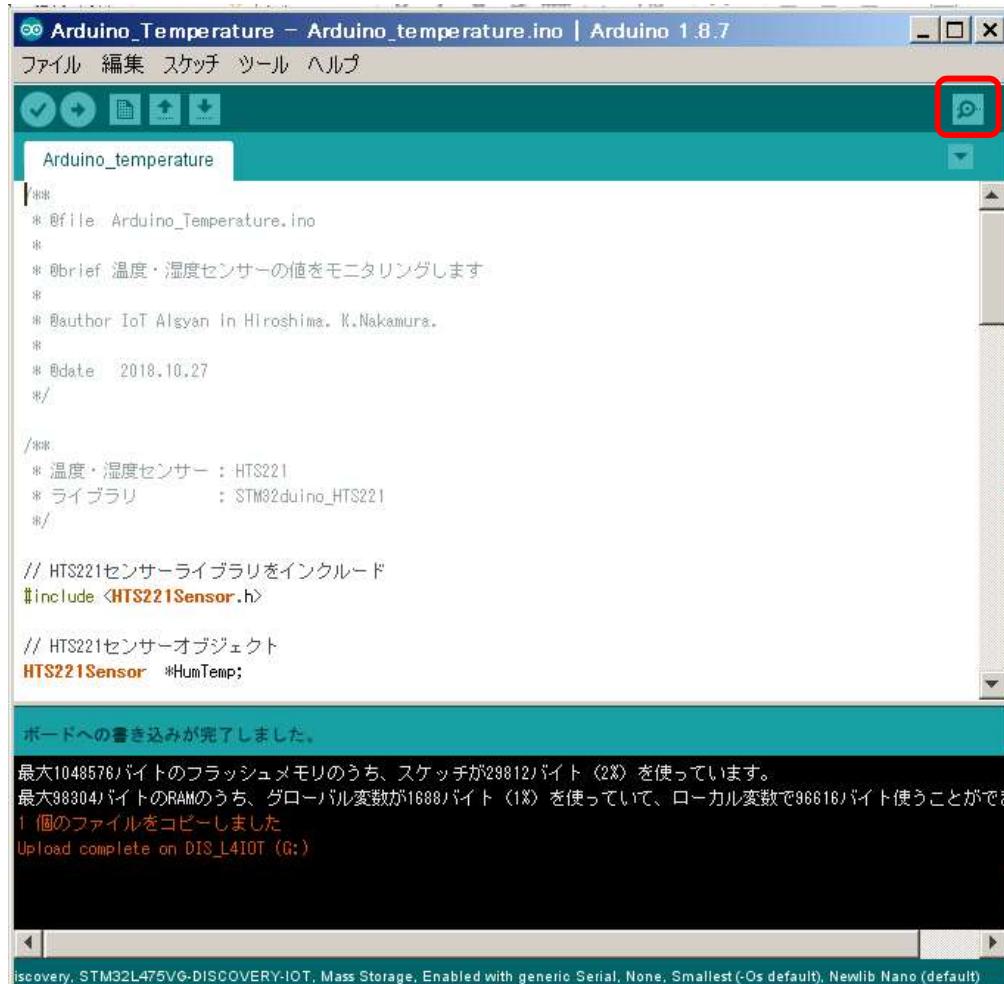
「2. STM32のLEDを点滅させる」の「2-1 サンプルコードをダウンロードする」でダウンロードして解凍したフォルダからArduino_Temperatureを開きます。

サンプルコードの書き込みの手順は

「2. STM32のLEDを点滅させる」の「2-2 STM32にサンプルコードを書込む」同様です。

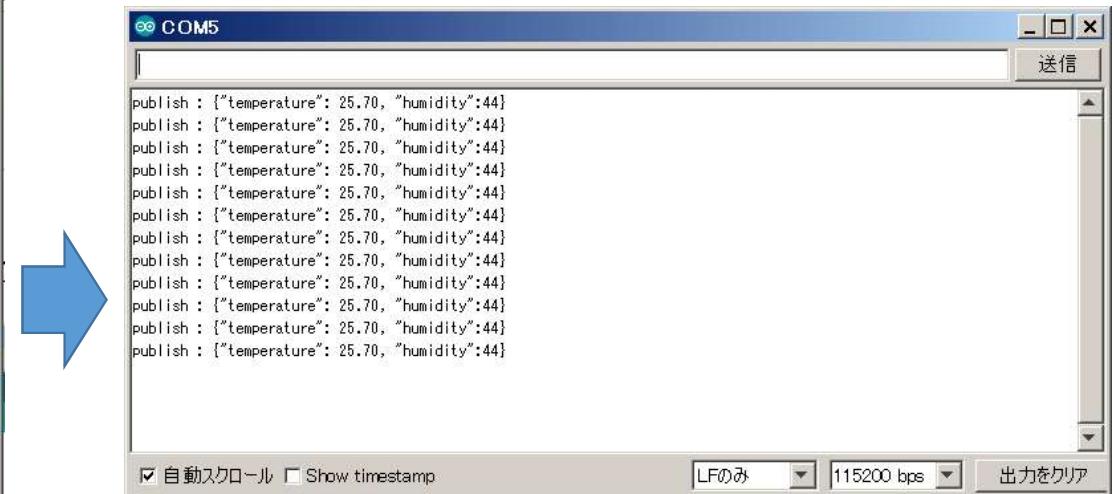


サンプルコードの書き込み後、右上の虫メガネマークを押してSTM32からUSBを経由して送られてくるデータをモニタします。



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** Arduino_Temperature - Arduino_temperature.ino | Arduino 1.8.7
- Menu Bar:** ファイル 編集 スケッチ ツール ヘルプ
- Toolbar:** Includes icons for Open, Save, Print, and Upload.
- Code Area:** The code is for monitoring temperature and humidity using an HTS221 sensor. It includes comments about the sensor type, library, and date. It also includes the HTS221Sensor.h header file and initializes a variable *HumTemp;.
- Status Bar:** Displays the message "ボードへの書き込みが完了しました。" (Write to board completed), memory usage information, and "Upload complete on DIS_L4IOT (Q:)".
- Bottom Status Bar:** Shows the connection as "discovery, STM32L475VG-DISCOVERY-IOT, Mass Storage, Enabled with generic Serial, None, Smallest (-Os default), Newlib Nano (default)".



The screenshot shows the COM5 monitor window with the following details:

- Title Bar:** COM5
- Content Area:** Displays a series of JSON objects being published over serial port. Each object contains "temperature": 25.70 and "humidity": 44.
- Bottom Control Panel:** Includes checkboxes for "自動スクロール" (Auto Scroll) and "Show timestamp", and dropdowns for "LFのみ" (LF only), "115200 bps", and "出力をクリア" (Clear output).

ボーレートを
115200bpsに設定

(参考) 書き込んだサンプルプログラムの内容

Arduino 標準ライブラリはここを参照のこと
<http://www.musashinodenpa.com/arduino/ref/>
<https://www.arduino.cc/en/Reference/Libraries>

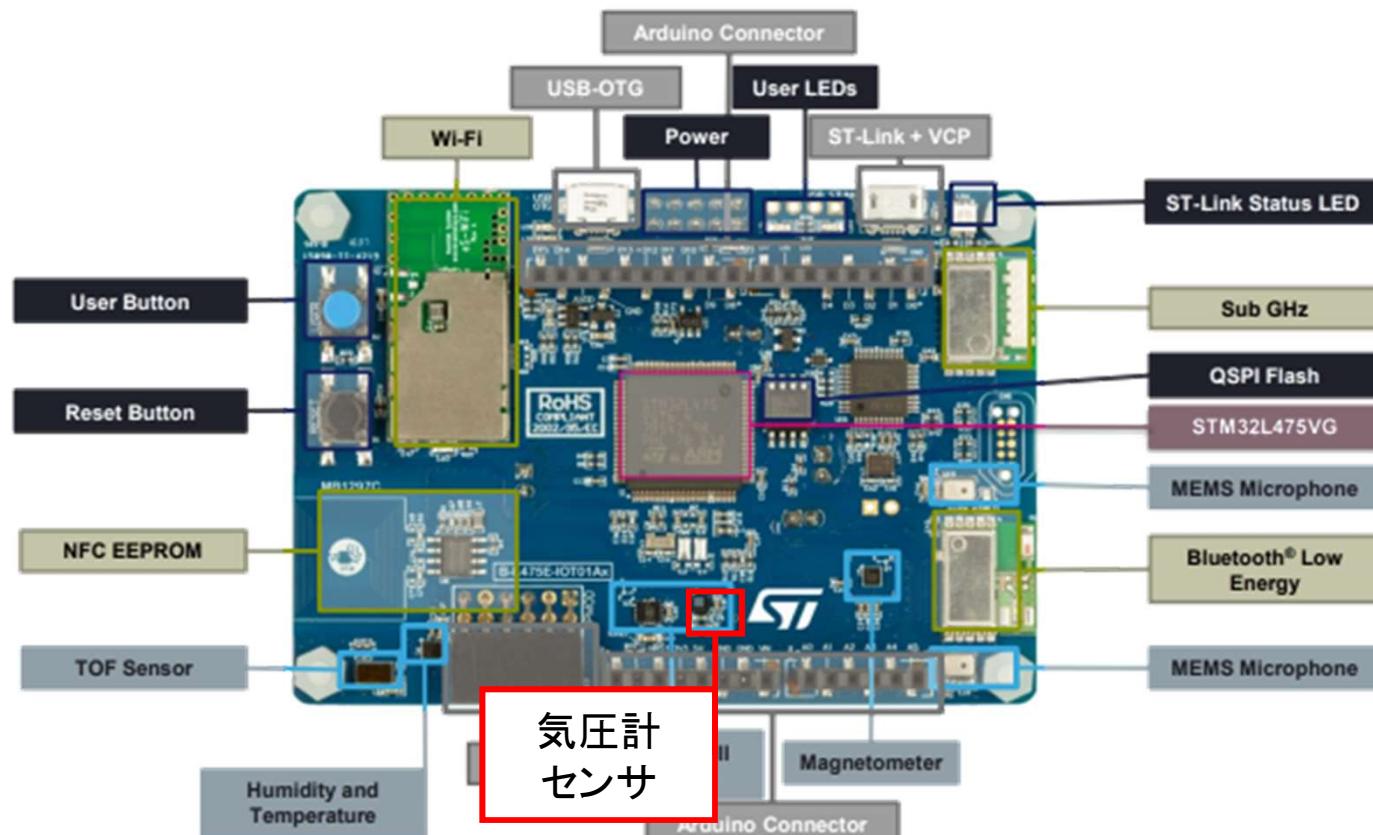
```
void setup() {  
    :  
    // I2Cバス接続オブジェクトのインスタンス生成と初期化  
    dev_i2c = new TwoWire(I2C2_SDA, I2C2_SCL);  
    dev_i2c->begin();  
  
    // HTS221センサーオブジェクトのインスタンス生成と初期化  
    // (HTS221SensorではI2Cを使用します)  
    HumTemp = new HTS221Sensor(dev_i2c);  
    HumTemp->Enable();  
}
```

```
void loop() {  
    :  
    // 温度の取得  
    HumTemp->GetTemperature(&temperature);  
  
    // 湿度の取得  
    HumTemp->GetHumidity(&humidity);  
  
    :  
}
```

STM32Duino-HTS221の使い方はここ
<https://github.com/stm32duino/HTS221>

3-2 気圧計センサをUSB経由でモニタ

気圧センサのライブラリをインストールし、その後STM32にサンプルコードを書き込み
センサからのデータをUSB経由でモニタします。



3-2 気圧計センサをUSB経由でモニタ

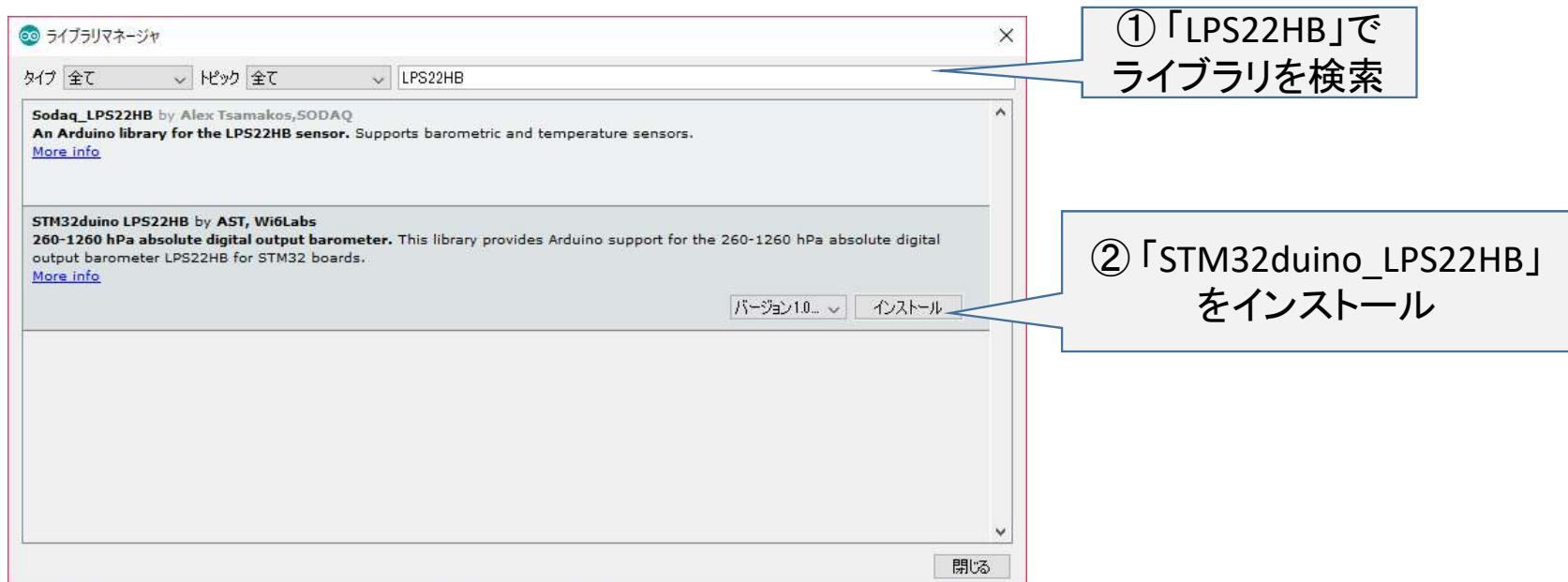
■STMduino_LPS22HB のインストール

Discovery Kitの気圧計センサーはSTM32と同じ STMicroelectronics製の LPS22HBを使用しています。ライブラリマネージャで「STM32duino_LPS22HB」をインストールします。

気圧計センサ: **LPS22HB**

ライブラリ : **STMduino_LPS22HB**

ライブラリのインストール手順は温度・湿度センサのライブラリのインストールと同様です。

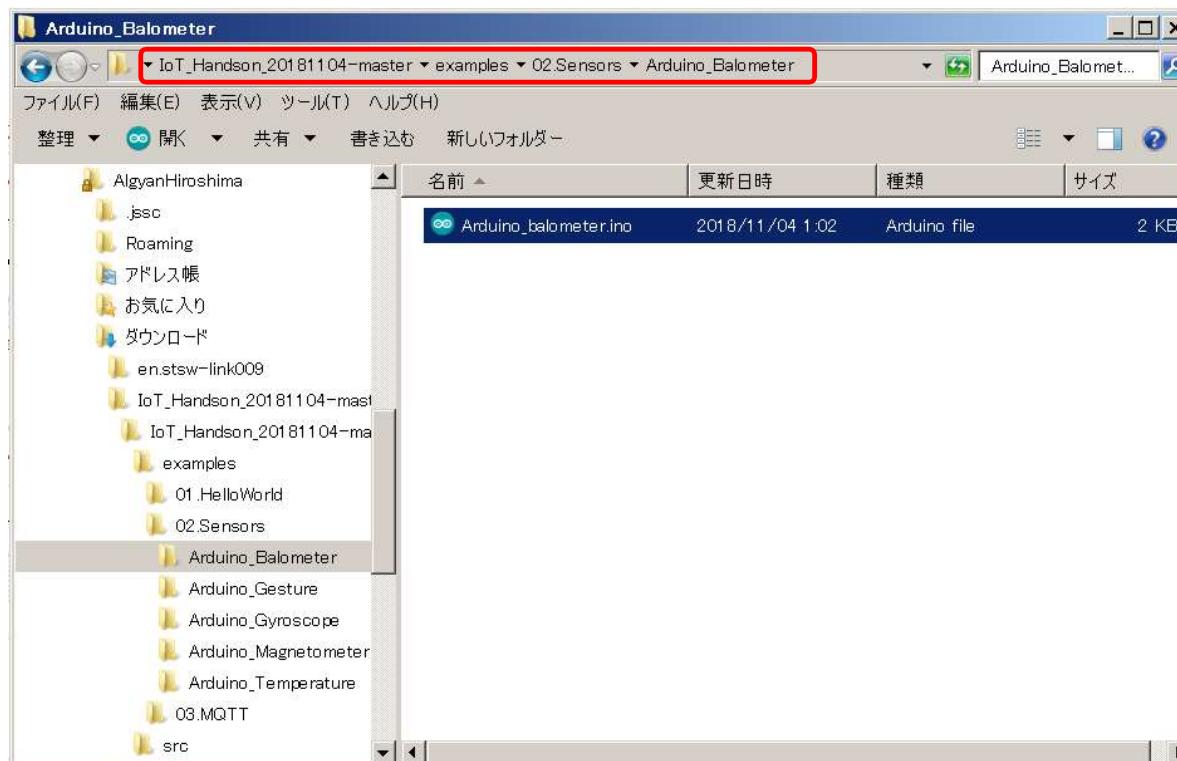


■サンプルコードを書込む

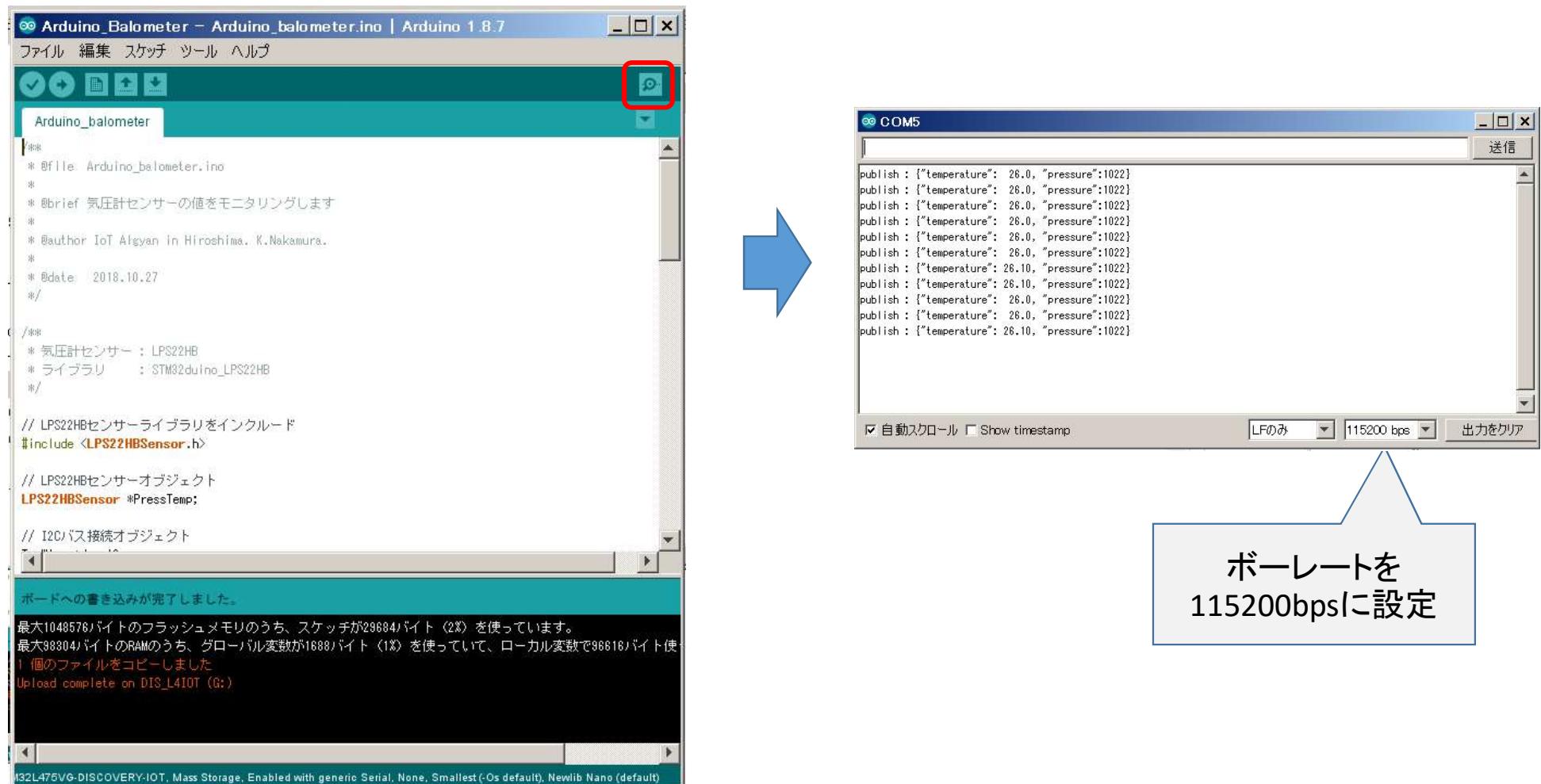
「2. STM32のLEDを点滅させる」の「2-1 サンプルコードをダウンロードする」でダウンロードして解凍したフォルダからArduino_Balometerを開きます。

サンプルコードの書き込みの手順は

「2. STM32のLEDを点滅させる」の「2-2 STM32にサンプルコードを書込む」同様です。



サンプルコードの書き込み後、右上の虫メガネマークを押してSTM32からUSBを経由して送られてくるデータをモニタします。



(参考) 書き込んだサンプルプログラムの内容

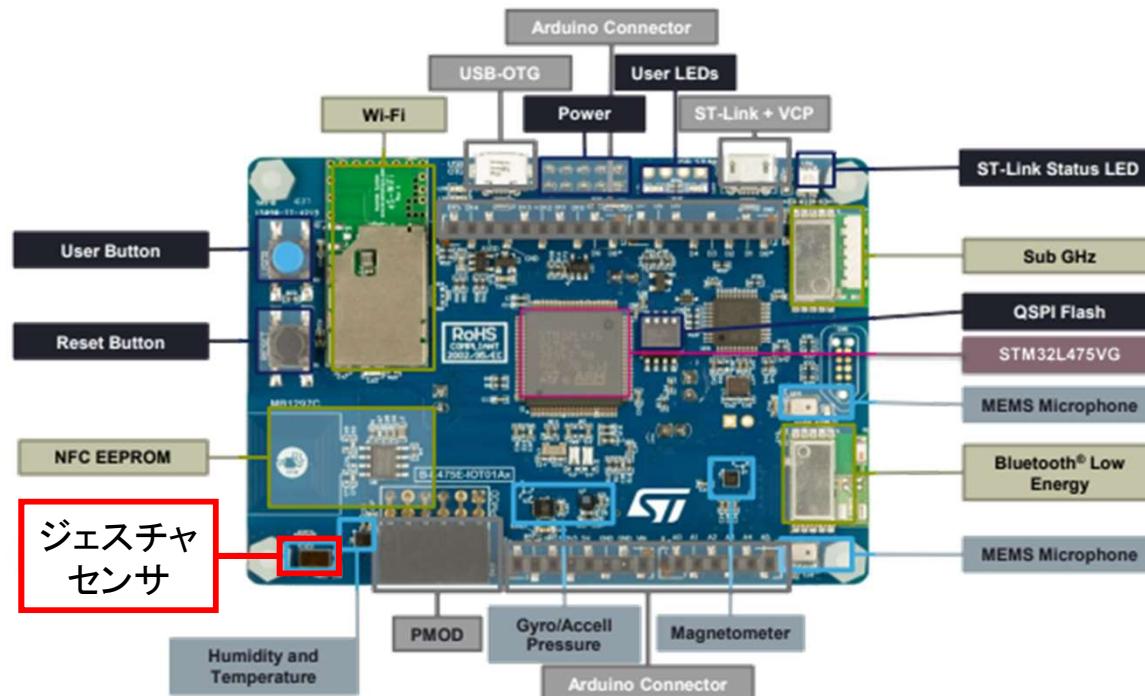
```
void setup() {  
    :  
    // I2Cバス接続オブジェクトのインスタンス生成と初期化  
    dev_i2c = new TwoWire(I2C2_SDA, I2C2_SCL);  
    dev_i2c->begin();  
  
    // LPS22HBセンサーオブジェクトのインスタンス生成  
    // (LPS22HB SensorではI2Cを使用します)  
    PressTemp = new LPS22HBSensor(dev_i2c);  
    PressTemp->Enable();  
}
```

```
void loop() {  
    :  
    // 温度の取得  
    PressTemp->GetTemperature(&temperature);  
  
    // 気圧の取得  
    PressTemp->GetPressure(&pressure);  
    :  
}
```

STM32Duino_STの使い方はここ
<https://github.com/stm32duino/LPS22HB>

3-3 ジェスチャセンサをUSB経由でモニタ

ジェスチャセンサのライブラリをインストールし、その後STM32にサンプルコードを書き込みセンサからのデータをUSB経由でモニタします。



3-3 ジェスチャセンサをUSB経由でモニタ

■STMduino_VL53L0X のインストール

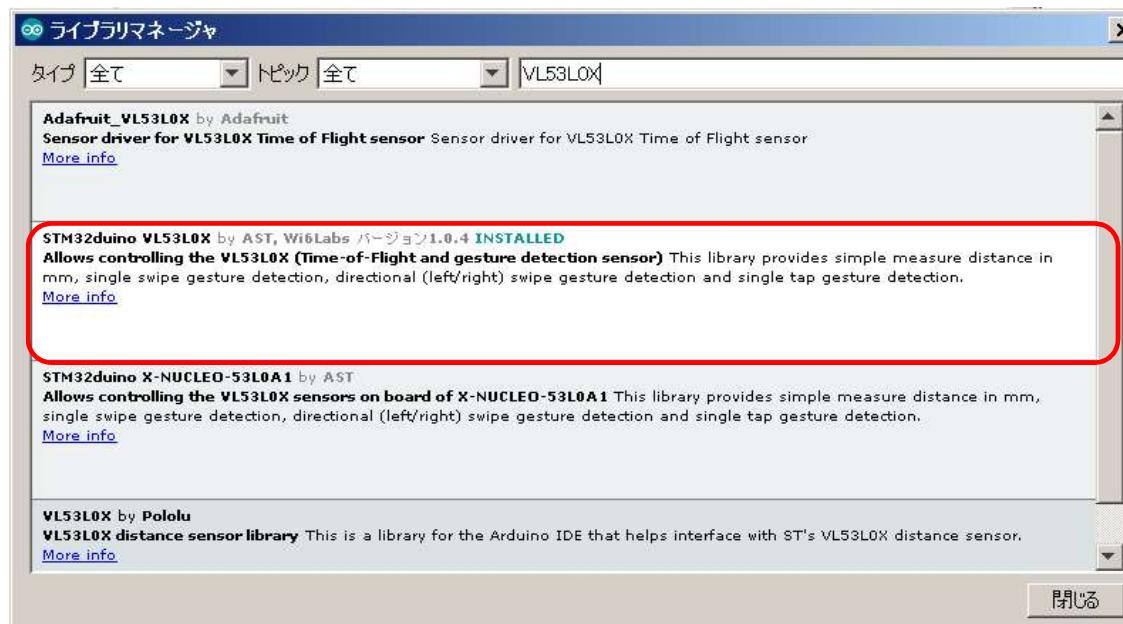
ToF(Time-Of-Flight)、光の飛行時間(反射時間)を測定することで距離を測定します。

複数のジェスチャセンサを組み合わせると、手の動きを認識したりすることが可能となります。

ジェスチャセンサ : [VL53L0X](#)

ライブラリ : [STMduino_VL53L0X](#)

ライブラリのインストール手順は温度・湿度センサのライブラリのインストールと同様です。



3-3 ジェスチャセンサをUSB経由でモニタ

■ ArduinoSTL のインストール

Standard Template Library を使用できるようにします。Vector, List, Map, Queueなどのコンテナを使用できるようになります。

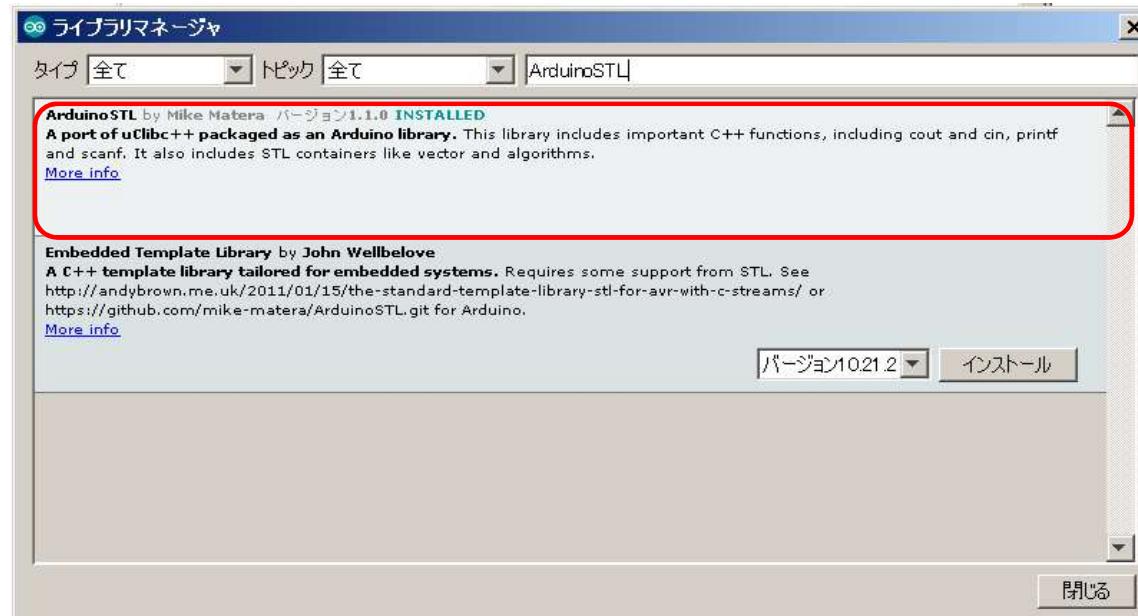
今回のサンプルでは、配列データを簡単に扱える vector を使用しています。

ライブラリ :ArduinoSTL

ライブラリのインストール手順は

「1-2 STM32にstm32duino/Arduino_Core_STM32をインストールする」の

「Step2 [Arduino IDE のボードマネージャーでstm32duino/Arduino_Core_STM32をダウンロードします](#)」同様です。

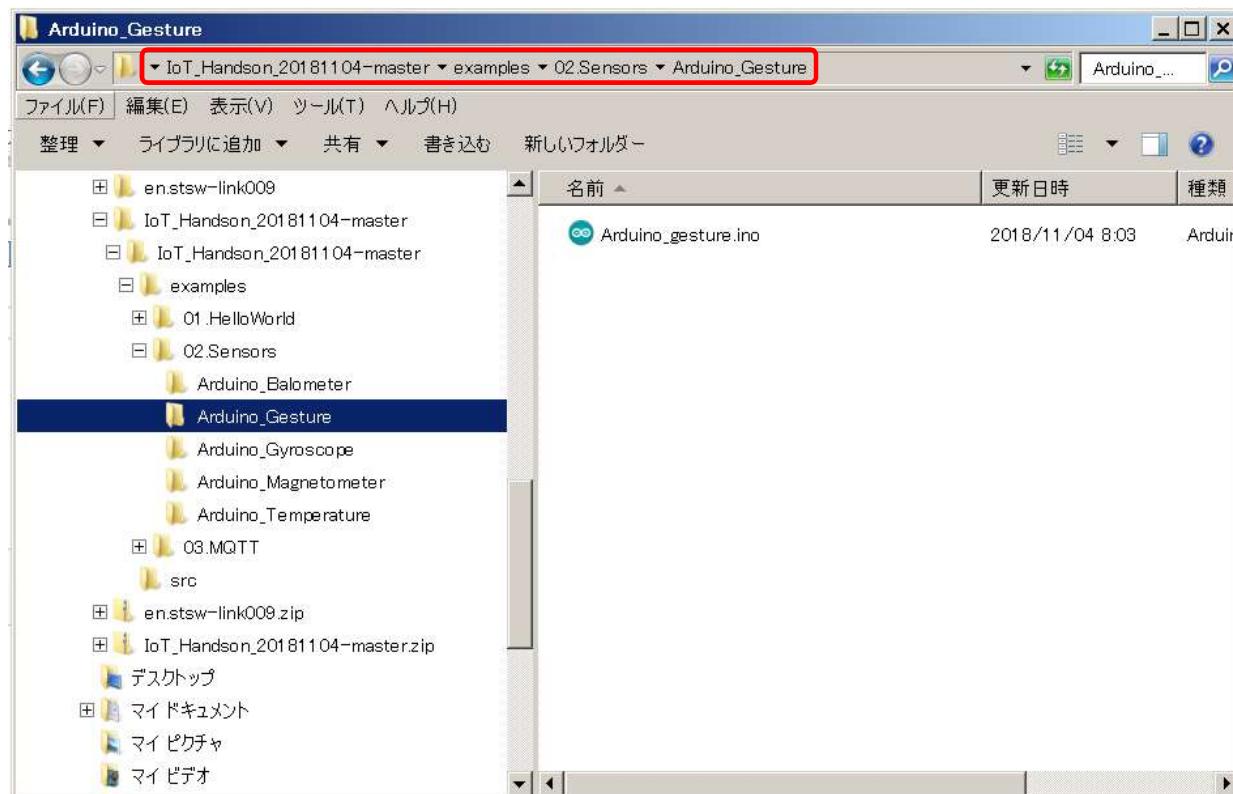


■サンプルコードを書込む

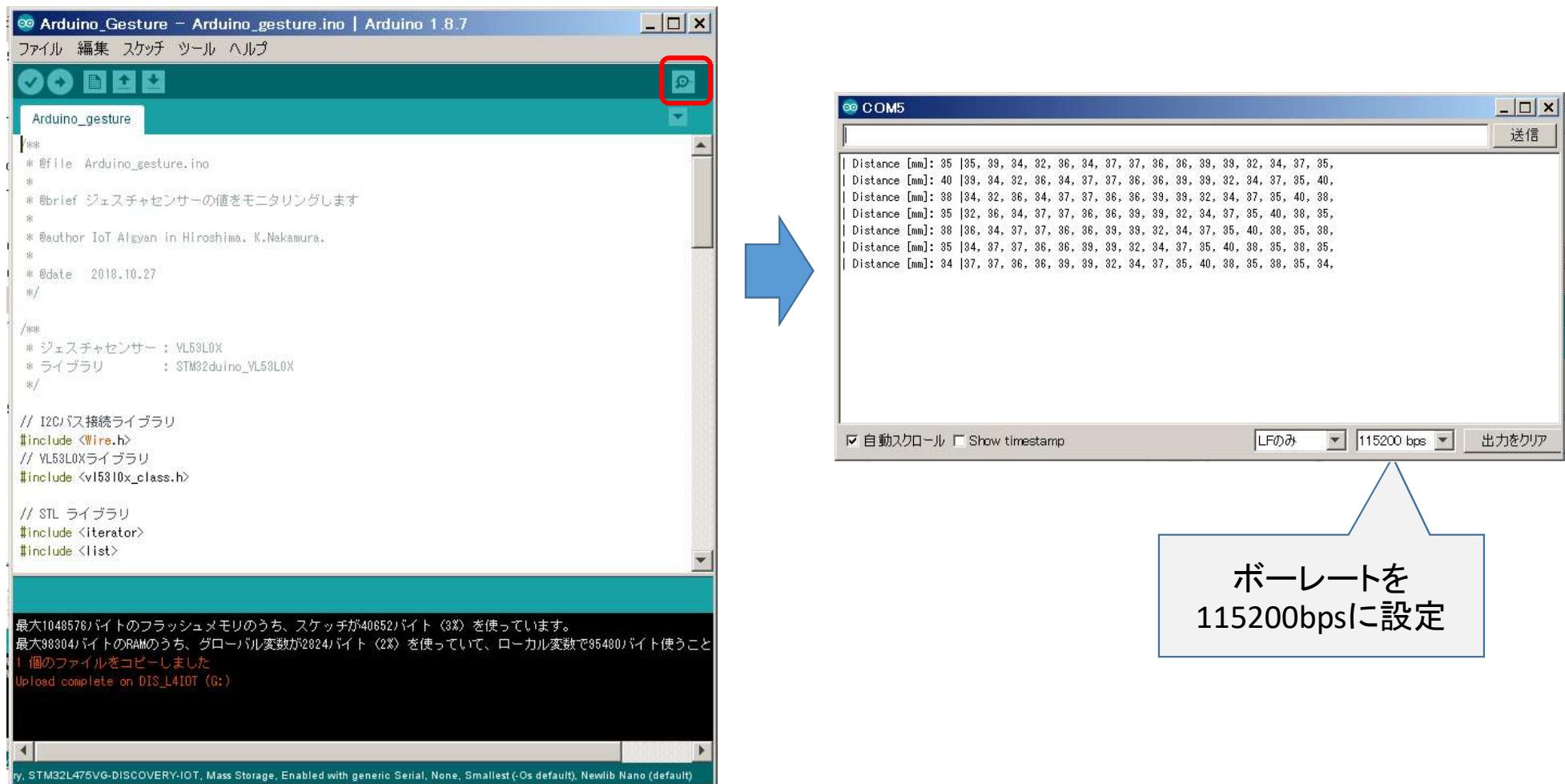
「2. STM32のLEDを点滅させる」の「2-1 サンプルコードをダウンロードする」でダウンロードして解凍したフォルダからArduino_Gestureを開きます。

サンプルコードの書き込みの手順は

「2. STM32のLEDを点滅させる」の「2-2 STM32にサンプルコードを書込む」同様です。



サンプルコードの書き込み後、右上の虫メガネマークを押してSTM32からUSBを経由して送られてくるデータをモニタします。



(参考) 書き込んだサンプルプログラムの情報

STM32や、センサー等にはデータシート、ユーザーマニュアルが公開されています。
また他に使用している方が使用方法を公開している場合も多数あります。
まずは、情報を入手してみましょう。

使い方サイト : <https://github.com/stm32duino/VL53L0X>

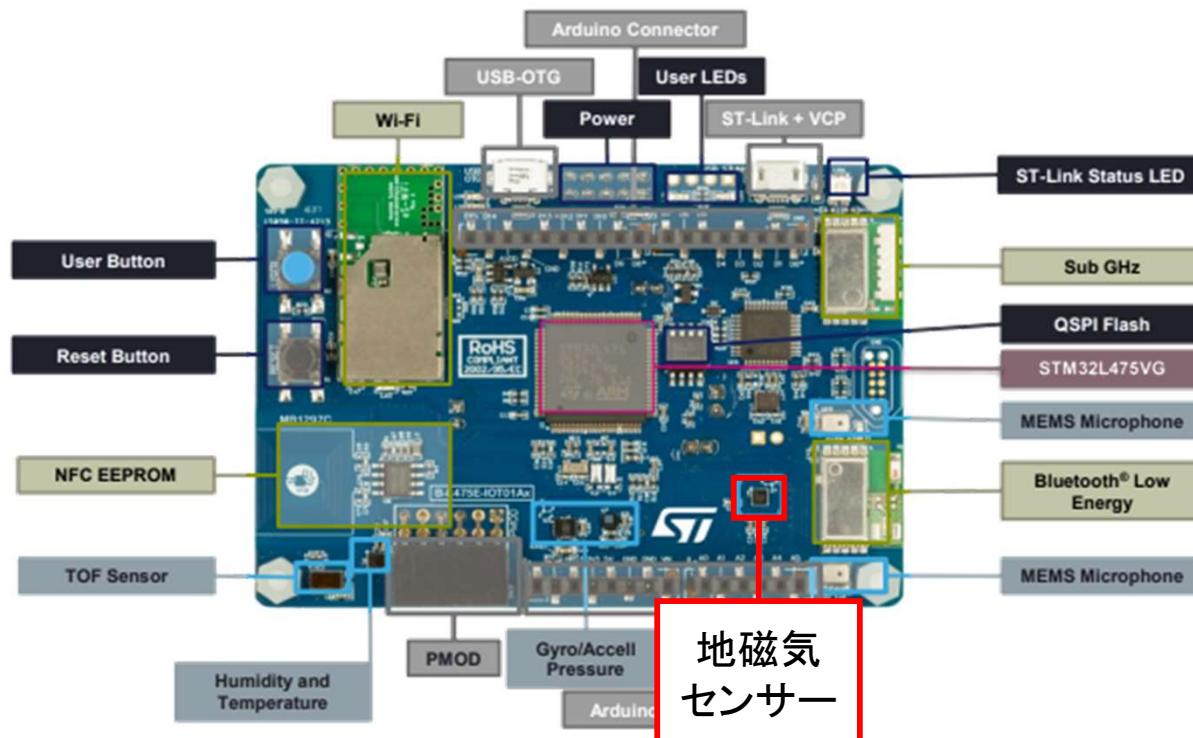
データシート : <https://www.st.com/resource/en/datasheet/vl53l0x.pdf>

ユーザーマニュアル : <en.DM00279088.pdf>

keyword : VL53L0X, ToF(Time-Of-Flight) ranging sensor, SPAD(光飛行時間型距離画像センサー), ジェスチャーセンサー

3-4 3軸地磁気センサをUSB経由でモニタ

3軸地磁気センサのライブラリをインストールし、その後STM32にサンプルコードを書き込みセンサからのデータをUSB経由でモニタします。



3-4 3軸地磁気センサをUSB経由でモニタ

■STMduino_LIS3MDL のインストール

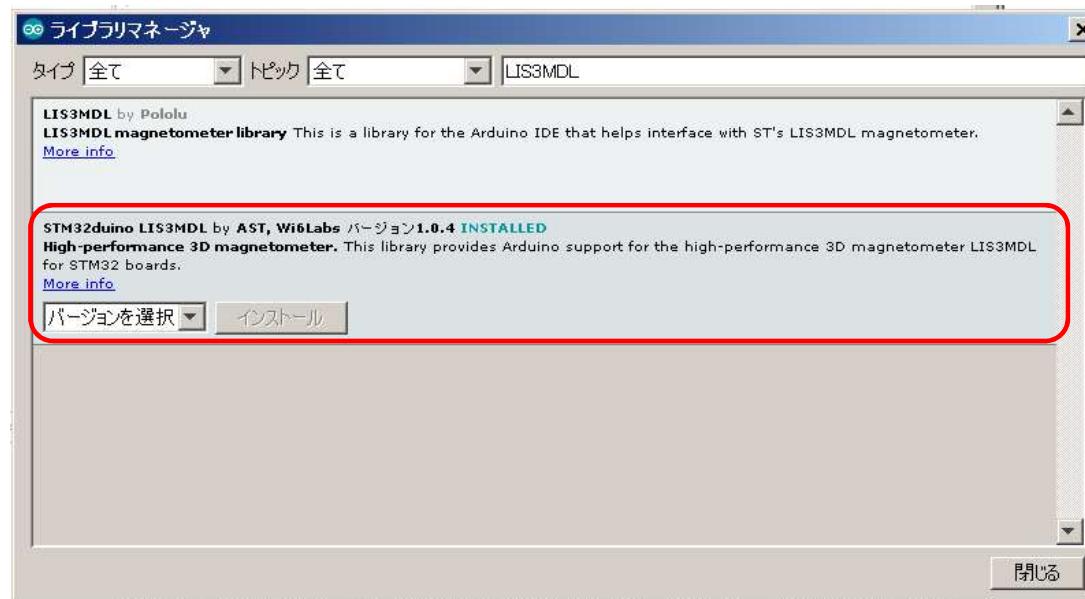
3軸地磁気センサ : LIS3MDL

ライブラリ: STMduino_LIS3MDL

サンプル: 02.Sensors/Arduino_Magnetometer/Arduino_magnetometer.ino

使い方サイト: <https://github.com/stm32duino/LIS3MDL>

ライブラリのインストール手順は温度・湿度センサのライブラリのインストールと同様です。

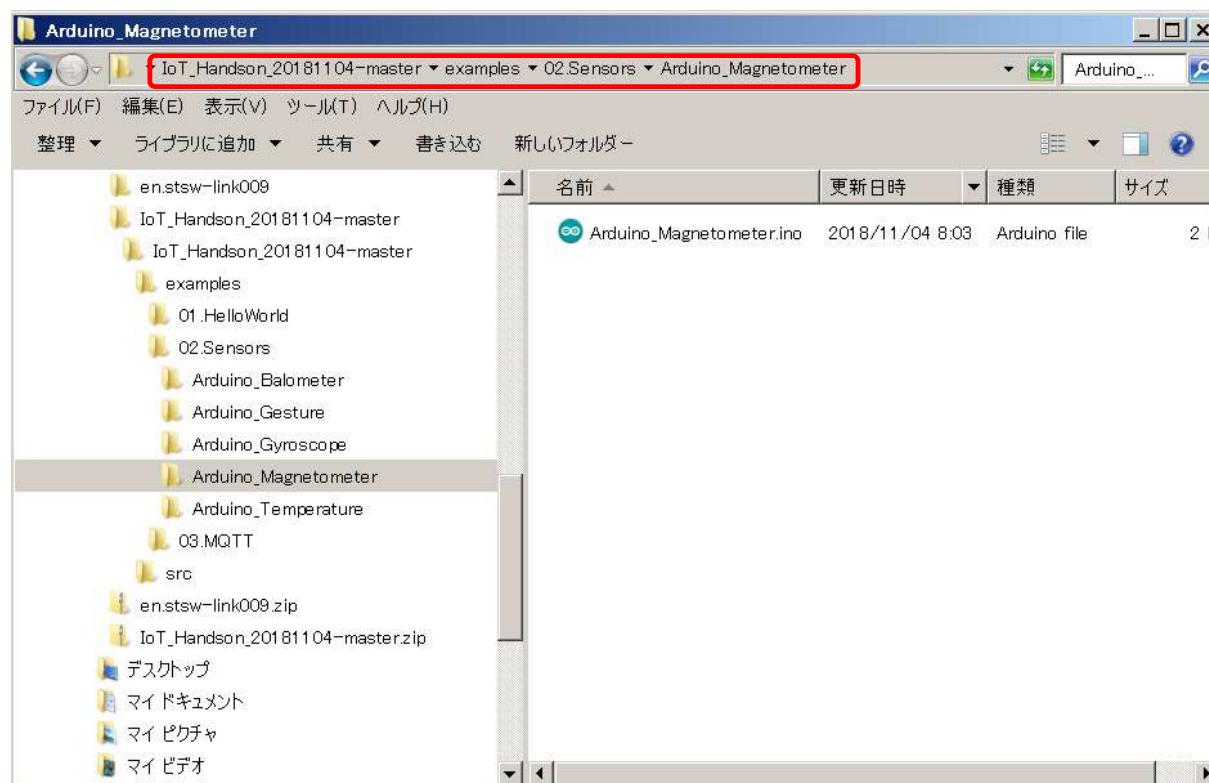


■サンプルコードを書込む

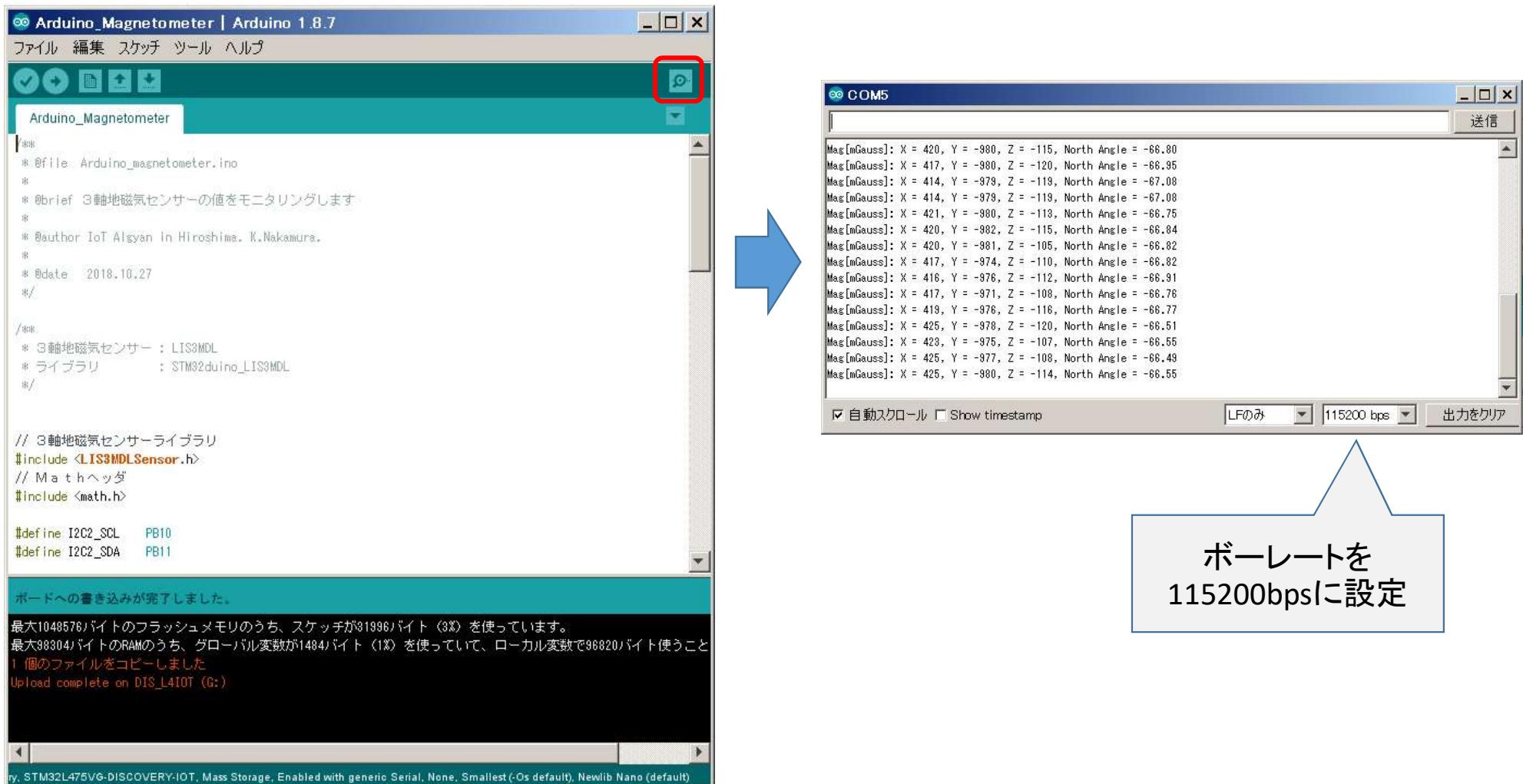
「2.[STM32のLEDを点滅させる](#)」の「2-1 サンプルコードをダウンロードする」でダウンロードして解凍したフォルダから[Arduino_Magnetometer](#)を開きます。

サンプルコードの書き込みの手順は

「2.[STM32のLEDを点滅させる](#)」の「2-2 [STM32にサンプルコードを書込む](#)」同様です。

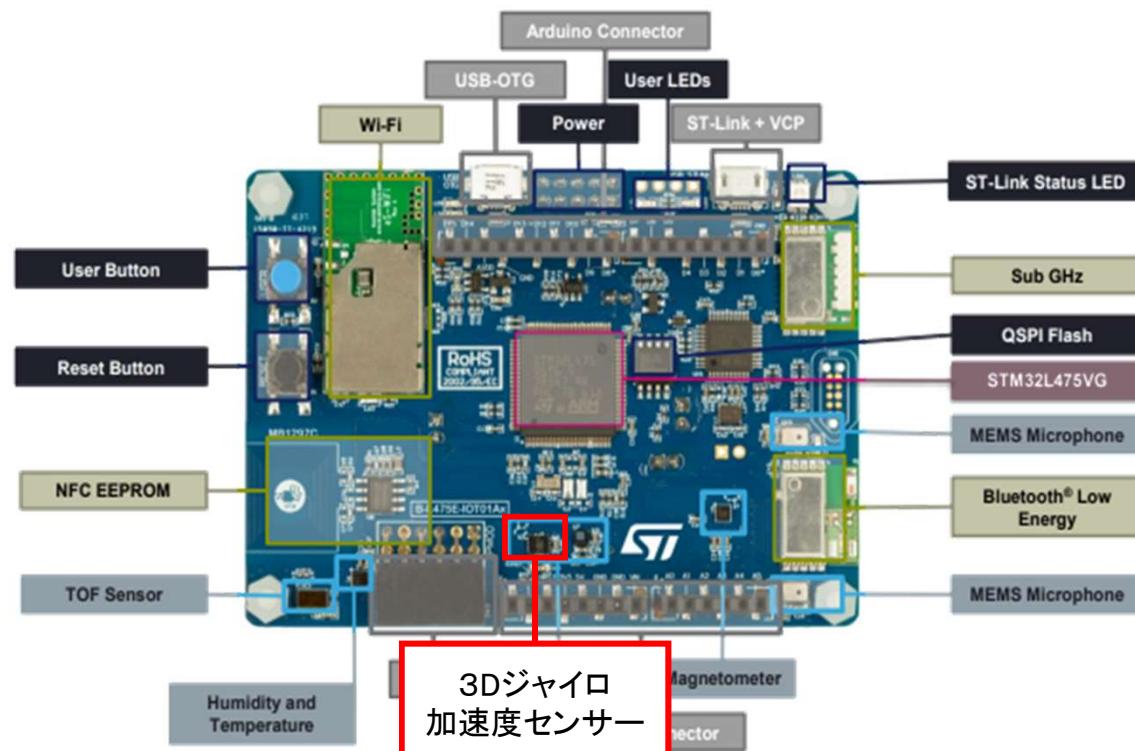


サンプルコードの書き込み後、右上の虫メガネマークを押してSTM32からUSBを経由して送られてくるデータをモニタします。



3-5 3DジャイロセンサをUSB経由でモニタ

3Dジャイロセンサのライブラリをインストールし、その後STM32にサンプルコードを書き込み
センサからのデータをUSB経由でモニタします。



3-5 3DジャイロセンサをUSB経由でモニタ

■STMduino_LSM6DSLのインストール

3Dジャイロ・加速度センサ : [LSM6DSL](#)

ライブラリ : [STMduino_LSM6DSL](#)

サンプル : 02.Sensors/Arduino_Gyroscope/Arduino_gyroscope.ino

使い方サイト : <https://github.com/stm32duino/LSM6DSL>

ライブラリのインストール手順は温度・湿度センサのライブラリのインストールと同様です。

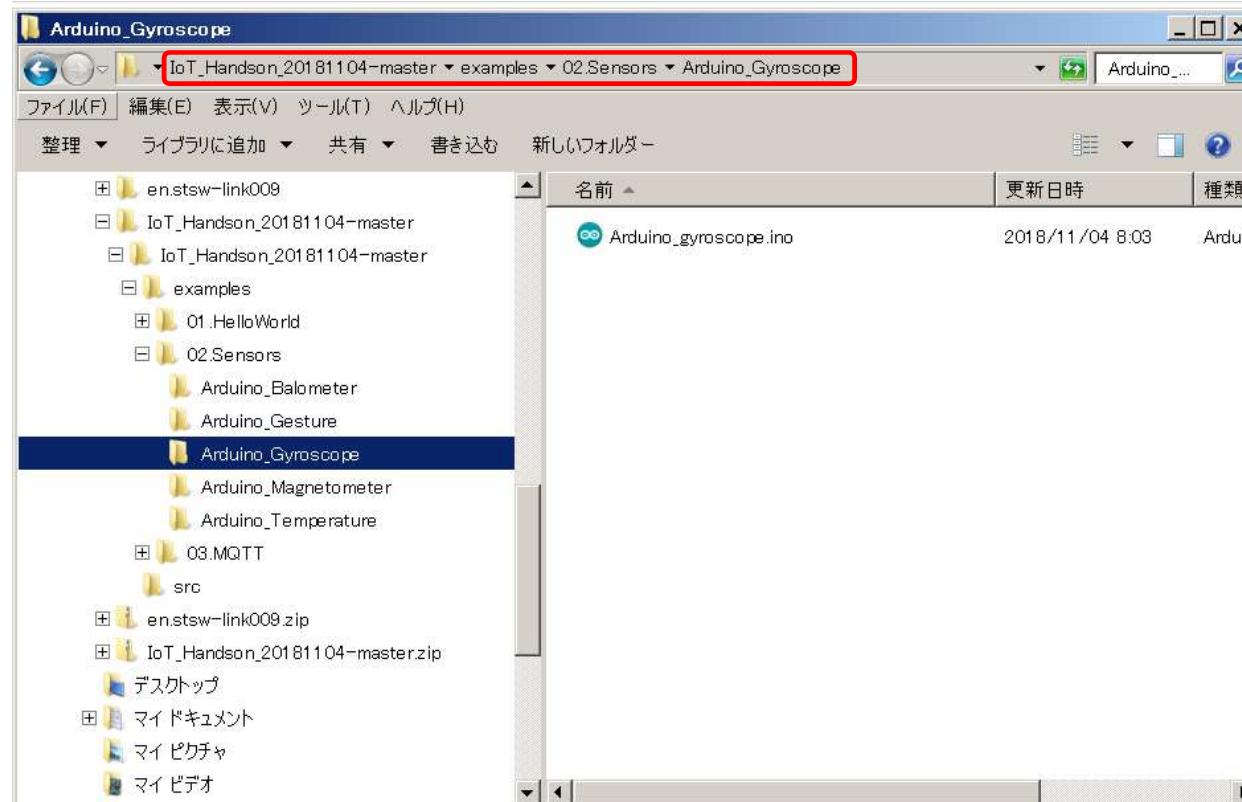


■サンプルコードを書込む

「2. STM32のLEDを点滅させる」の「2-1 サンプルコードをダウンロードする」でダウンロードして解凍したフォルダからArduino_Gyroscopeを開きます。

サンプルコードの書き込みの手順は

「2. STM32のLEDを点滅させる」の「2-2 STM32にサンプルコードを書込む」同様です。



サンプルコードの書き込み後、右上の虫メガネマークを押してSTM32からUSBを経由して送られてくるデータをモニタします。

```
Arduino_Gyroscope - Arduino_gyroscope.ino | Arduino 1.8.7
ファイル 編集 スケッチ ツール ヘルプ

Arduino_gyroscope

/*
 * @file Arduino_gesture.ino
 *
 * @brief 3 Dジャイロセンサ・加速度センサーの値をモニタリングします
 *
 * @author IoT Algyan in Hiroshima, K.Nakamura.
 *
 * @date 2018.10.27
 */

// 3 Dジャイロセンサ・加速度センサーライブライブラリ
#include <LSM6DSL.h>

// STL ライブライブラリ
#include <iomanip>
#include <list>

#define I2C2_SCL PB10
#define I2C2_SDA PB11

ボードへの書き込みが完了しました。
最大1048576バイトのフラッシュメモリのうち、スケッチが32388バイト(3%)を使っています。
最大98304バイトのRAMのうち、グローバル変数が1684バイト(1%)を使っていて、ローカル変数で96620バイト使うこと
1 個のファイルをコピーしました
Upload complete on DIS_L4IOT (G:)
```

STM32L475VG-DISCOVERY-IOT, Mass Storage, Enabled with generic Serial, None, Smallest (-Os default), Newlib Nano (default)

```
Single Tap Detected!
Tilt Detected!
```

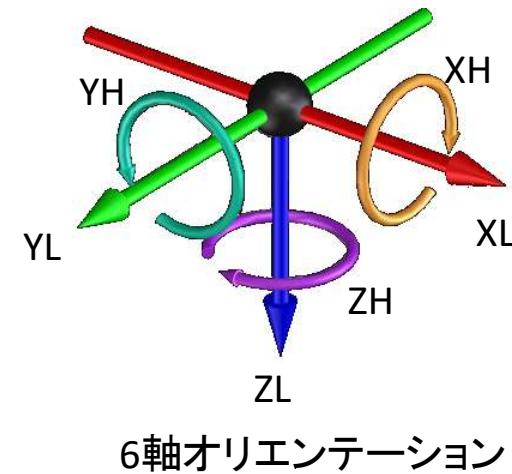
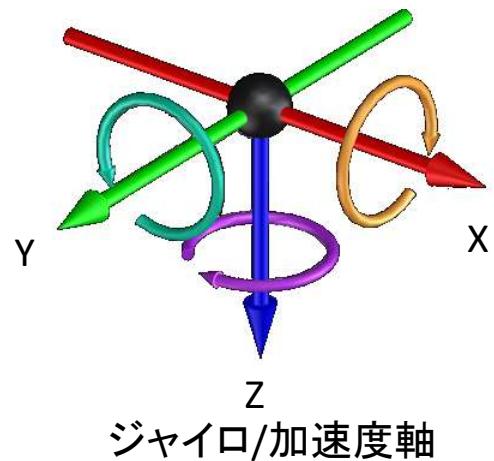
自動スクロール Show timestamp LFのみ 115200 bps 出力をクリア

STM32を動かすとジャイロの
出力結果が表示されます

ポーレートを
115200bpsに設定

(参考) 書き込んだサンプルプログラムの情報

3Dジャイロ・加速度センサをモーションセンサとしても使用できます。



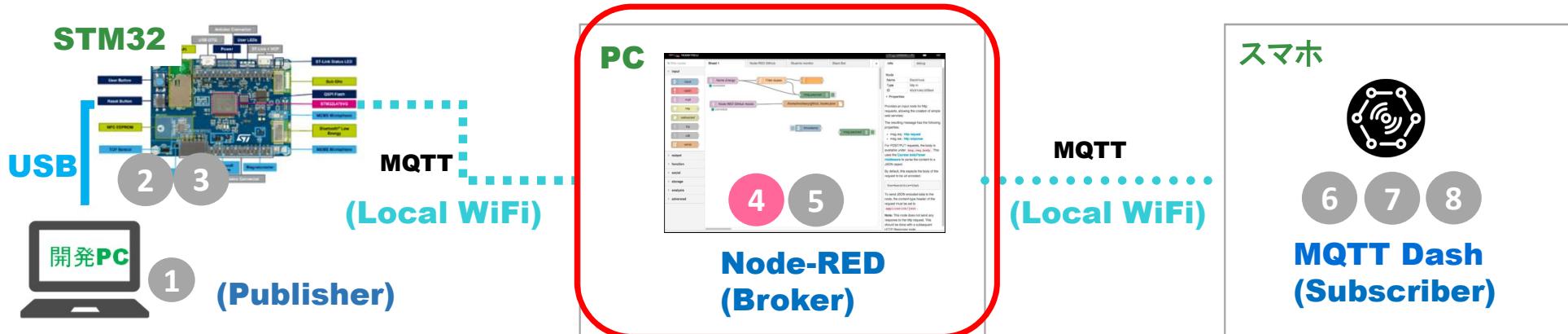
4 PCにNode.jsとNode-REDをインストールする

PCにNode.jsとNode-REDをインストールし、Brokerの環境を作ります。

4-1 Node.js のインストール

4-2 Node-REDのインストール

4-3 Node-RED側のMQTTインストール

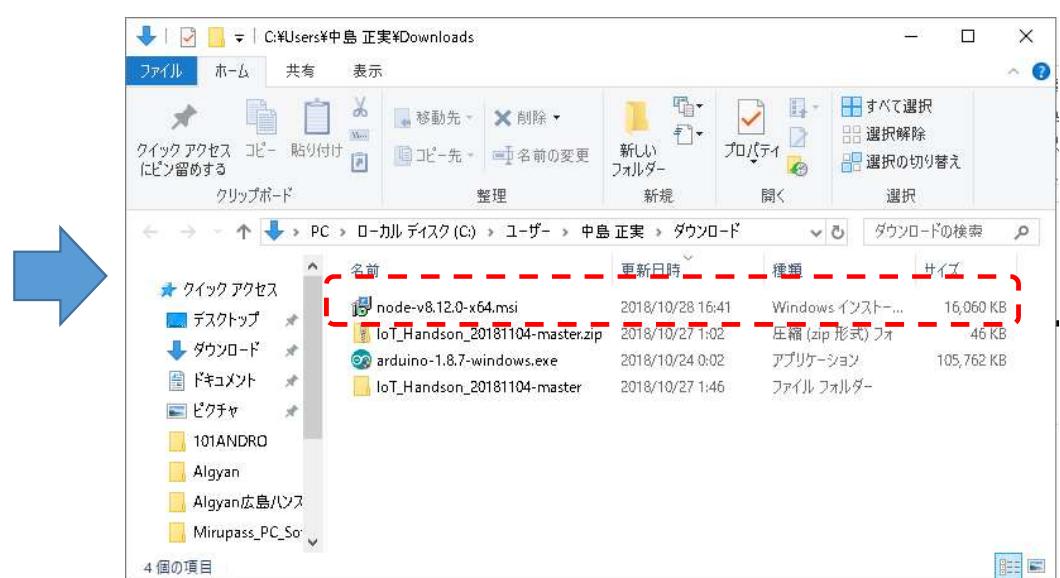


4-1 Node.jsをインストールする

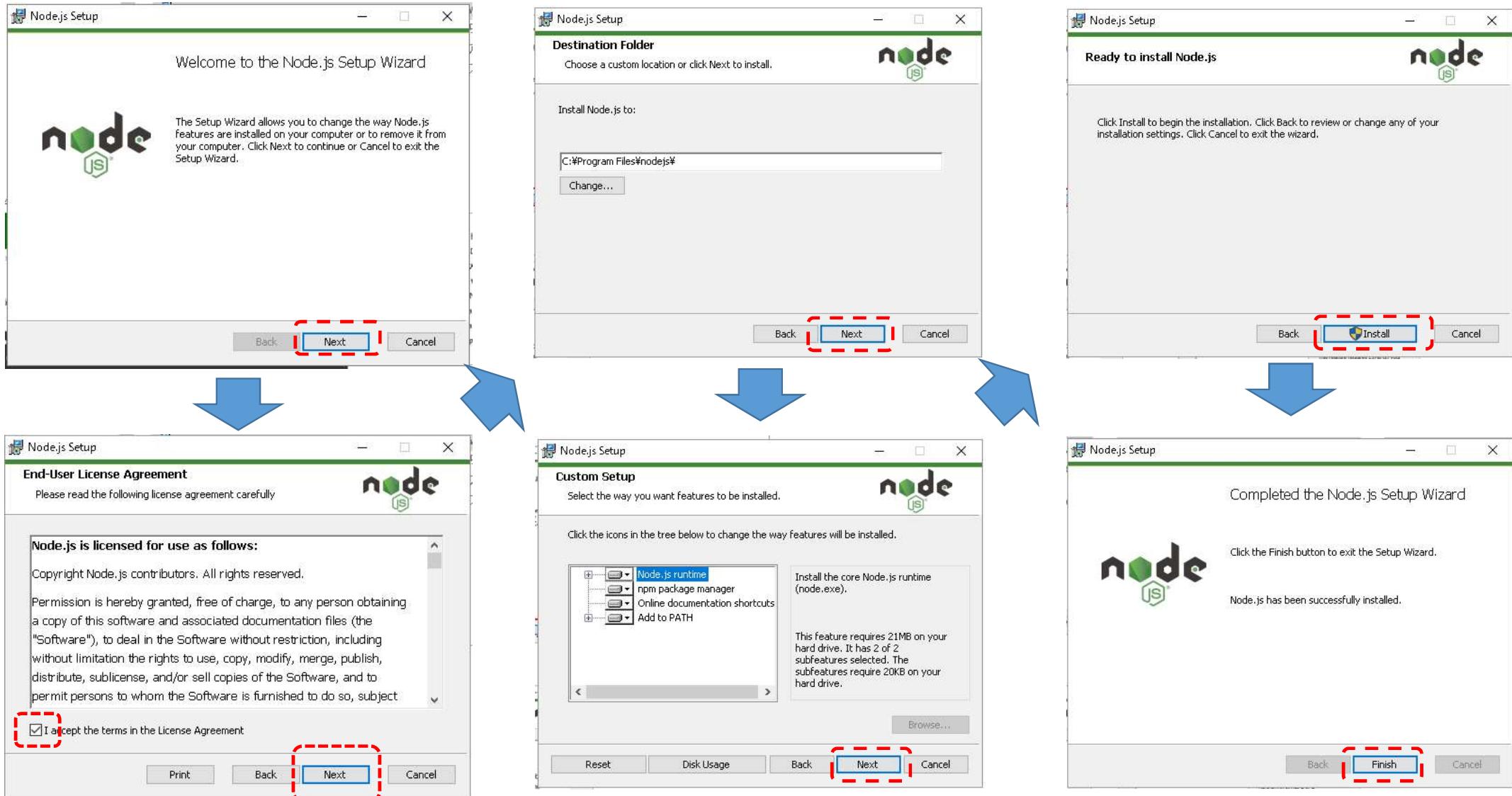
Node.js® は、[Chrome の V8 JavaScript エンジン](#) で動作する JavaScript 環境です。サーバーサイドのプログラムを JavaScript で構築することができます。また、開発環境で使用することにより、ローカル環境に Web サーバーを構築することもできます。後述の Node-RED を動かすエンジン部分となります。

URL : <https://nodejs.org/ja/>

The screenshot shows the official Node.js website. At the top, there's a navigation bar with links: ホーム, NODE.js とは, ダウンロード, ドキュメント, 参加する, セキュリティ, ニュース, 貢献. Below the navigation, a message says "Node.js® は、Chrome の V8 JavaScript エンジンで動作する JavaScript 環境です。". A large green button labeled "ダウンロード Windows (x64)" is centered. To its left is a box for "8.12.0 LTS" (推奨版) and to its right is a box for "11.0.0 最新版" (最新の機能). Both boxes have dashed red rectangles around them. Below these boxes, there are links for "他のバージョン | 変更履歴 | API ドキュメント" and "他のバージョン | 変更履歴 | API ドキュメント". A note below says "または、LTSのリリーススケジュールをご覧ください。" and "公式 Node.js ニュースレター Node.js Everywhere を購読しましょう。". At the bottom, there's a footer with links: □ LINUX FOUNDATION COLLABORATIVE PROJECTS, Node.js の不具合を報告 | サイトの不具合を報告 | ヘルプ, © Node.js Foundation. All Rights Reserved. Portions of this site originally © Joyent., Node.js is a trademark of Joyent, Inc. and is used with its permission. Please review the Trademark Guidelines of the Node.js Foundation., Linux Foundation is a registered trademark of The Linux Foundation.



Node.js のインストールを行います。



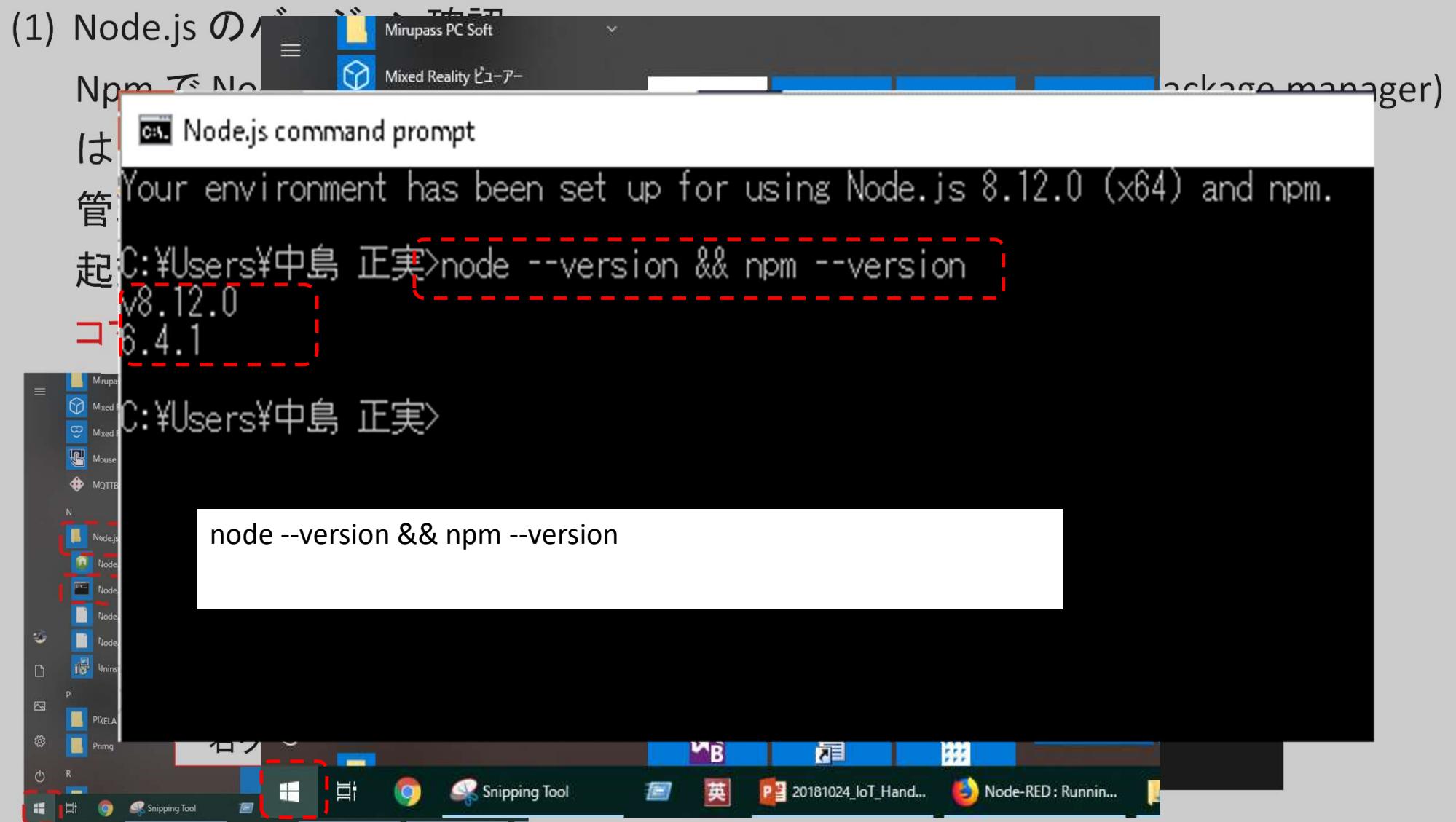
4-2 Node-REDのインストール

Node-RED は、Node.js 上で動作する、ハードウェアデバイスおよび、オンラインサービスを接続するためのツールです。

ここに記載された手順でインストールします。

URL : <https://nodered.org/docs/platforms/windows>

The screenshot shows the Node-RED documentation page for "Running on Windows". The left sidebar has a "Getting Started" section with links for Installation, Running, Adding Nodes, Upgrading, Creating your first flow, and Creating your second flow. The main content area is titled "Running on Windows" and contains instructions for setting up Node-RED on Windows. It notes that the instructions are specific to Windows 10 but should work for Windows 7 and Windows Server from 2008R2. A note states: "Note: Some of the following instructions mention the "command prompt". Where this is used, it refers to either the Windows cmd or PowerShell terminal shells. It is recommended to use PowerShell on all newer versions of Windows as this gives you access to commands and folder names that are closer to those of Linux/Mac." Below this, there's a "Quick Start" section with a link to "1. Install Node.js". The footer includes links for "Using Node-RED", "Configuration", "Security", "Command-line", "Admin", and "Working Functions".



```
(S) Node.js command prompt  
Your environment has been set up for using Node.js 8.12.0 (x64) and npm.  
C:\Users\中島 正実>node --version && npm --version  
v8.12.0  
6.4.1  
C:\Users\中島 正実>npm install -g --unsafe-perm node-red  
npm [WARN] deprecated nodemailer@1.11.0: All versions below 4.0.1 of Nodemailer are deprecated. See https://nodemailer.com/status/  
npm [WARN] deprecated mailparser@0.6.2: Mailparser versions older than v2.3.0 are deprecated  
npm [WARN] deprecated mimelib@0.3.1: This project is unmaintained  
npm [WARN] deprecated mailcomposer@2.1.0: This project is unmaintained  
npm [WARN] deprecated buildmail@2.0.0: This project is unmaintained  
C:\Users\中島 正実\AppData\Roaming\npm\node-red -> C:\Users\中島 正実\AppData\Roaming\npm\node_modules\node-red\node-red.js  
C:\Users\中島 正実\AppData\Roaming\npm\node-red-pi -> C:\Users\中島 正実\AppData\Roaming\npm\node_modules\node-red\bin\node-red-pi  
+ node-red@0.19.4  
added 11 packages from 1 contributor, removed 1 package and updated 15 packages in 44.363s  
C:\Users\中島 正実>  
  
npm install -g --unsafe-perm node-red
```

(3) Node-RED IDEをインストール

コマンド: npm install --global --production windows-build-tools

```
Administrator: Windows PowerShell
Your environment has been set up for using Node.js 8.12.0 (x64) and npm.
C:\Windows\System32>npm install --global --production windows-build-tools
> windows-build-tools@5.0.0 postinstall C:\Users\中島 正実\AppData\Roaming\npm\node_modules\windows-build-tools
> node ./dist/index.js

Administrator: Windows PowerShell
Your environment has been set up for using Node.js 8.12.0 (x64) and npm.
C:\Windows\System32>npm install --global --production windows-build-tools
> windows-build-tools@5.0.0 postinstall C:\Users\中島 正実\AppData\Roaming\npm\node_modules\windows-build-tools
> node ./dist/index.js

Administrator: Windows PowerShell
npm install --global --production windows-build-tools
[=====]
Downloaded python
[=====]
Downloading vs_BuildTools.exe
[=====] 100.0% of 1.12 MB (1.12 MB/s)

ACTION :1 rules :
: to our content delivery network servers. This might be a temporary server outage, or it could be an issue with your internet connection. Fix
: First, select View Logs and try to go to the URL in the report. If you can, try installing Visual Studio again. If you can not access the URL in
: the View Logs report, or if you are on an unreliable network, please try our recommended steps to [install Visual Studio 2017 on low-&ndash;bandwidth or
: unreliable network environments](https://docs.microsoft.com/en-us/visualstudio/install/install-vs-inconsistent-quality-network).", "Resources": [{"Locale
: "en-US", "Message": "**Error 0x80072ee7**: No internet connection**\nThere is a problem connecting to our content delivery network servers. This m
ight be a temporary server outage, or it could be an issue with your internet connection.**\nFix**"}, {"First, select View Logs and try to go to
the URL in the report. If you can, try installing Visual Studio again. If you can not access the URL in the View Logs report, or if you are on an unrel
2018-10-28T17:23:00 : Verbose : Calling SetupEngine.Installer.EvaluateInstallParameters. [channelId: VisualStudio.15.Release, productId: Microsoft.Visua
lStudio.Product.BuildTools, installationPath: 'C:\Program Files (x86)\Microsoft Visual Studio\2017\BuildTools', languages: 'ja-JP' selectedPackageReferenc
es.length: 13]
2018-10-28T17:23:00 : Verbose : SetupEngine.Installer.EvaluateInstallParameters succeeded. [channelId: VisualStudio.15.Release, productId: Microsoft.Visua
lStudio.Product.BuildTools, installationPath: 'C:\Program Files (x86)\Microsoft Visual Studio\2017\BuildTools', languages: 'ja-JP' selectedPackageReferenc
es.length: 13]
2018-10-28T17:23:00 : Verbose : Calling SetupEngine.Installer.InstallProduct. [channelId: VisualStudio.15.Release, productId: Microsoft.VisualStudio.Pro
duct.BuildTools, installationPath: 'C:\Program Files (x86)\Microsoft Visual Studio\2017\BuildTools']
----- Python -----
Successfully installed Python 2.7.
```

終了後画面閉じる

(4) Node-RED を起動

コマンドプロンプトを立ち上げ、“node-red”を起動します。

コマンド: node-red

The screenshot shows a Windows Command Prompt window titled "node-red". The command "node-red" has been entered and is highlighted with a red dashed box. The output shows the Node-RED startup sequence:

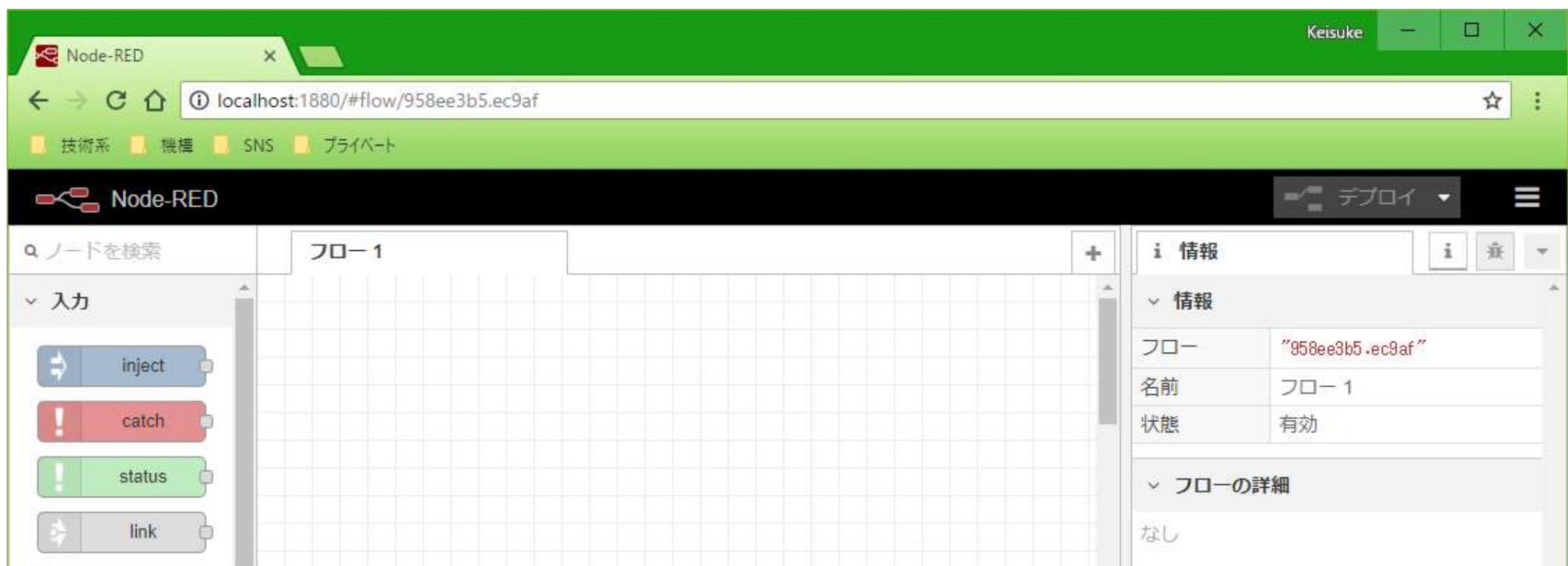
```
C:\$Users\$中島 正実>node-red
28 Oct 17:28:40 - [info] [node-red]
Welcome to Node-RED
=====
28 Oct 17:28:40 - [info] Node-RED version: v0.19.4
28 Oct 17:28:40 - [info] Node.js version: v8.12.0
28 Oct 17:28:40 - [info] Windows NT 10.0.17134 x64 LE
[28 Oct 17:29:07 - [info]] Server now running at http://127.0.0.1:1880/
28 Oct 17:29:07 - [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.
-----
28 Oct 17:29:07 - [info] Starting flows
28 Oct 17:29:12 - [info] Started flows
```

A red box highlights the "Starting flows" and "Started flows" messages at the bottom of the log.

A red bracket on the right side of the window indicates that the window is not closed, with the text "画面閉じない" (Do not close the window) written below it.

(5) Node-RED IDEを起動

ブラウザを起動し、URLに“localhost:1880”と打ってみよう！

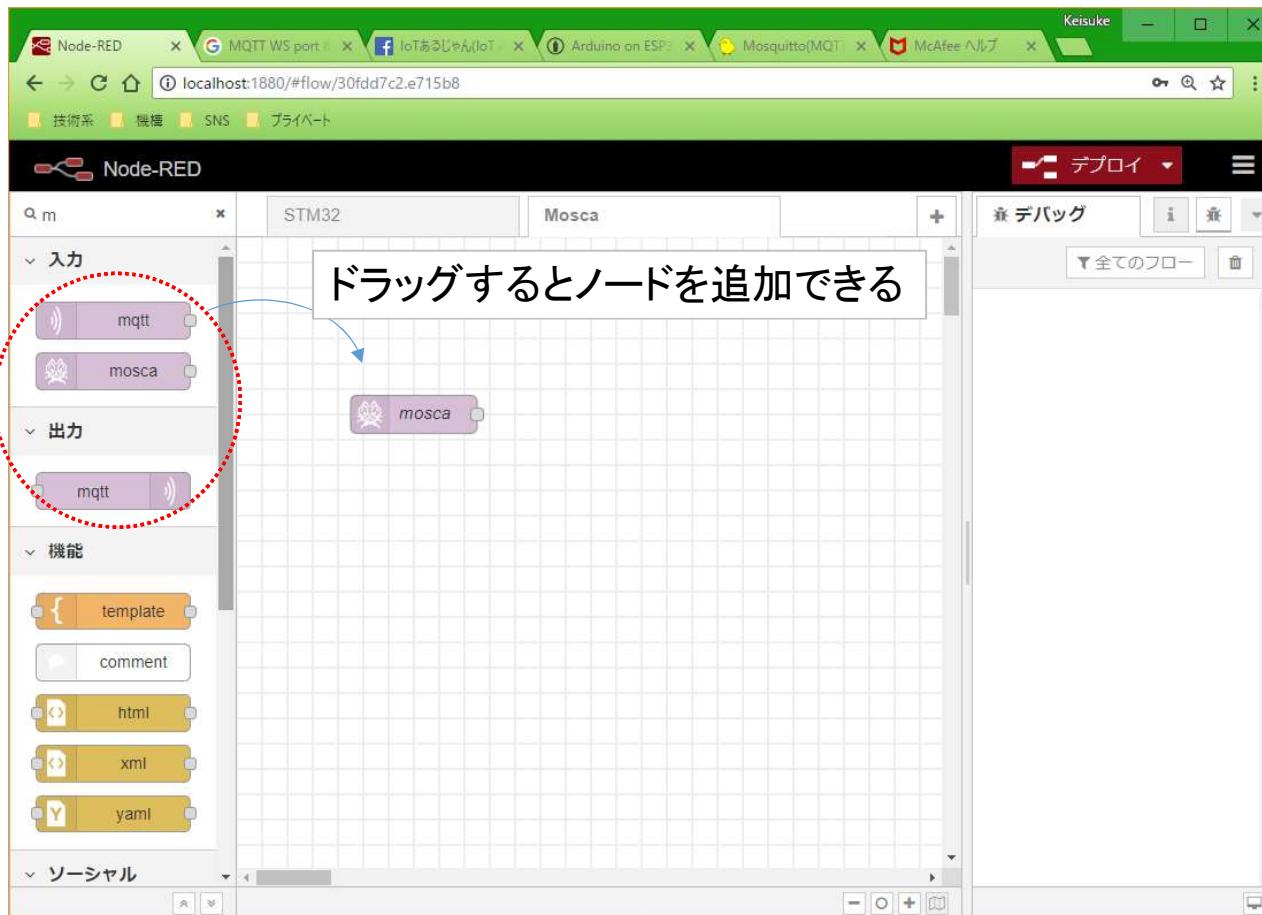


4-3 Node-RED側のMQTTインストール

- (1) メニューからパレットの管理を開く
- (2) ノード “node-red-contrib-mqtt-broker” を追加



(3) mqtt , mosca のノードが追加されたことを確認します。
フィルタで "m" を入力。"m" が含まれるノードのみが表示されます。



5 STM32からのデータをNode-REDで見る

PCにBrokerを立ち上げSTM32からデータが届いていることを確認します。

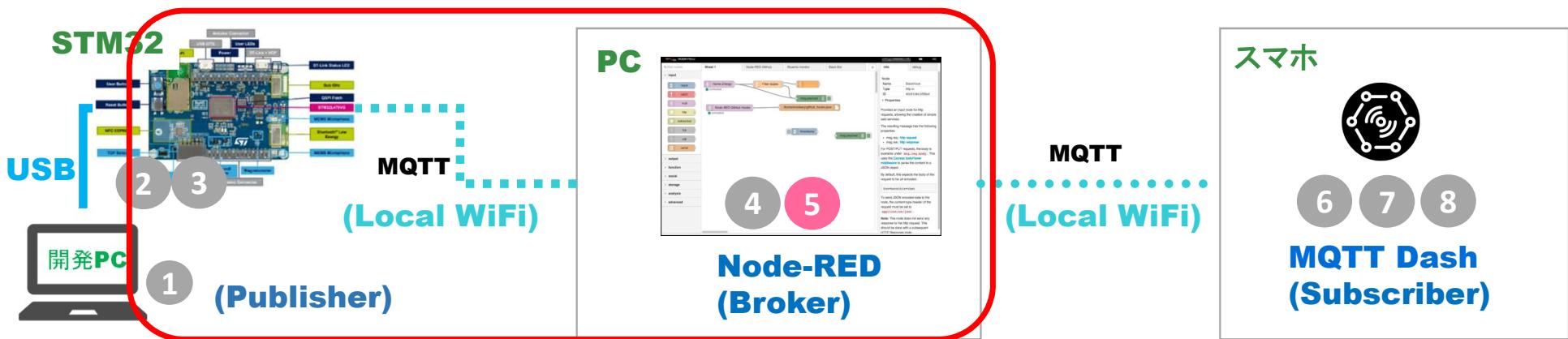
5-1 Node-REDにMQTTサーバーを立てる

5-2 STM32(Publisher)とNode-RED(Broker)を繋げる

5-3 Arduino IDE にWiFi ドライバーをインストール

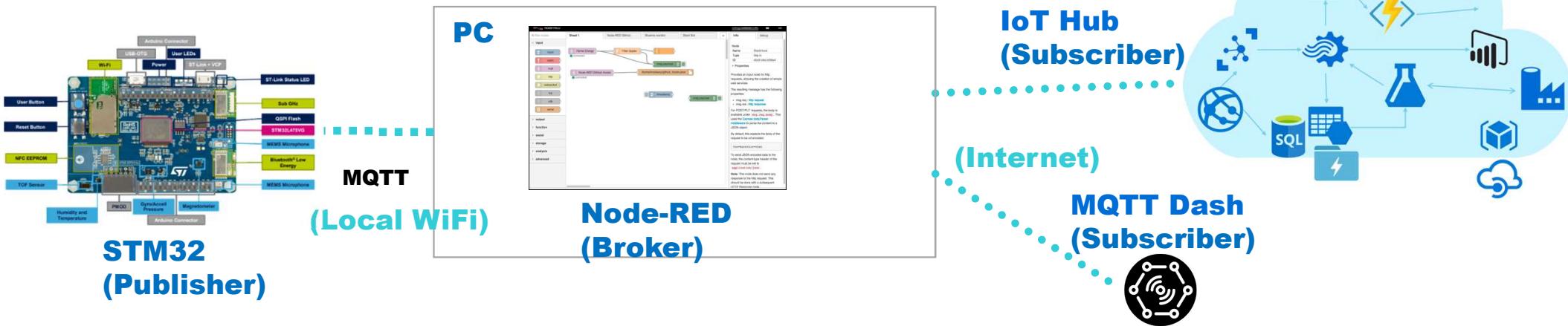
5-4 Arduino IDE にMQTTクライアントをインストール

5-5 STM32にサンプルプログラムを書き込む



5. STM32からのデータをNode-REDで見る

MQTTとは... IoTを意識した軽量なプロトコル



- Brokerは仲介役(一時的にデータを集める)
- PublisherはBrokerへひらすらデータを送る
- BrokerはPublisherから届いたデータを加工して新しいデータを用意することもできる
- Subscriberは一度Brokerへデータの送信を依頼するとその後Publisherからのデータが送られてくる

5-0 Node-RED 上にMQTTノードを配置

MQTTノードを作成する方法は以下の3通りです。

- ・手動で追加

手動でノードを追加します。

- ・クリップボードからのインポート

あらかじめ作成されたノードの構成をインポートします。

ホームページ等で検索するとこの形で配信されている場合が多いので、こちらの方法もみておきましょう。

- ・ライブラリからのインポート

自分で作成したノードの構成をライブラリとして保存することができます。

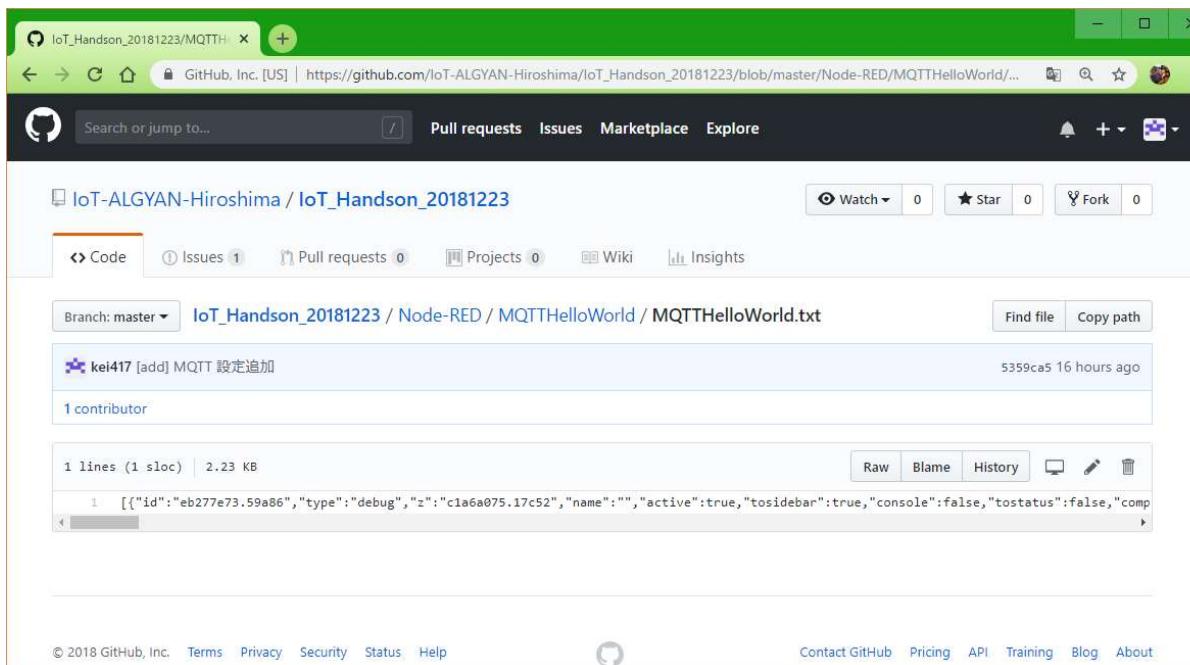
この章ではクリップボードからのインポートの方法を説明します。

(1) ノード構成のページにアクセス

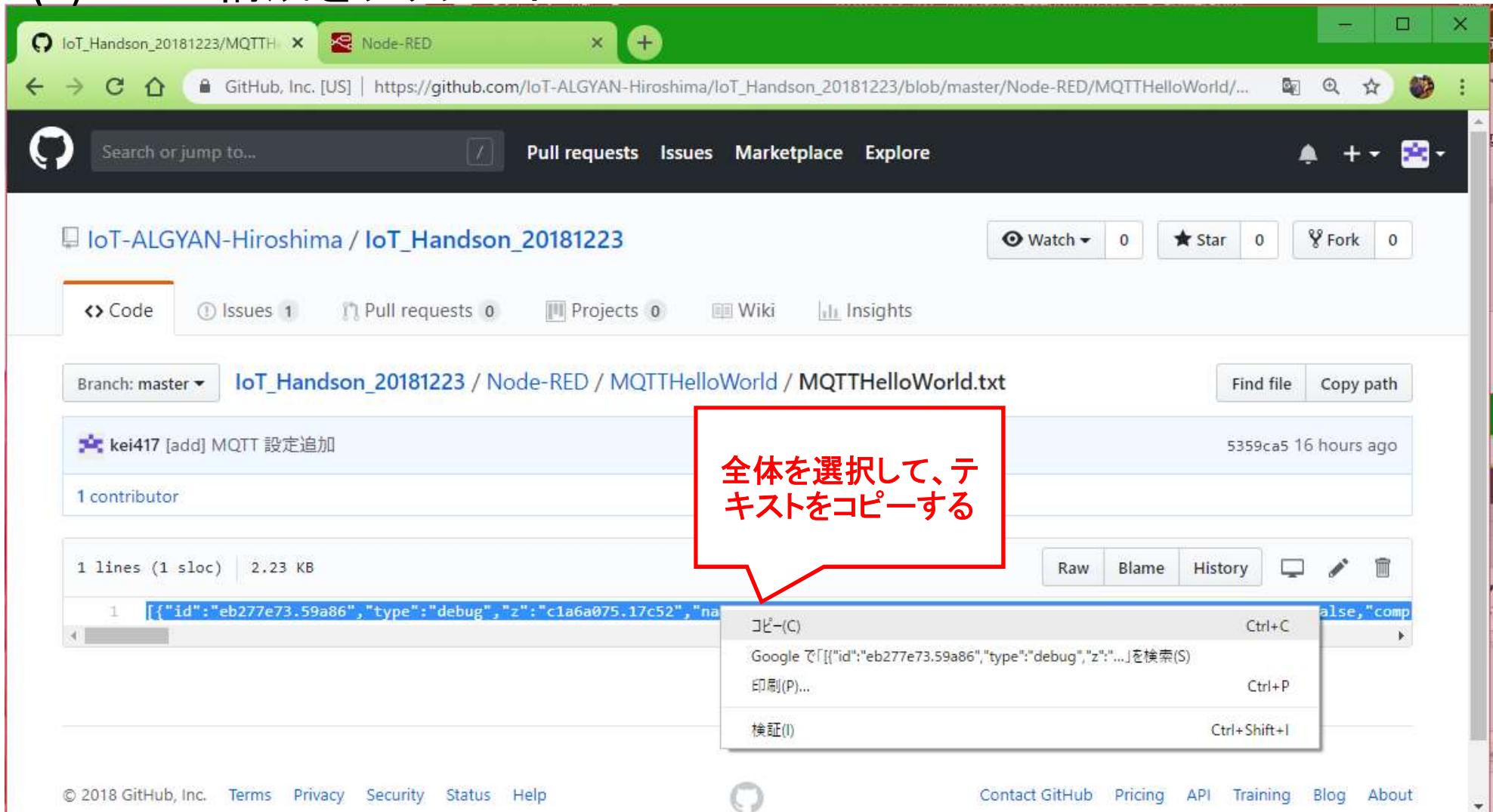
GitHubにNode-REDの構成を用意しました。

以下にアクセスしてください。

https://github.com/IoT-ALGYAN-Hiroshima/IoT_Handson_20181223/blob/master/Node-RED/MQTTHelloWorld/MQTTHelloWorld.txt



(2) ノード構成をクリップボードにコピー



IoT_Handson_20181223 / IoT_Handson_20181223

Code Issues 1 Pull requests 0 Projects 0 Wiki Insights

Branch: master IoT_Handson_20181223 / Node-RED / MQTTHelloWorld / MQTTHelloWorld.txt

kei417 [add] MQTT 設定追加 5359ca5 16 hours ago

1 contributor

1 lines (1 sloc) 2.23 KB

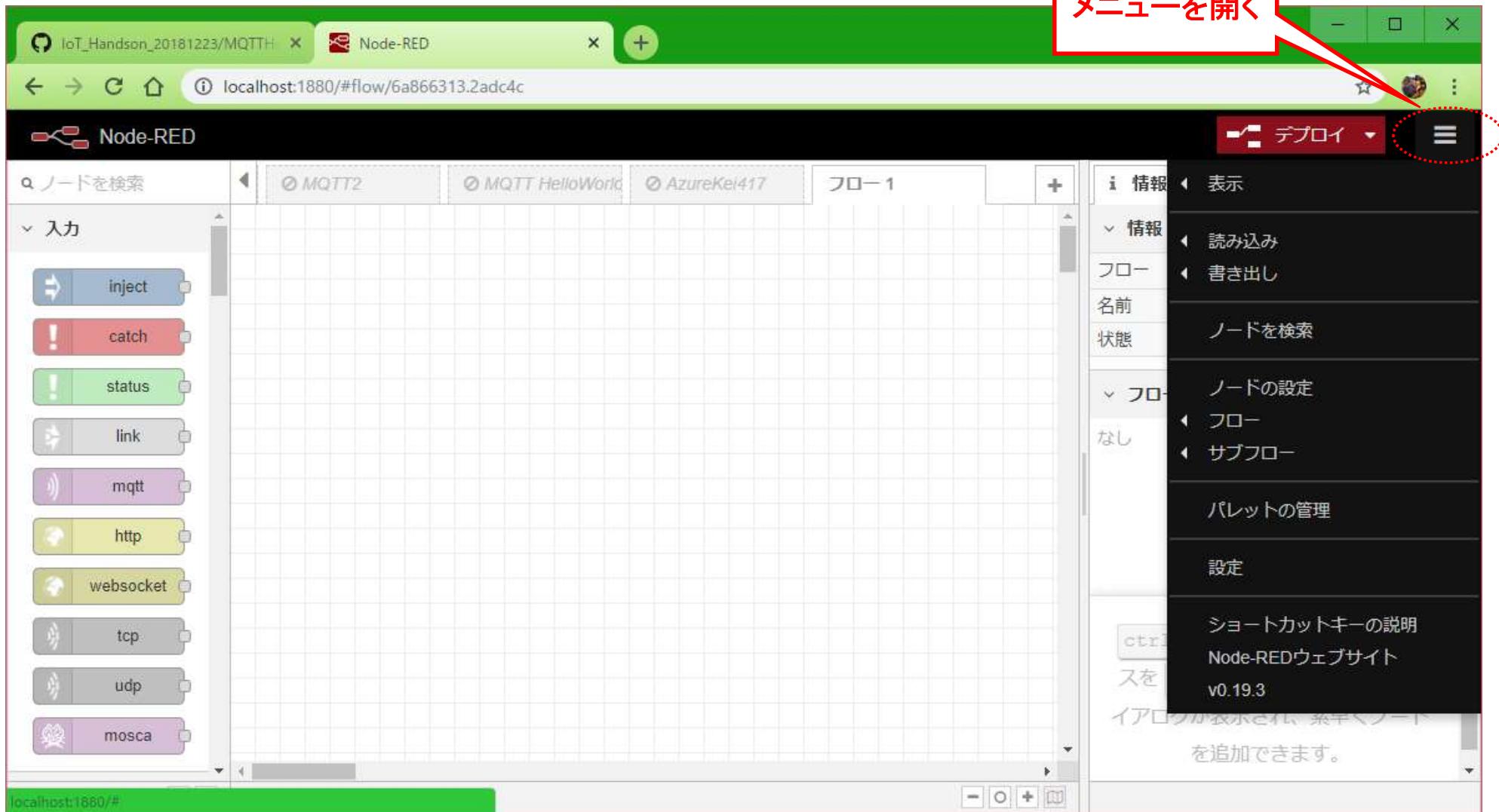
```
[{"id": "eb277e73.59a86", "type": "debug", "z": "c1a6a075.17c52", "na..."}]
```

全体を選択して、テキストをコピーする

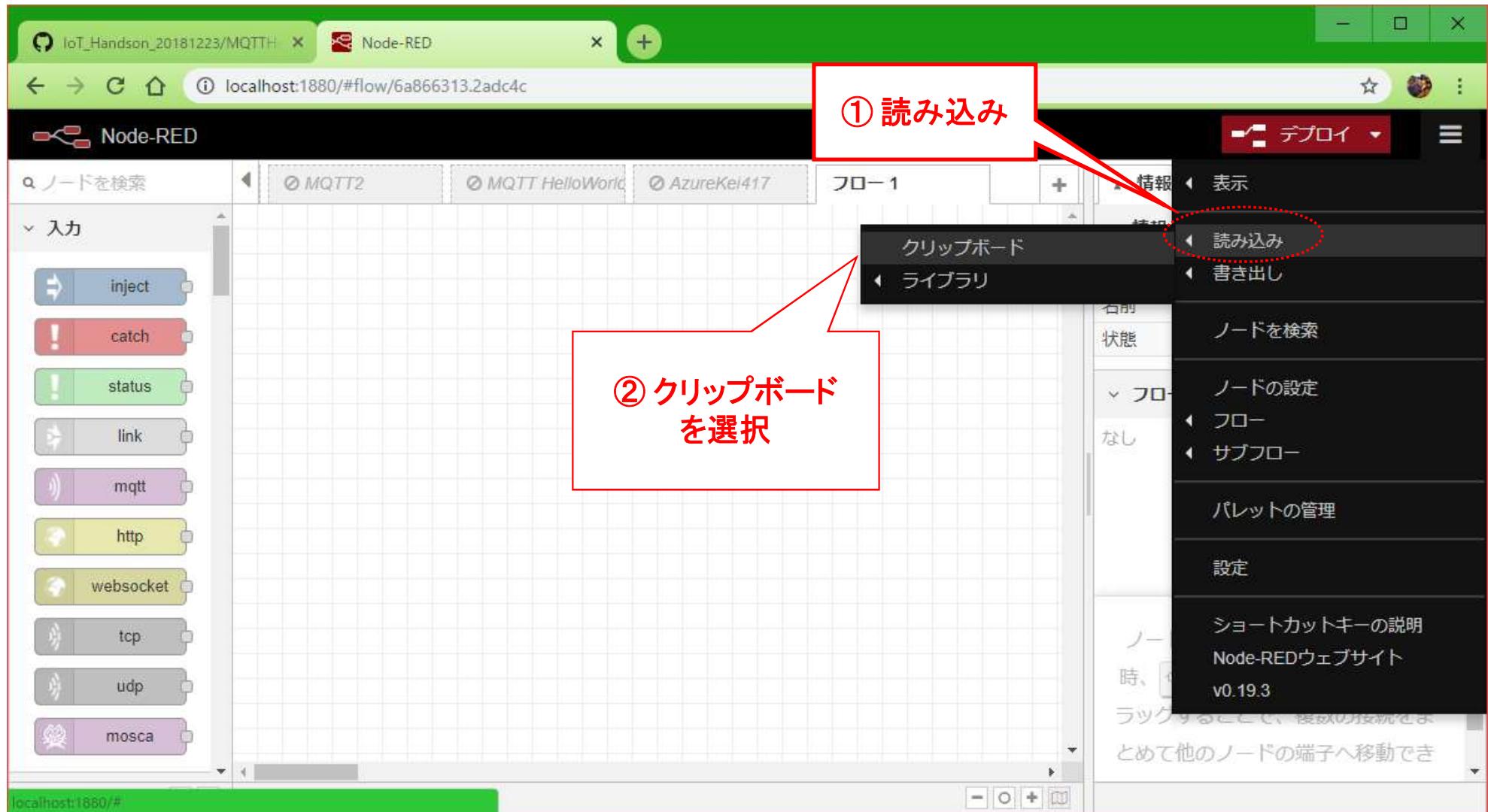
Raw Blame History コピー(C) Ctrl+C Googleで「[{"id": "eb277e73.59a86", "type": "debug", "z": "..."}」を検索(S) 印刷(P)... Ctrl+P 検証(I) Ctrl+Shift+I

© 2018 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About

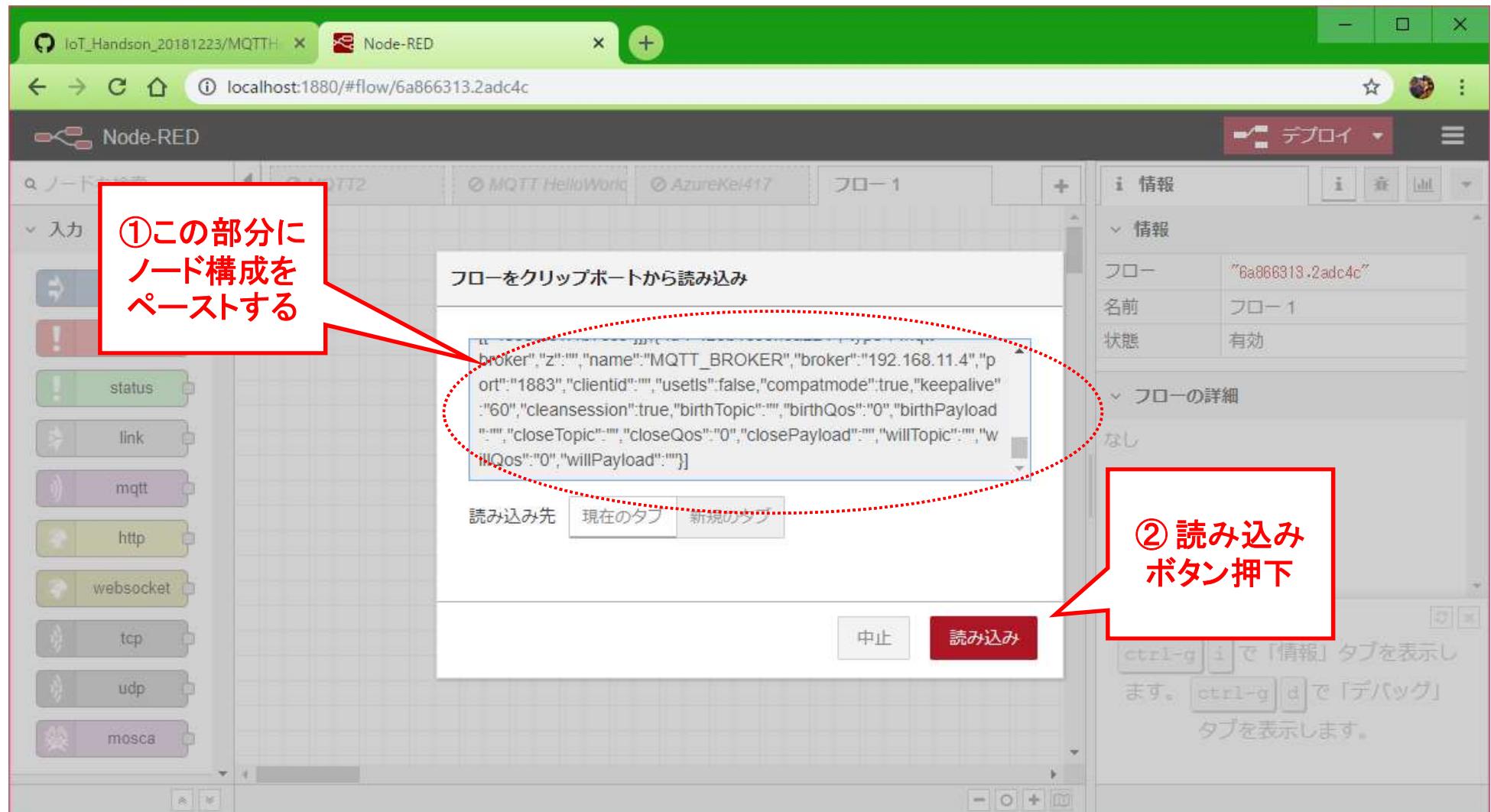
(3) Node-REDのメニューを開く



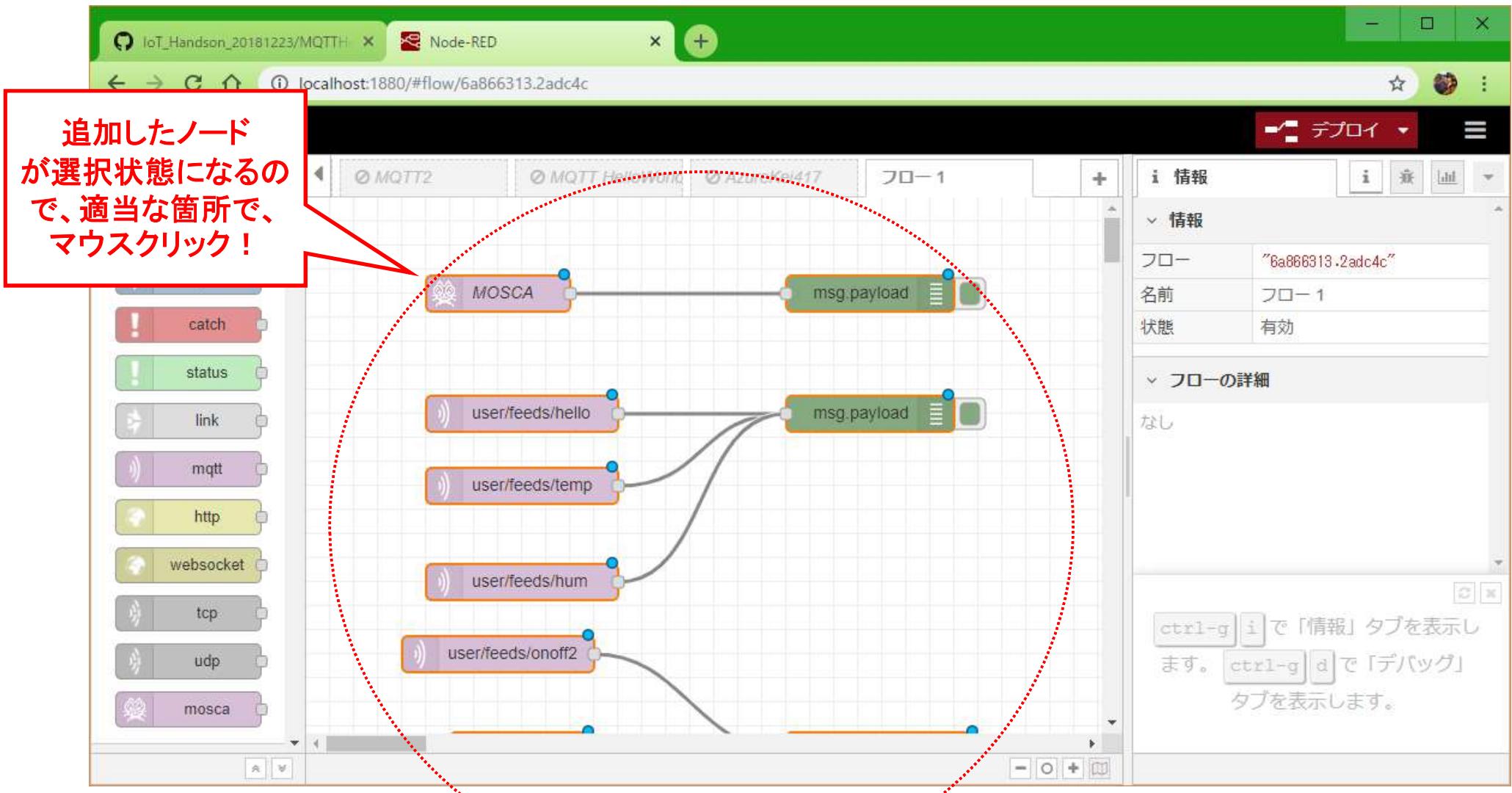
(3)クリップボードからの読み込み



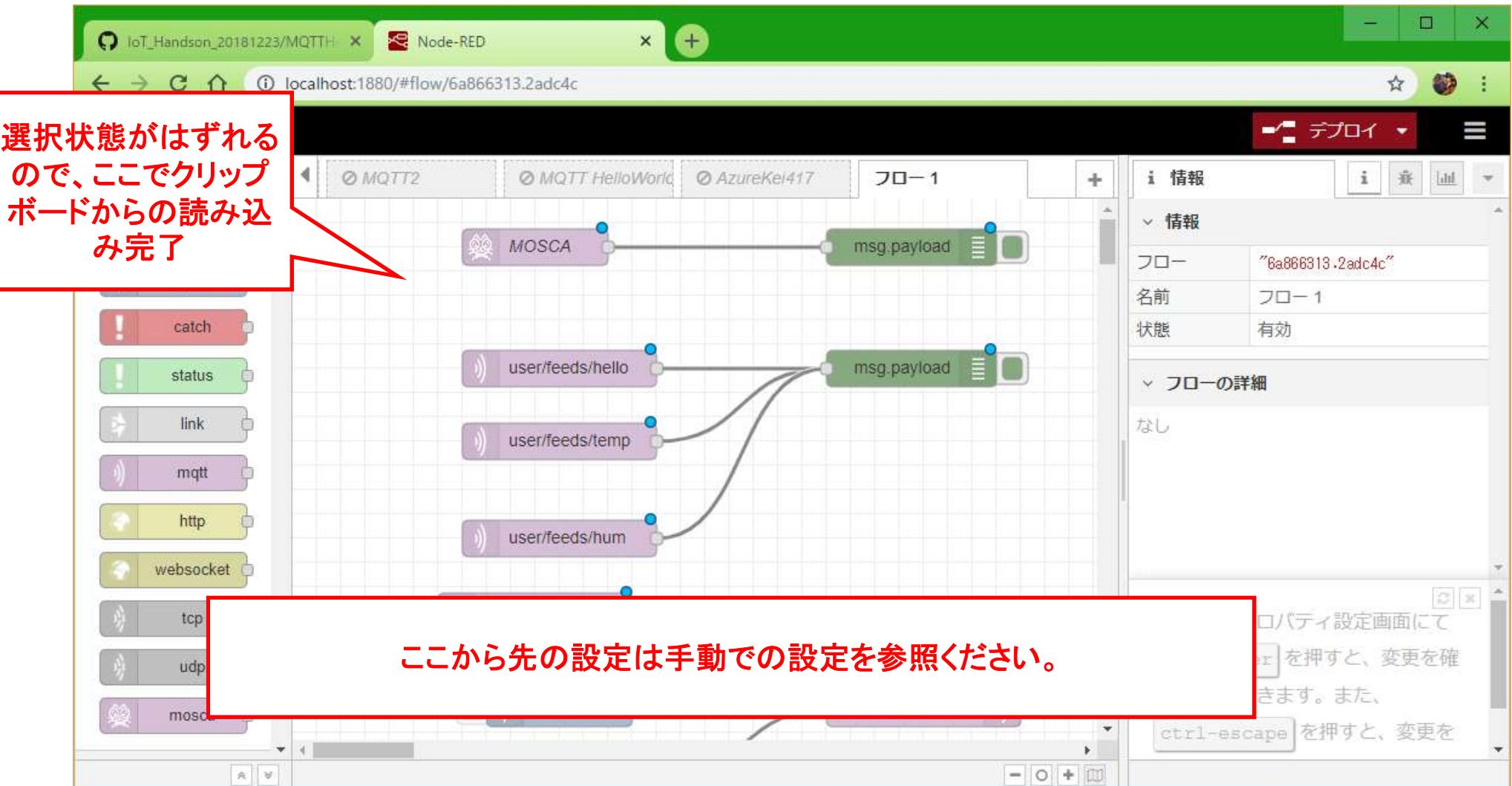
(4)クリップボードからの読み込み



(4)クリップボードからの読み込み



(5) クリップボードからの読み込み完了

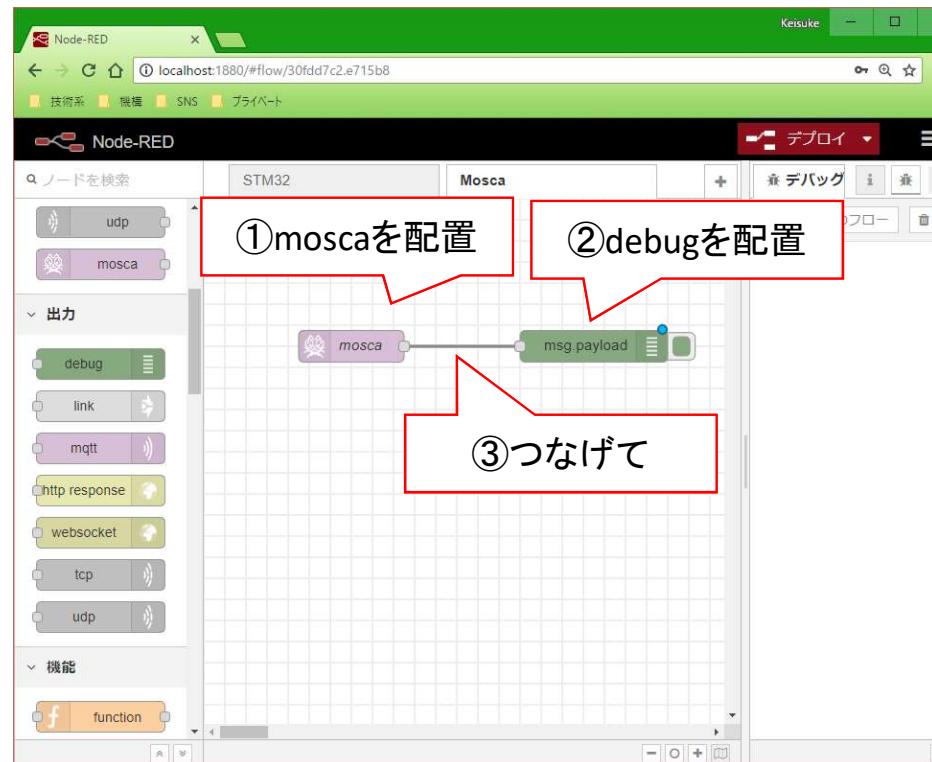


5-1 Node-REDにMQTTサーバーを立てる

(1) Moscaノードを配置

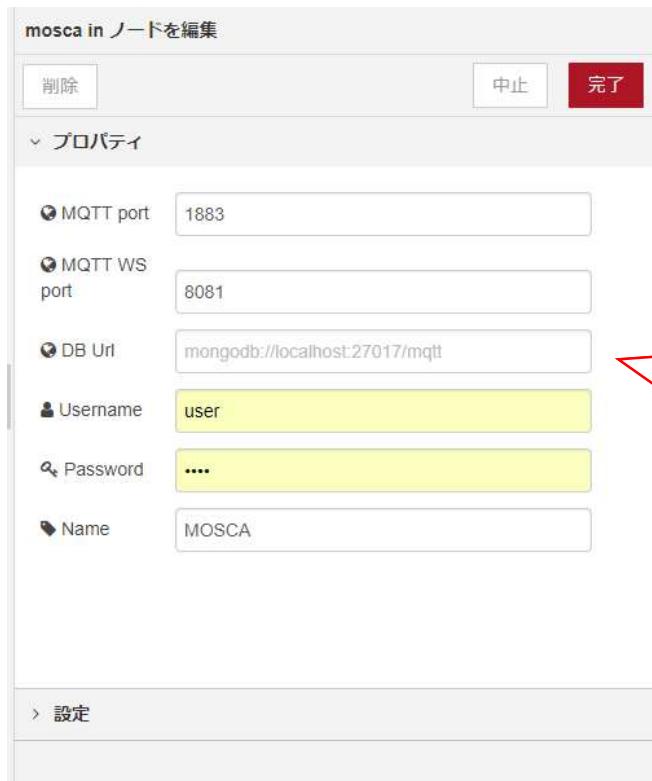
MQTTサーバー本体を配置します。

debugノードにつなげることで、接続時のログを記録することができます。



(2) Moscaノードの設定を変更

MQTTサーバーのポート番号、MQTTサーバーにアクセスできるユーザーアカウントを設定します。編集するには Mosca ノードをダブルダブルクリックします



② 完了

① Moscaノードの設定を変更
MQTT port: 1883 (デフォルト)
MQTT WS port: 任意 (今回未使用)
Username:user
Password:pass
Name:MOSCA

```
(2) 東西中密ナードー / node-red
-----
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.

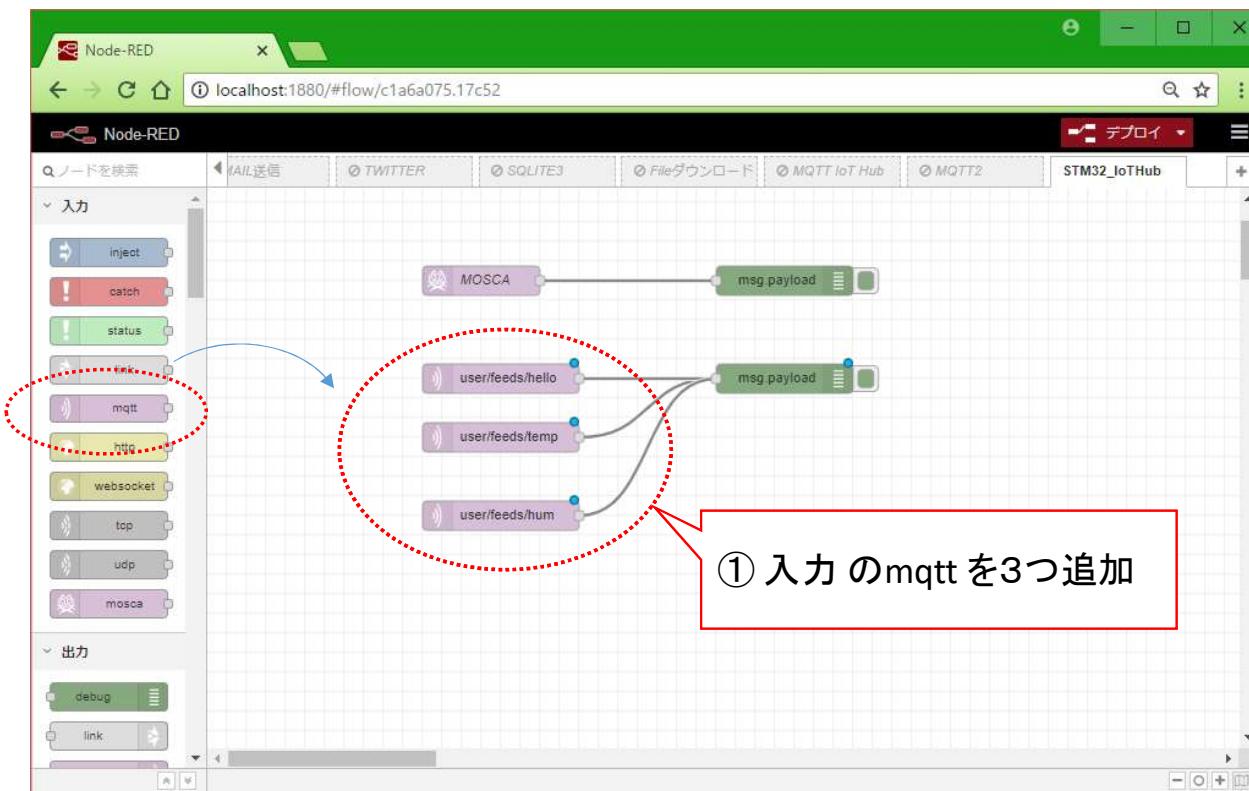
You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.

3 Nov 16:07:30 - [info] Starting flows
3 Nov 16:07:30 - [info] Started flows
3 Nov 16:07:40 - [info] Stopping flows
3 Nov 16:07:40 - [info] Stopped flows
3 Nov 16:07:40 - [info] Starting flows
3 Nov 16:07:40 - [info] [mosca in:MOSCA] Binding mosca mqtt server on port: 1883
3 Nov 16:07:40 - [info] Started flows
```

5-2 STM32(Publisher)とNode-RED(Broker)を繋げる

(1) mqtt 購読設定を3つ追加 (詳細は次ページ)

STM32が Publish してくるメッセージを Node-RED側で購読(入力)できるようにします。

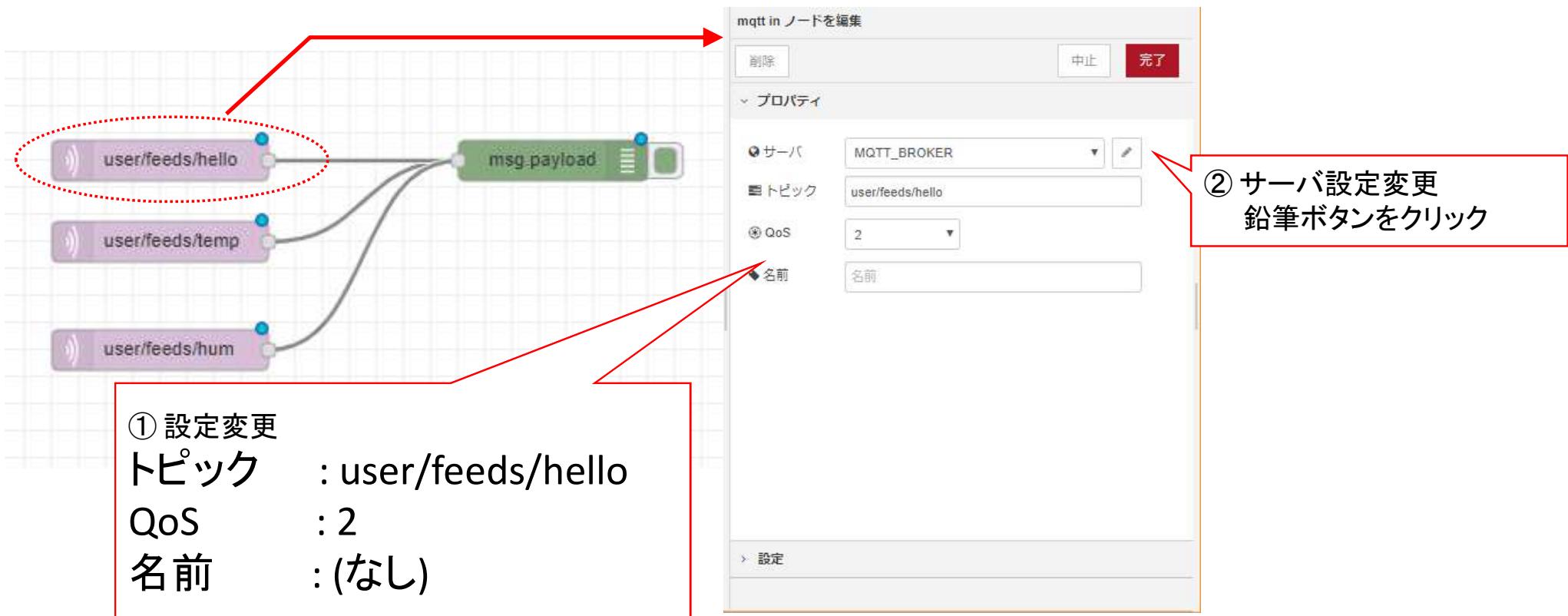


(2) mqtt 購読設定 (1/3)

STM32がMQTTサーバーと接続直後のメッセージとなります。

トピック名がMQTT通信のメッセージ名となります。

STM32側もこのトピック名でメッセージを送ります。



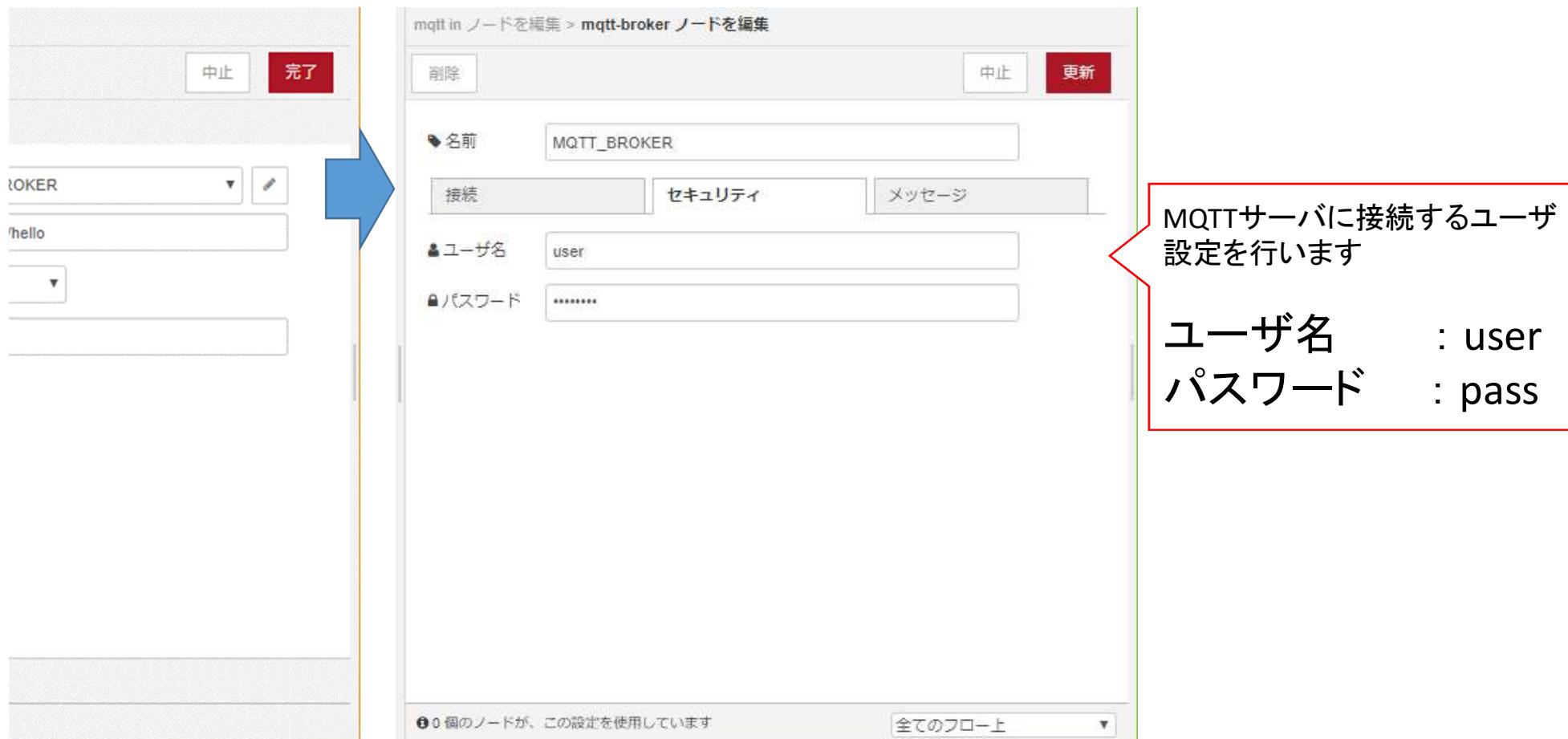
(4) MQTTサーバの設定 その1

STM32がMQTTサーバーと接続直後のメッセージとなります。



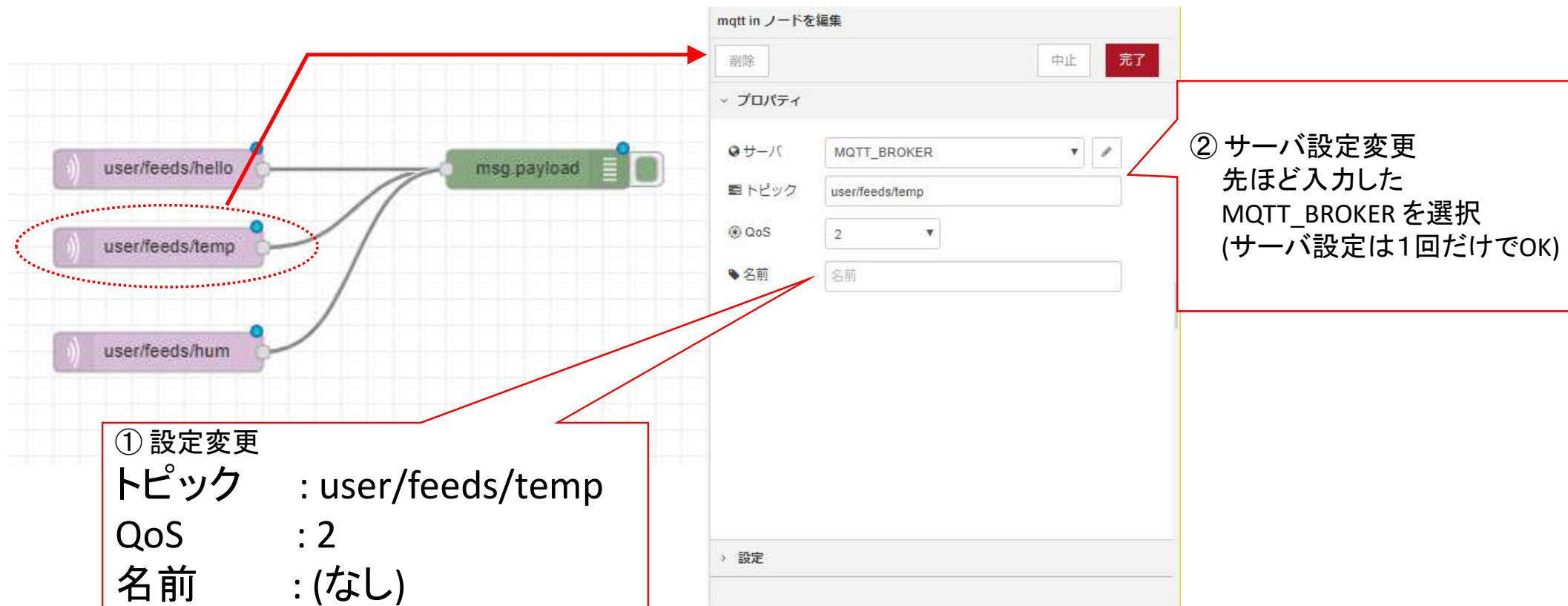
(5) MQTTサーバの設定 その2

STM32がMQTTサーバーと接続直後のメッセージとなります。



(6) mqtt 購読設定 (1/2)

STM32の温度・湿度センサの 温度 の値 (user/feeds/temp) を購読します。

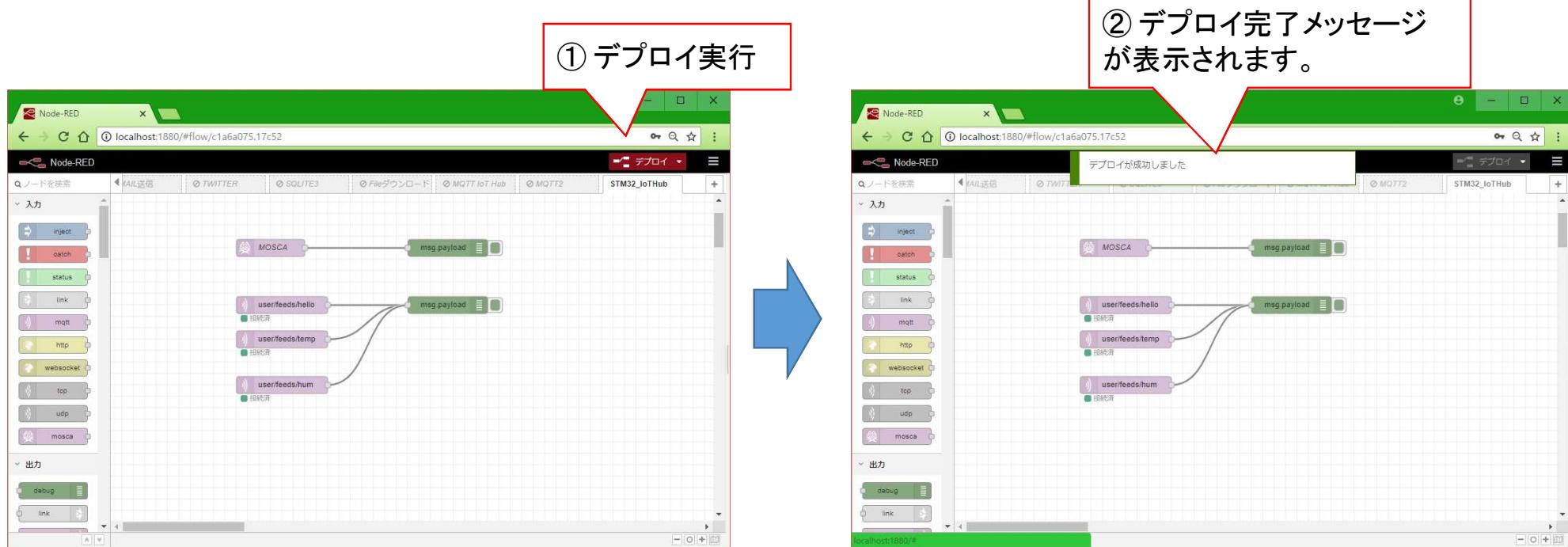


(7) mqtt 購読設定 (2/2)

STM32の温度・湿度センサの 湿度 の値 (user/feeds/hum) を購読します。
3つの購読設定をデバッグノードにつなげます。

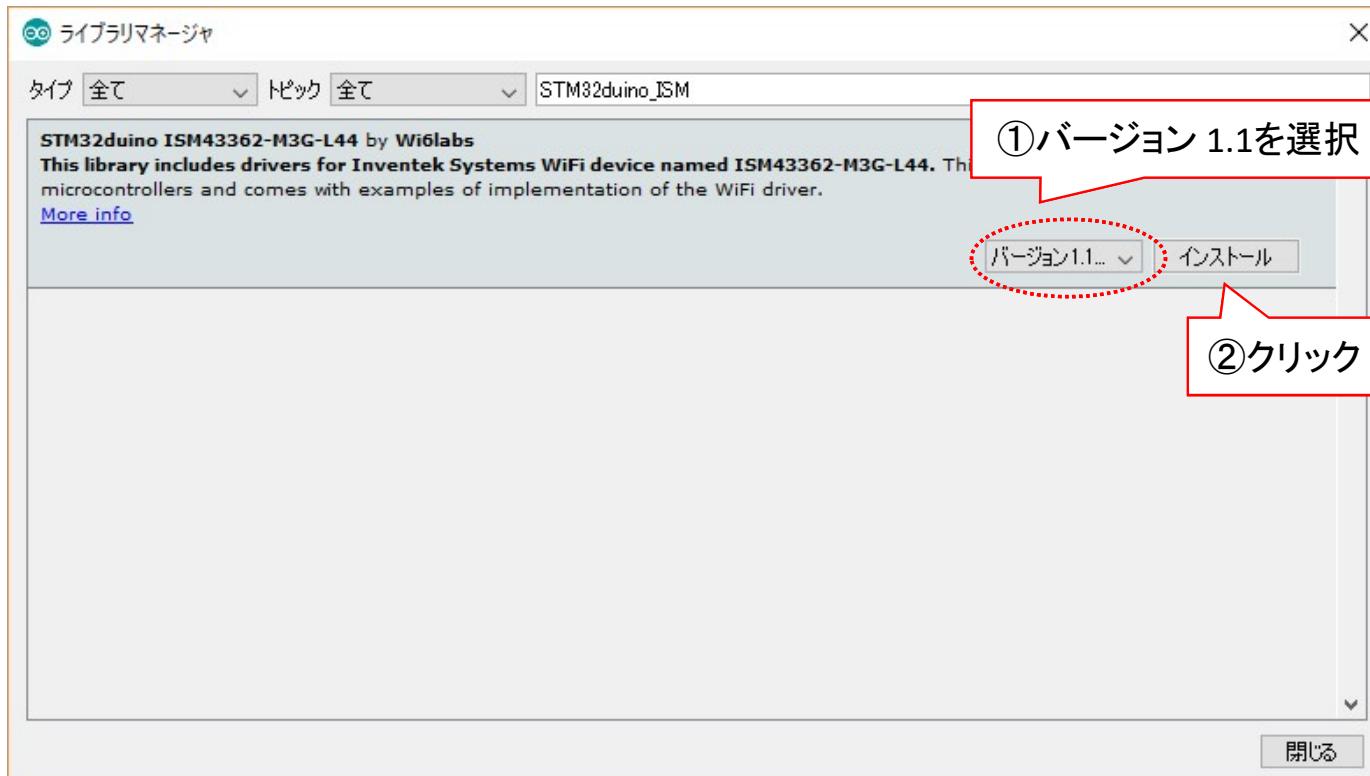


(8) 変更した設定をデプロイします



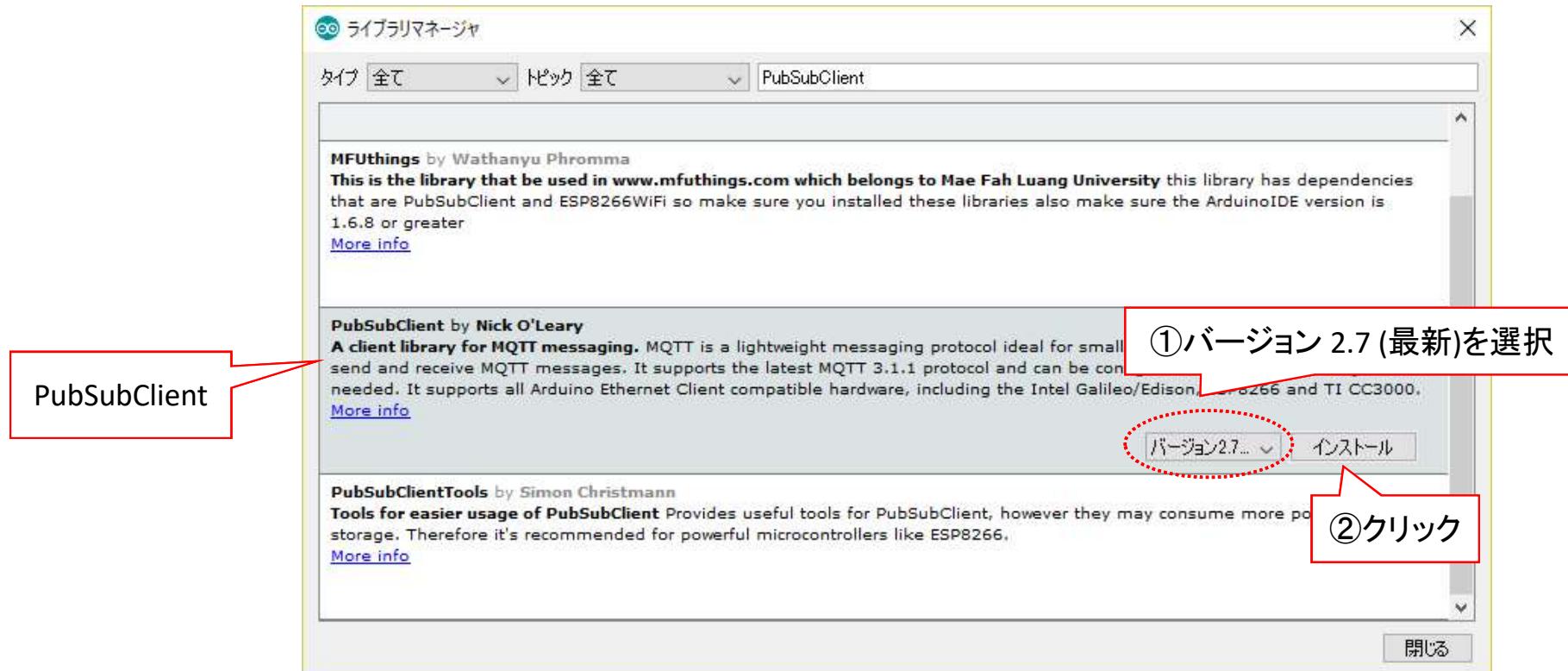
5-3 Arduino IDE に WiFi ドライバーをインストール

- (1) ライブラリマネージャで **STM32duino_ISM43362-M3G-L44** をインストール
STM32 評価ボード用の WiFi ドライバーの最新版をインストールします。
旧バージョンが入っている場合は最新版にアップデートしてください。



5-4 Arduino IDE にMQTTクライアントをインストール

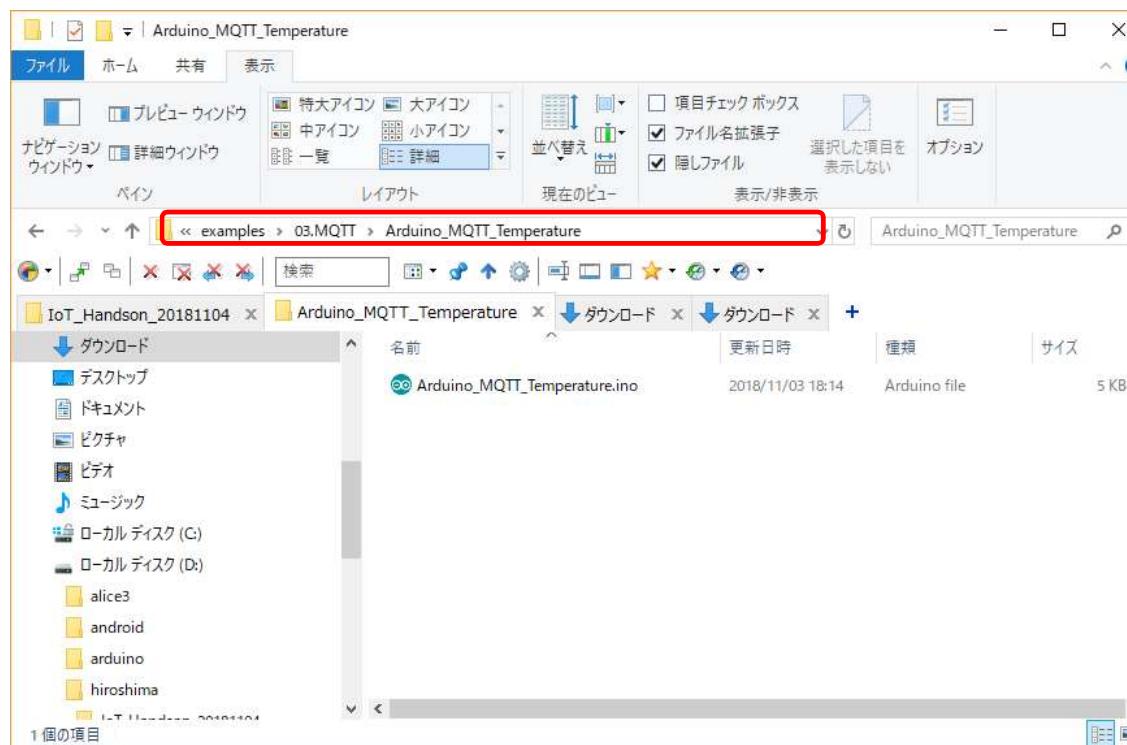
- (1) ライブラリマネージャで **PubSubClient** をインストール
汎用のMQTTクライアントライブラリです。



5-5 STM32にサンプルプログラムを書き込む

■サンプルコードを開く

「2. STM32のLEDを点滅させる」の「2-1 サンプルコードをダウンロードする」でダウンロードして解凍したフォルダから **03.MQTT/Arduino_MQTT_Temperature**を開きます。



(3) Wifi と MQTT サーバの設定を変更

サンプルコード(Arduino_MQTT_Temperature.ino) の以下の赤字の部分を自分の環境に合わせて変更します。

```
// WiFi 設定 (SSID)  
char* ssid      = "uffalo-G-AE22";  
// WiFi 設定 (パスワード)  
const char* password  = "12345678";  
  
// MQTT サーバ (IPアドレス)  
#define AIO_SERVER    "<Your MQTT Server IPアドレス>"  
// MQTT サーバ (ポート番号)  
#define AIO_SERVERPORT 1883  
// MQTT サーバ (ユーザーID)  
#define AIO_USERNAME   "user"  
// MQTT サーバ (パスワード)  
#define AIO_KEY        "pass"
```

今回はこのまま

今回はこのまま

次ページ参照

ノスです。

```
Microsoft Windows [Version 10.0.17134.345]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>ipconfig

Windows IP 構成

イーサネット アダプター イーサネット:
  メディアの状態.: . . . . : メディアは接続されていません
  接続固有の DNS サフィックス . . . . :

Wireless LAN adapter ローカル エリア接続* 1:
  メディアの状態.: . . . . : メディアは接続されていません
  接続固有の DNS サフィックス . . . . :

Wireless LAN adapter ローカル エリア接続* 12:
  メディアの状態.: . . . . : メディアは接続されていません
  接続固有の DNS サフィックス . . . . :

Wireless LAN adapter Wi-Fi:
  接続固有の DNS サフィックス . . . . :
    IPv6 アドレス . . . . . : fe80::f141:6134%10
    IPv6 アドレス . . . . . : fe80::f141:6134%10
    IPv6 アドレス . . . . . : fe80::f141:6134%10
    一時 IPv6 アドレス . . . . . : fe80::f141:6134%10
    リンクローカル IPv6 アドレス . . . . . : ::192.168.100.103
    IPv4 アドレス . . . . . : 192.168.100.103
    ネットマスク . . . . . : 255.255.255.0
    デフォルト ゲートウェイ . . . . . : fe80::3a37:8ff%10
                                         192.168.100.1

イーサネット アダプター Bluetooth ネットワーク接続:
  メディアの状態.: . . . . : メディアは接続されていません
  接続固有の DNS サフィックス . . . . :

C:\WINDOWS\system32>
```

(5) STM32へサンプルプログラムの書き込み

(6) USBを経由して送られてくるデータを確認

WiFiに接続しています。
接続完了時に、

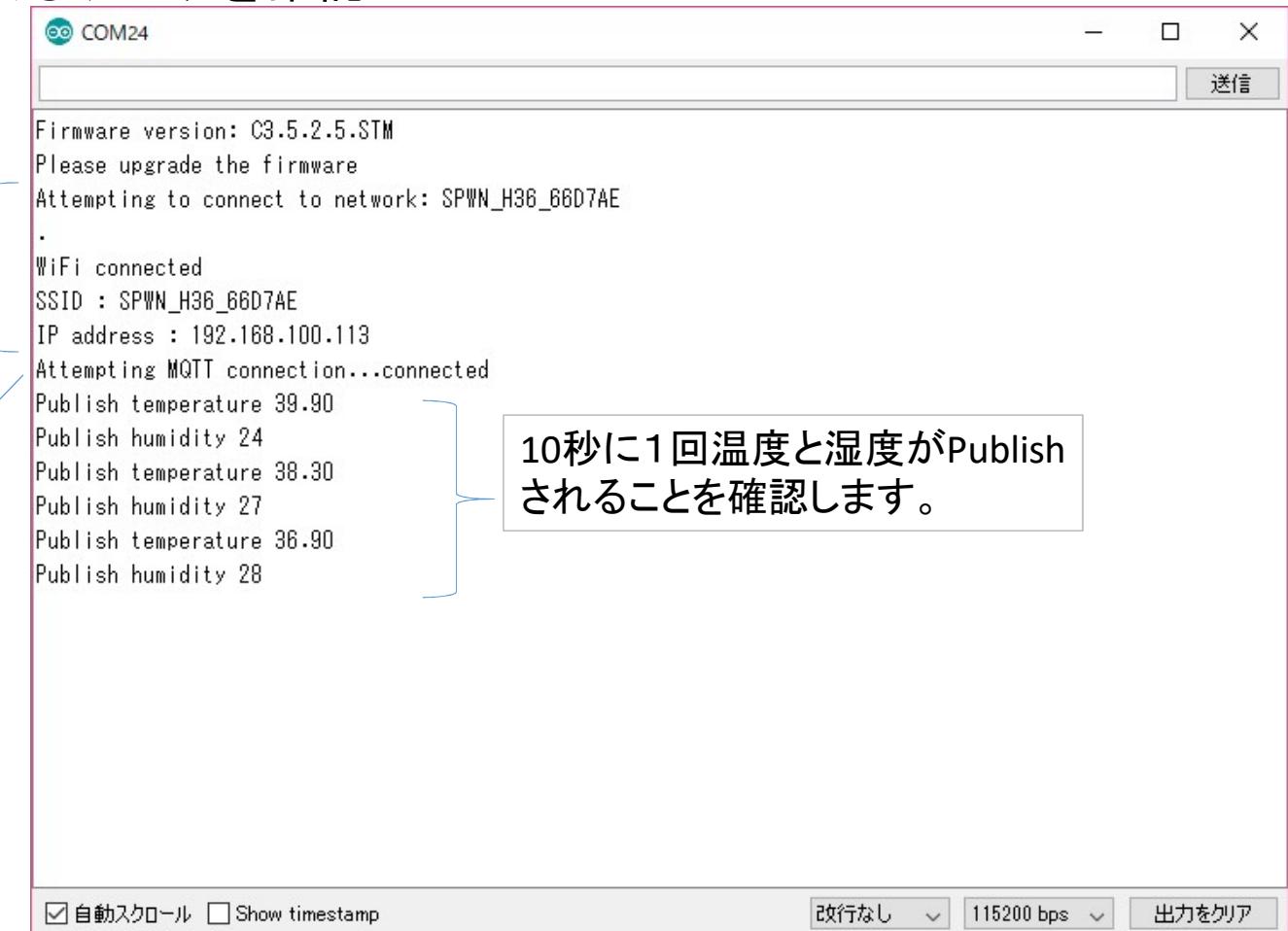
“WiFi connected”

と出力された後にSSIDと現在の
IPアドレスが出力されます。

MQTTサーバに接続しています。
接続完了時に、

“connected”

と出力されます。

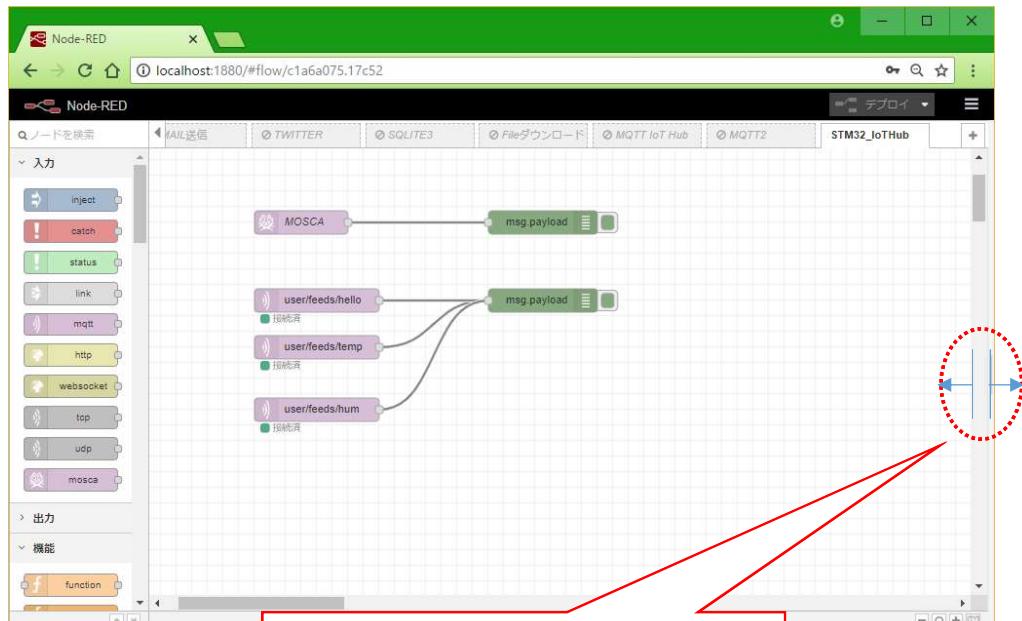


```
COM24
Firmware version: C3.5.2.5.STM
Please upgrade the firmware
Attempting to connect to network: SPWN_H36_66D7AE
.
WiFi connected
SSID : SPWN_H36_66D7AE
IP address : 192.168.100.113
Attempting MQTT connection...connected
Publish temperature 39.90
Publish humidity 24
Publish temperature 38.30
Publish humidity 27
Publish temperature 36.90
Publish humidity 28
```

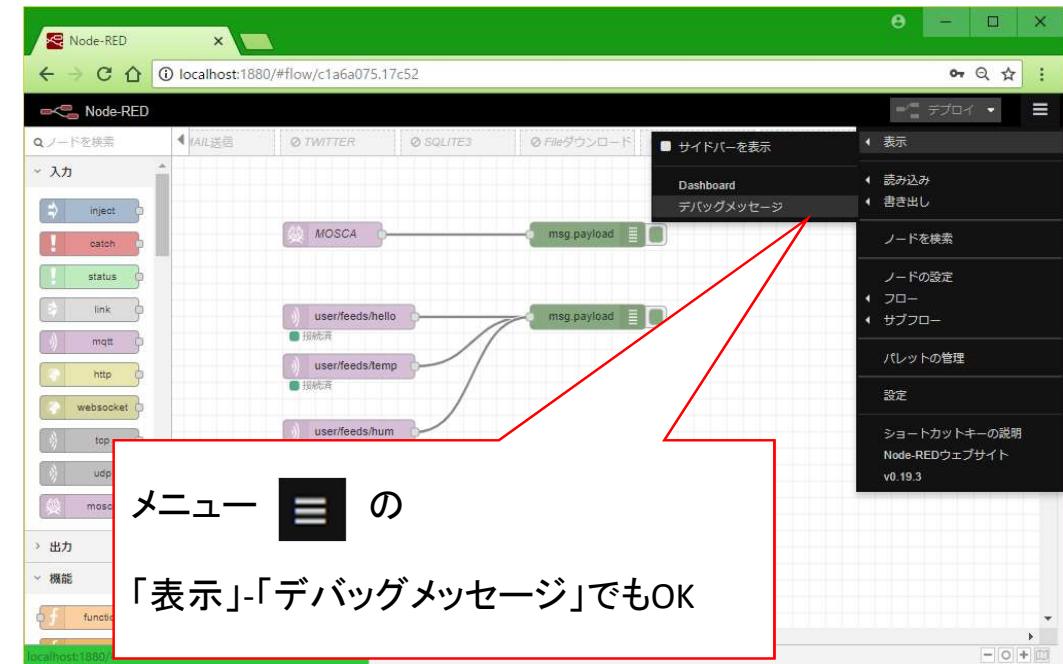
10秒に1回温度と湿度がPublishされることを確認します。

自動スクロール Show timestamp 改行なし 115200 bps 出力をクリア

(7) MQTTサーバの確認(その1) デバッグの表示を行います。



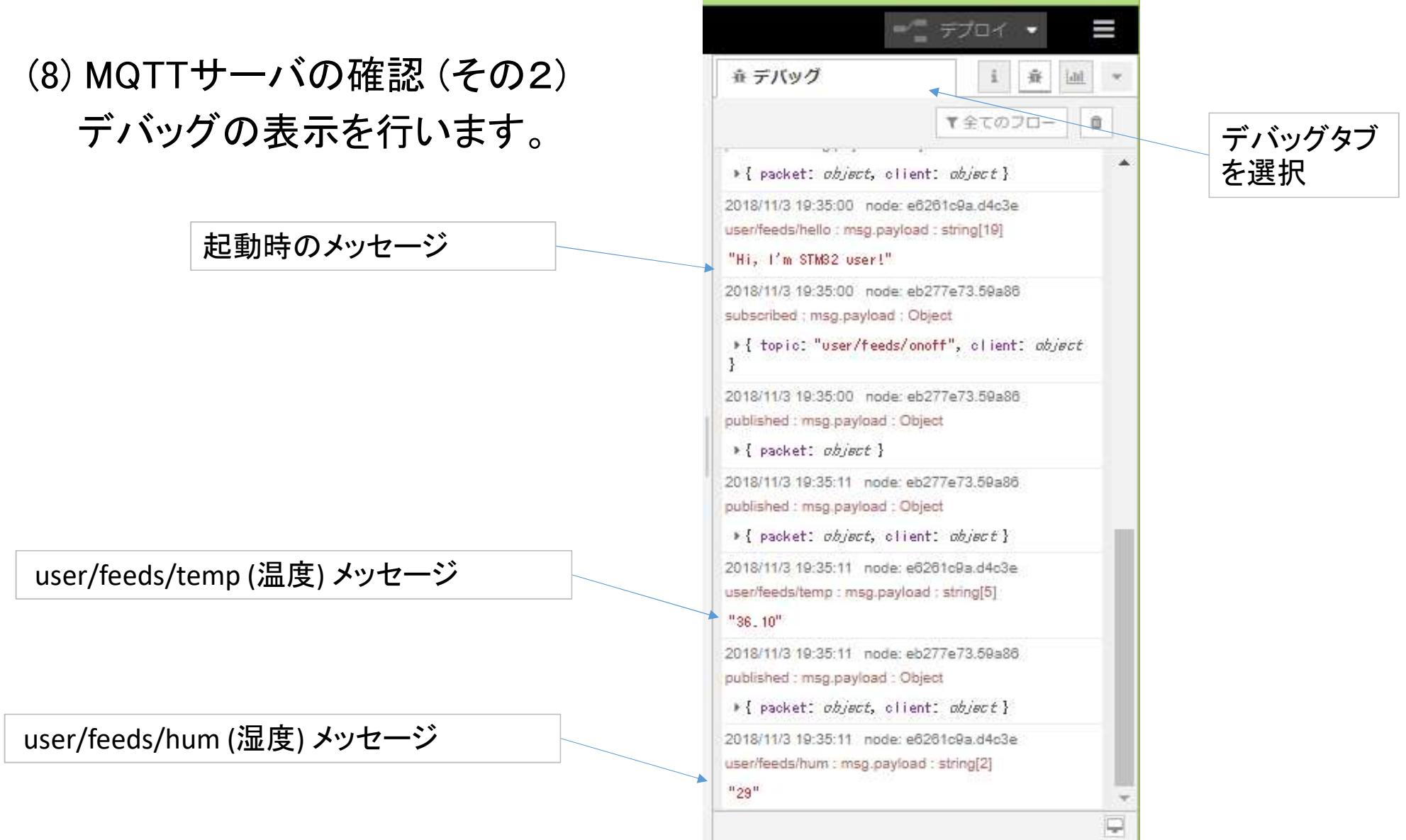
ツールバーの横中央あたりに
マウスカーソルを合わせて、
図の状態でドラッグすると情報
ウィンドウが表示されます。



メニュー の

「表示」-「デバッグメッセージ」でもOK

(8) MQTTサーバの確認 (その2) デバッグの表示を行います。



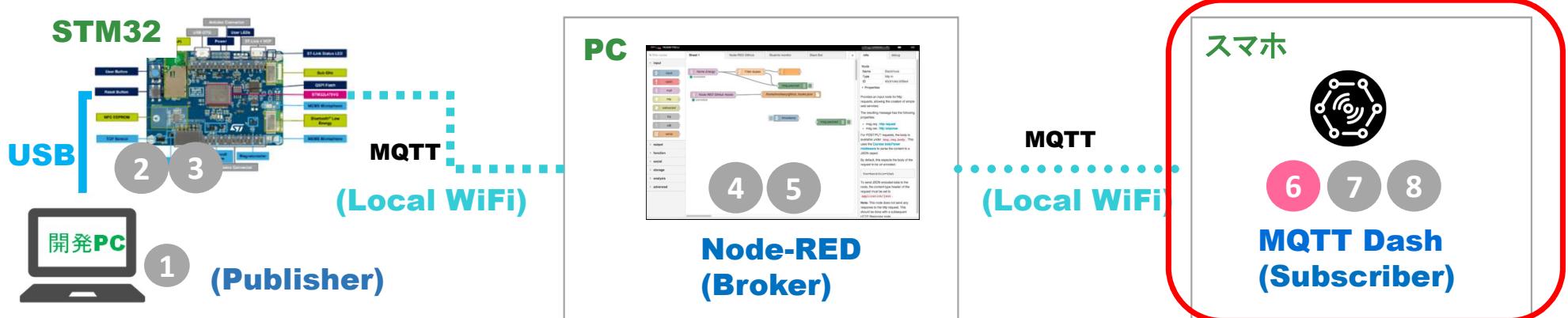
6 アンドロイドスマホにMQTT Dashをインストールする

スマホに Publisherとなるアプリをインストールし設定します。

6-1 MQTT Dashのインストール

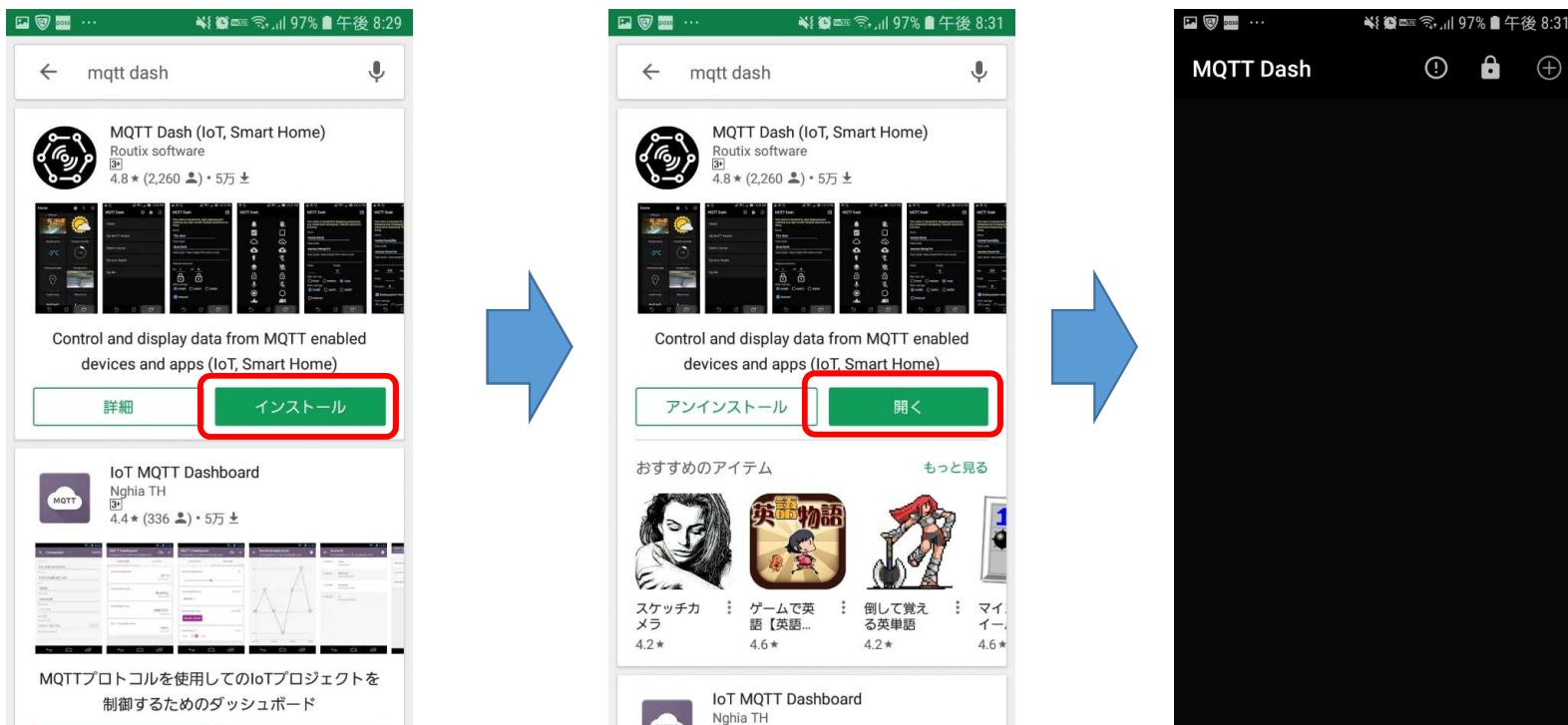
6-2 MQTT dash の設定

今回はAndroid スマホのアプリでNode-REDからのデータを確認します。
iPhoneの場合は別のアプリを試してください。



6-1 MQTT Dashのインストール

- (1) Google Play にて「mqtt dash」を検索
- (2) 「mqtt dash」をインストール



6-2 MQTT dash の設定

(1) MQTT サーバの設定を行います

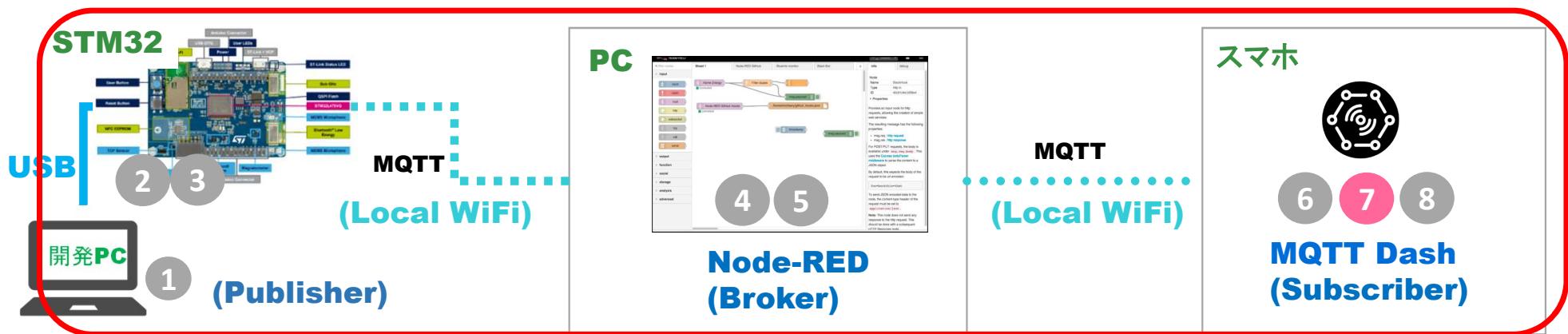


7 Node-REDに集めたデータをMQTT Dashで見る

STM32 からスマホにデータが届いていることを確認します。

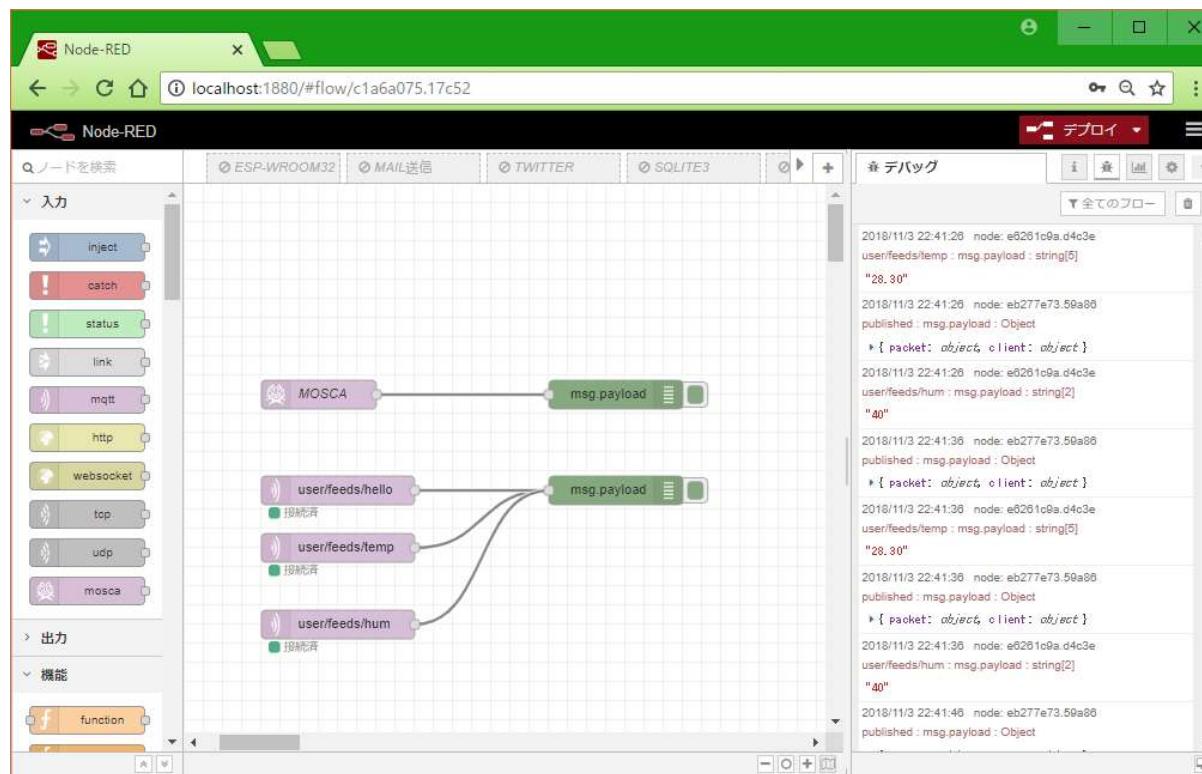
7-1 STM32(Publisher) とMQTT dash(Subscriber)を繋げる

7-2 Node-RED(Broker) とMQTT dash(Subscriber)を繋げる



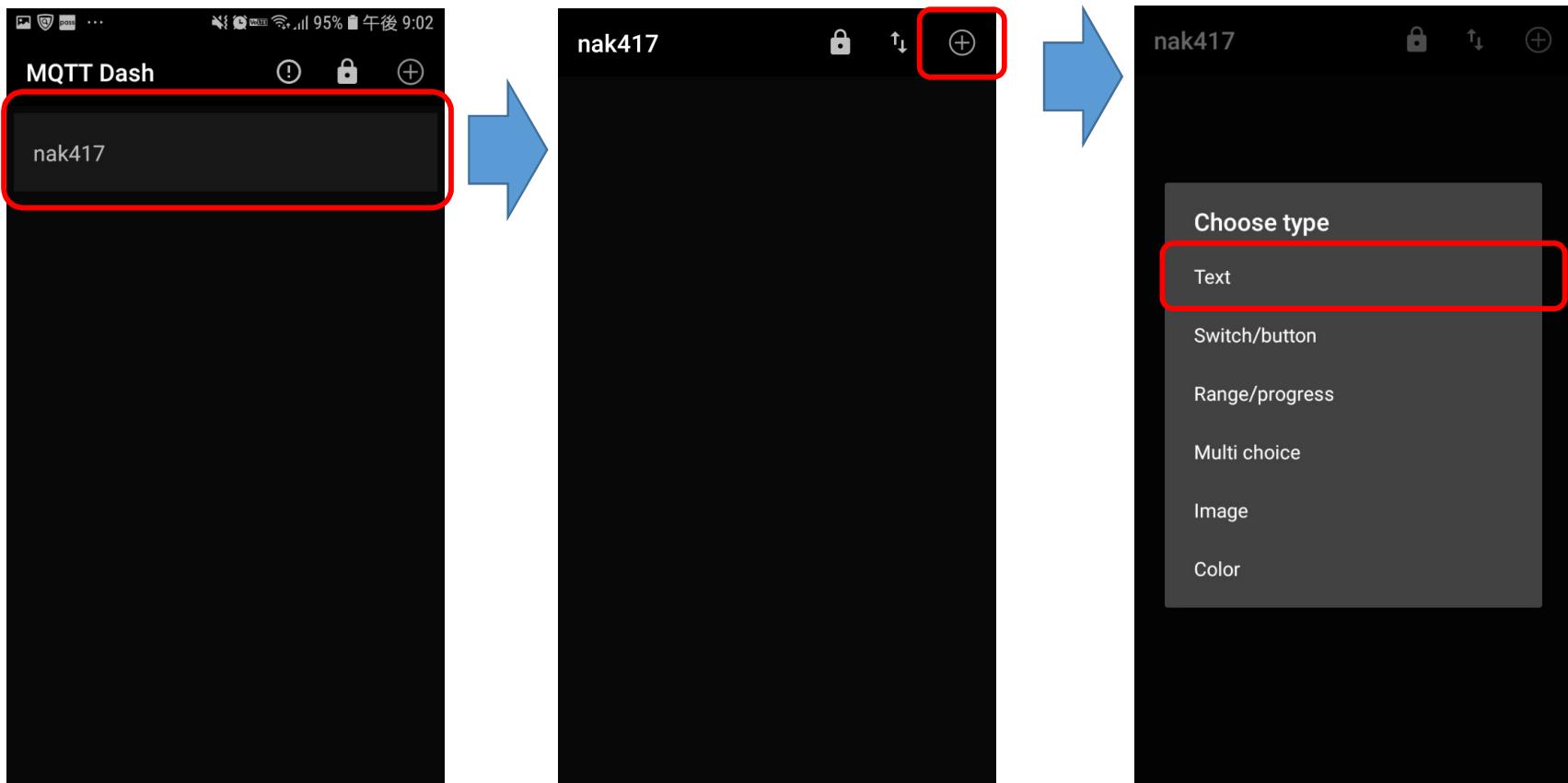
7-1 STM32(Publisher) とMQTT dash(Subscriber)を繋げる

(1) 5章で作成した Node-RED の構成をそのまま使用します。

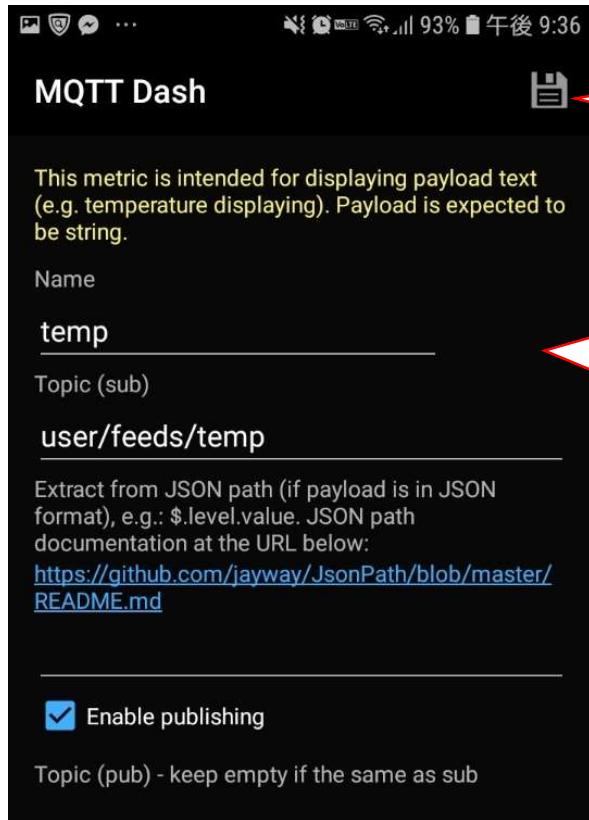


7-2 Node-RED(Broker) とMQTT dash(Subscriber)を繋げる

(1) Payload をテキストで表示する Subscriber を追加 (温度)

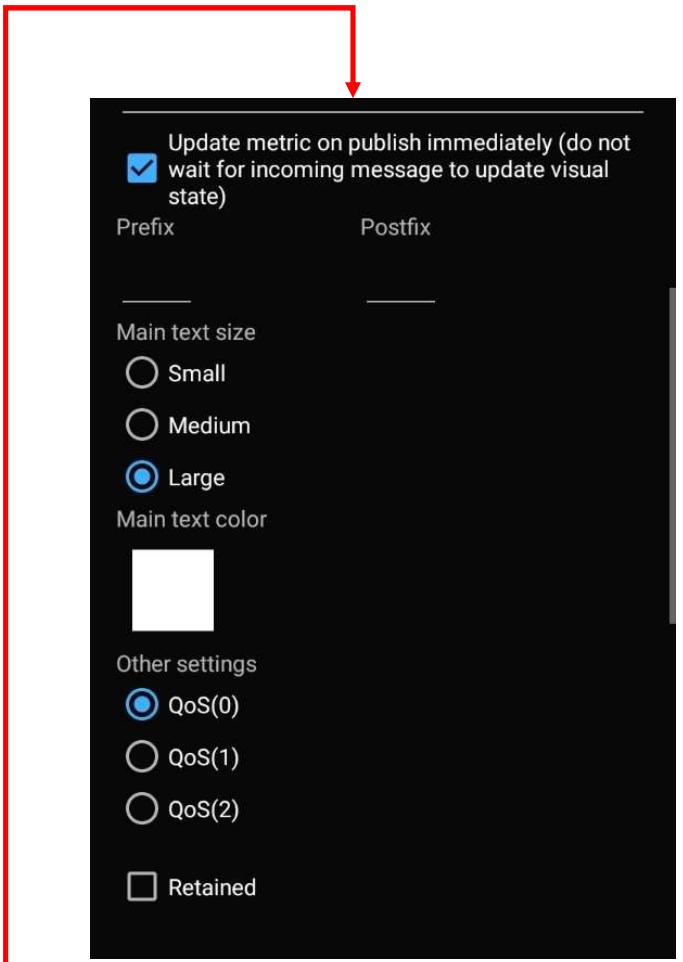


(2) Subscriberの設定

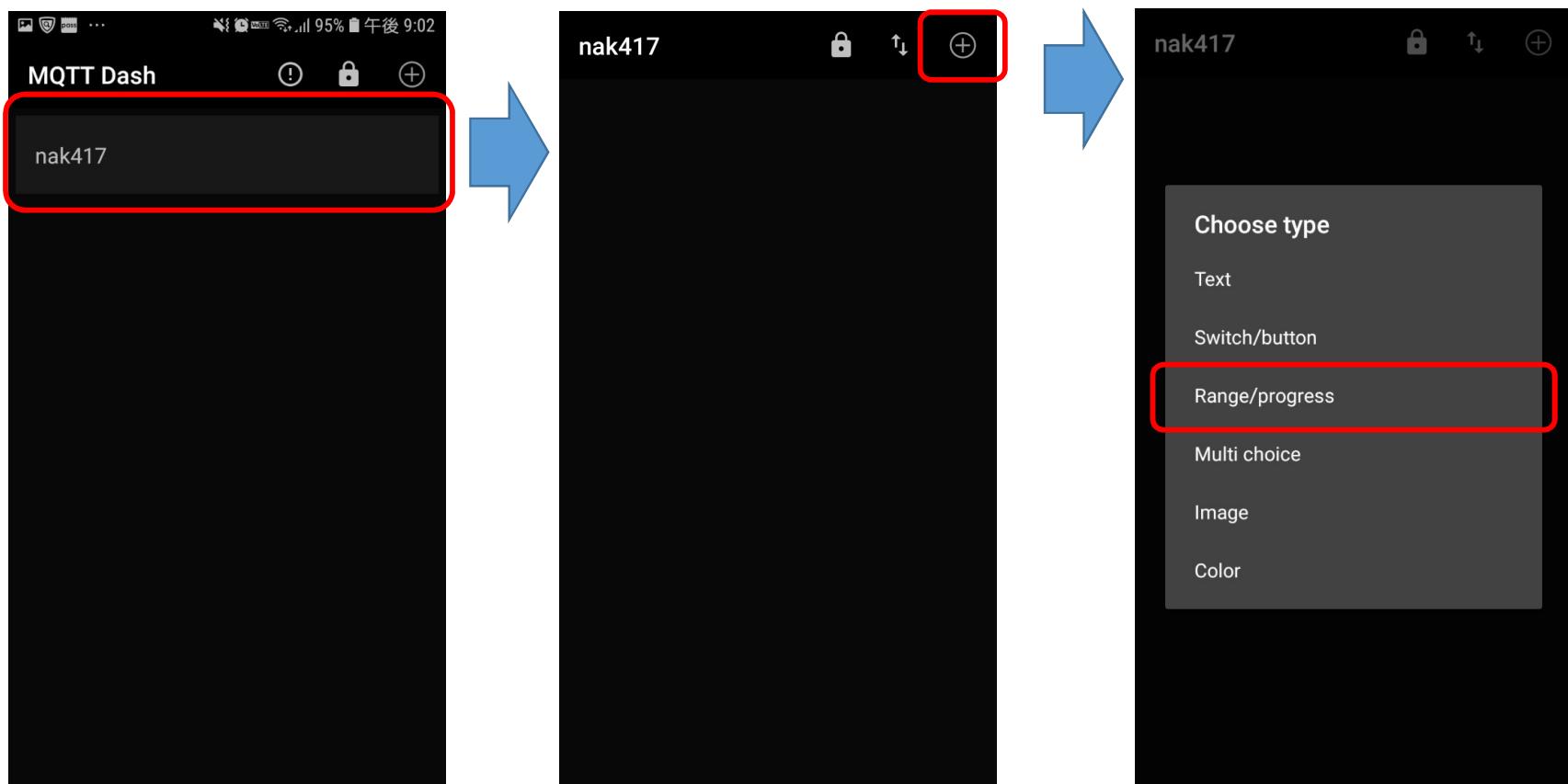


② 保存ボタン押下

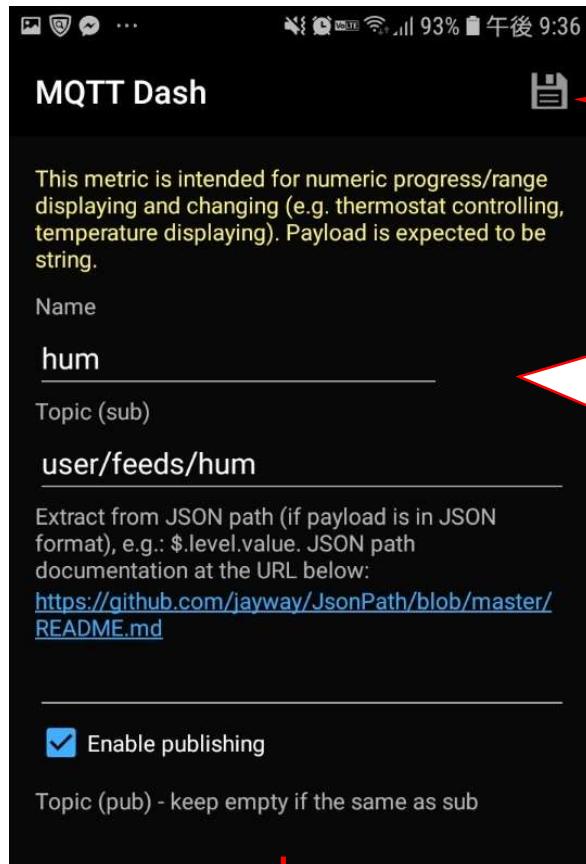
① Subscriberの設定
Name : temp
Topic : user/feeds/temp
その他 : (デフォルト)



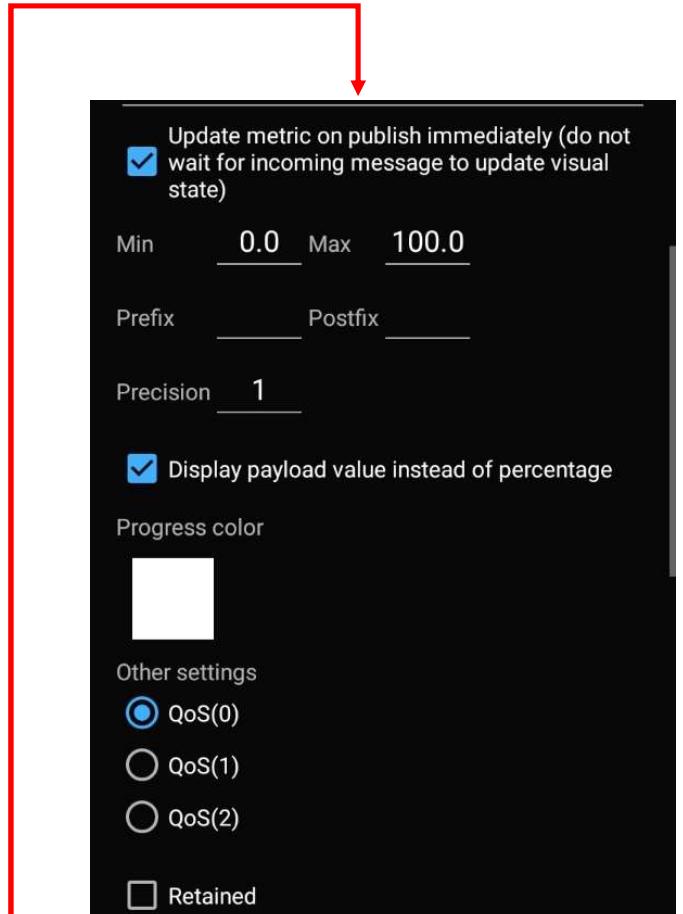
(3) Payload を Range/Progress で表示する Subscriber を追加 (湿度)



(4) Subscriberの設定



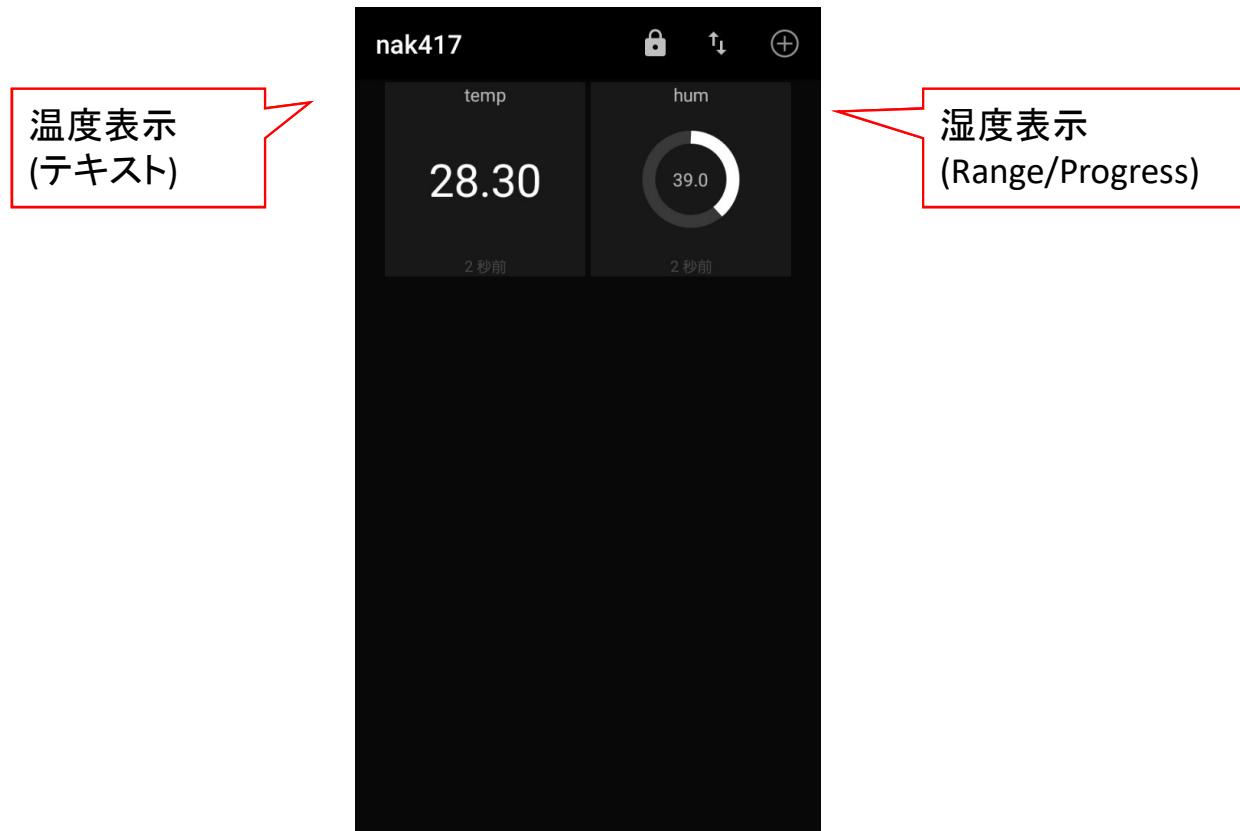
① Subscriberの設定
Name : hum
Topic : user/feeds/hum
Min : 0.0
Max : 100.0
Precision : 1 (小数点桁数)
その他 : デフォルト



(5) ダッシュボード作成完了

作成した Subscriber がダッシュボードに表示されることを確認します。

STM32から PublishされたメッセージをMQTTサーバを通して Subscriber で購読します。

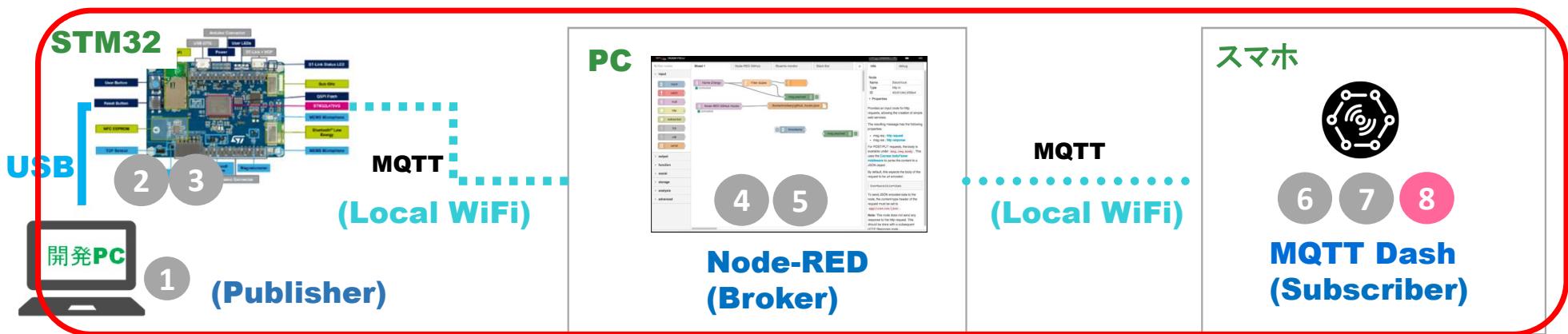


8 MQTT DashからSTM32のLEDを操作する

スマホ から STM32 のLEDを操作します。

8-1 Node-RED から STM32 のLEDを点灯させる

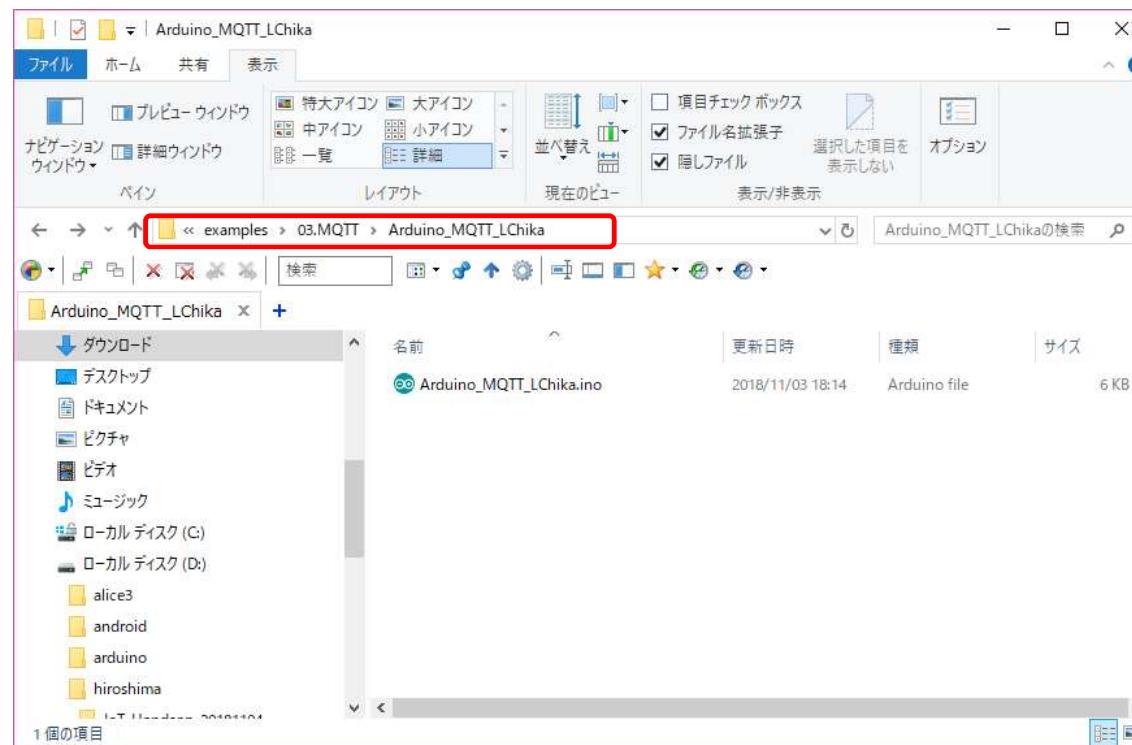
8-2 MQTT Dash から Node-RED に LED点灯



8-1 Node-RED から STM32 のLEDを点灯させる

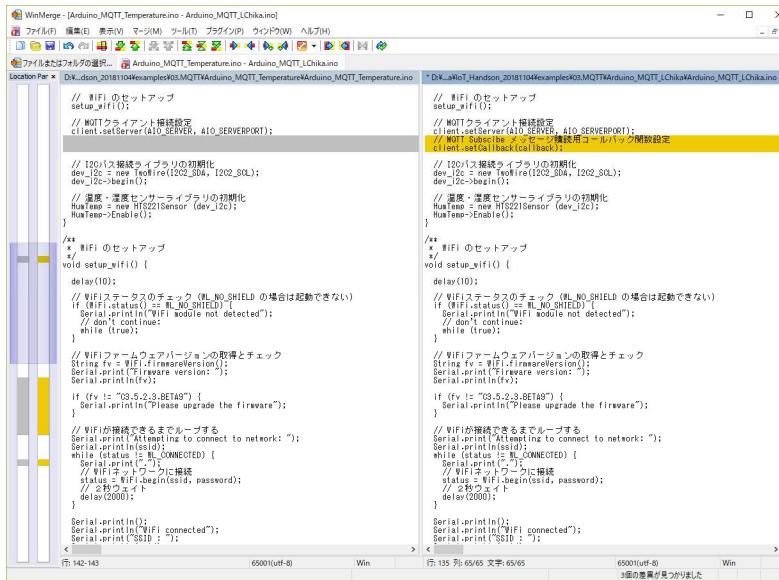
■サンプルコードを開く

「2. STM32のLEDを点滅させる」の「2-1 サンプルコードをダウンロードする」でダウンロードして解凍したフォルダから **03.MQTT/Arduino_MQTT_LChika**を開きます。



(参考) 5-5 の「Arduino_MQTT_Temperature」と「Arduino_MQTT_LChika」の違いを見てましょう

[WinMerge](#) が diff ツールとして有名。時間があるときに試してみてください。



```
// WiFi のセットアップ
setup_wifi();
// MQTT クライアント接続設定
client.setServer(AIO SERVER, AIO SERVERPORT);
// MQTT 接続用コールバック関数設定
client.setCallback(callback);

// I2C の初期化
dev_12c.begin(1202_SDA, 1202_SCL);
// 湿度・温度センサライブラリの初期化
HumTemp = new HCSR221Sensor(dev_12c);
HumTemp->Enable();

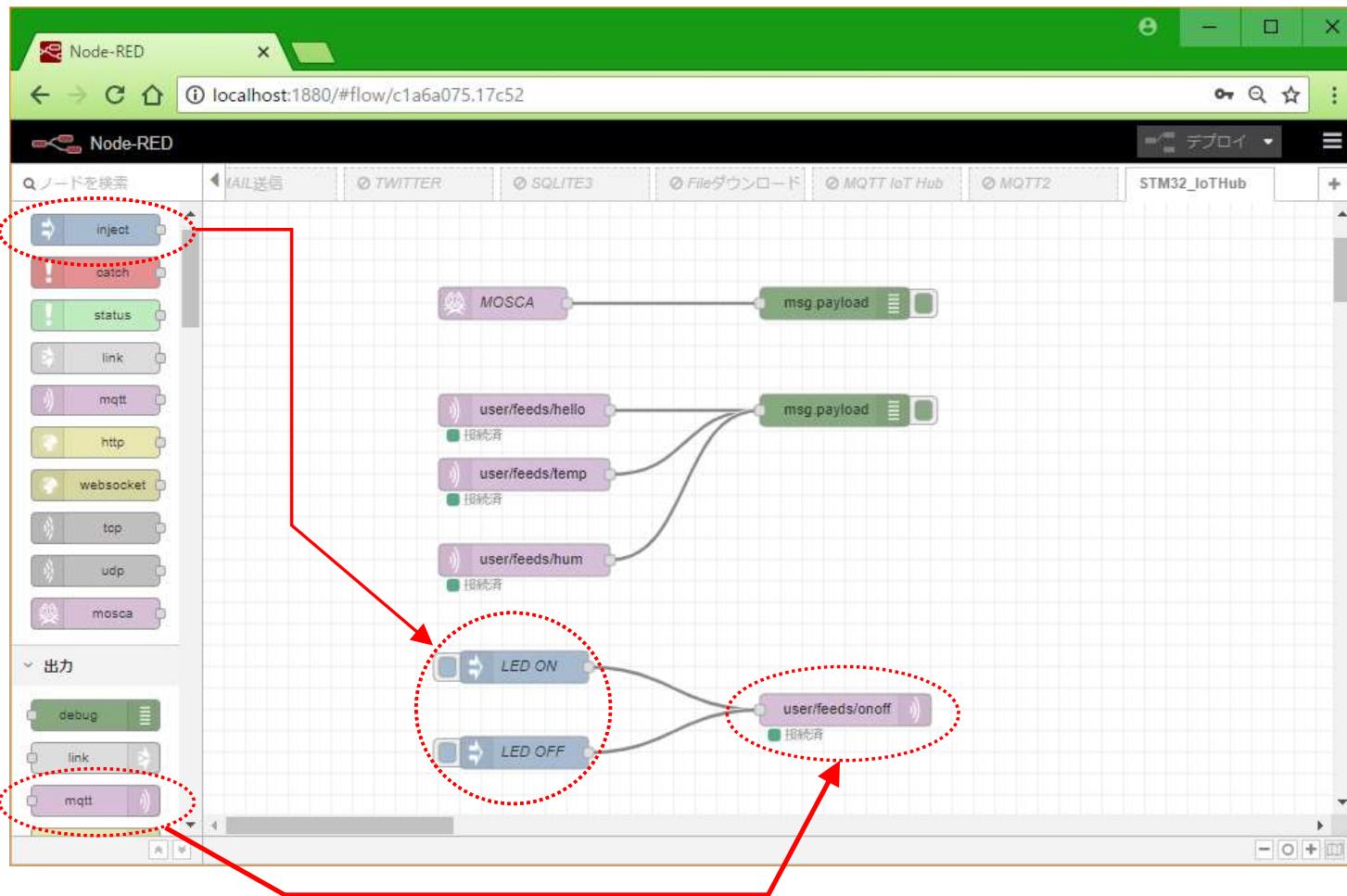
/*
 * WiFi のセットアップ
 */
void setup_wifi() {
    delay(10);
    // WiFi ライブドットのチェック (I2C_NO_SHIELD の場合は起動できない)
    if (WiFi.status() == I2C_NO_SHIELD) {
        Serial.println("WiFi module not detected");
        // don't continue
        while (true);
    }
    // WiFi ファームウェアバージョンの取得とチェック
    String fv = WiFi.firmwareVersion();
    Serial.print("Firmware version: ");
    Serial.println(fv);
    if (fv <= "0.9.5.2.0-BETA0") {
        Serial.println("Please upgrade the firmware");
    }
    // WiFiが接続できるまでループする
    Serial.println("Attempting to connect to network:");
    Serial.println(ssid);
    while (status != WL_CONNECTED) {
        delay(500);
        // WiFiネットワークに接続
        WiFi.begin(ssid, password);
        // 2秒ウェイド
        delay(2000);
    }
    Serial.println();
    Serial.println("WiFi connected");
    Serial.print("SSID : ");
}

// WiFi のセットアップ
void setup_wifi() {
    delay(10);
    // WiFi ライブドットのチェック (I2C_NO_SHIELD の場合は起動できない)
    if (WiFi.status() == I2C_NO_SHIELD) {
        Serial.println("WiFi module not detected");
        // don't continue
        while (true);
    }
    // WiFi ファームウェアバージョンの取得とチェック
    String fv = WiFi.firmwareVersion();
    Serial.print("Firmware version: ");
    Serial.println(fv);
    if (fv <= "0.9.5.2.0-BETA0") {
        Serial.println("Please upgrade the firmware");
    }
    // WiFiが接続できるまでループする
    Serial.println("Attempting to connect to network:");
    Serial.println(ssid);
    while (status != WL_CONNECTED) {
        delay(500);
        // WiFiネットワークに接続
        WiFi.begin(ssid, password);
        // 2秒ウェイド
        delay(2000);
    }
    Serial.println();
    Serial.println("WiFi connected");
    Serial.print("SSID : ");
}
```

WiFi / MQTT の設定は 5-5 と同じように修正します。WinMergeは比較しながら編集することもできます。

(3) STM32への書き込み

(1) Node-RED で MQTT(出力)ノードを追加



(2) Inject 設定 (1/2)

Node-RED上からメッセージを注入できます。

The image shows a Node-RED flow on the left and a configuration dialog for an 'inject' node on the right.

Node-RED Flow:

- A blue 'LED ON' node is connected to a purple 'user/feeds/onoff' node.
- A blue 'LED OFF' node is also connected to the same purple 'user/feeds/onoff' node.
- Both the 'LED ON' and 'LED OFF' nodes have red dotted ovals around them, indicating they are selected.
- A red arrow points from the top of the 'LED ON' node to the 'inject' node configuration dialog.

inject ノードを編集

① 設定変更

- ペイロードタイプ : 文字列(a-z)
- ペイロード : ON
- トピック : ON
- 繰り返し : なし
- 名前 : LED ON

② 完了

中止 完了

削除

プロパティ

ペイロード ON

トピック ON

Node-RED起動の 0.1 秒後、以下を行う

繰り返し なし

名前 LED ON

注記: 「指定した時間間隔、日時」と「指定した日時」(はcronを)します。詳細はノードの「情報」を確認してください。

設定

(3) Inject 設定 (2/2)

Node-RED上からメッセージを注入できます。

The screenshot shows a Node-RED flow. On the left, there are two blue button nodes: "LED ON" and "LED OFF". Arrows from both buttons point to a central purple "inject" node labeled "user/feeds/onoff". A green "接続済" (Connected) status indicator is visible below the inject node. A red arrow points from the "LED OFF" button to the inject node. On the right, the "inject" node configuration dialog is open. It has fields for "ペイロード" (Payload) set to "OFF", "トピック" (Topic) set to "OFF", "繰り返し" (Repeat) set to "なし" (None), and "名前" (Name) set to "LED OFF". A note at the bottom of the dialog says: "注釈: 「指定した時間間隔、日時」と「指定した日時」が重複する場合は「指定した時間間隔、日時」が優先されます。詳細はノードの「情報」を確認してください。" (Note: If the specified time interval and date/time overlap, the specified time interval takes precedence. For details, please refer to the node's 'Information' tab.)

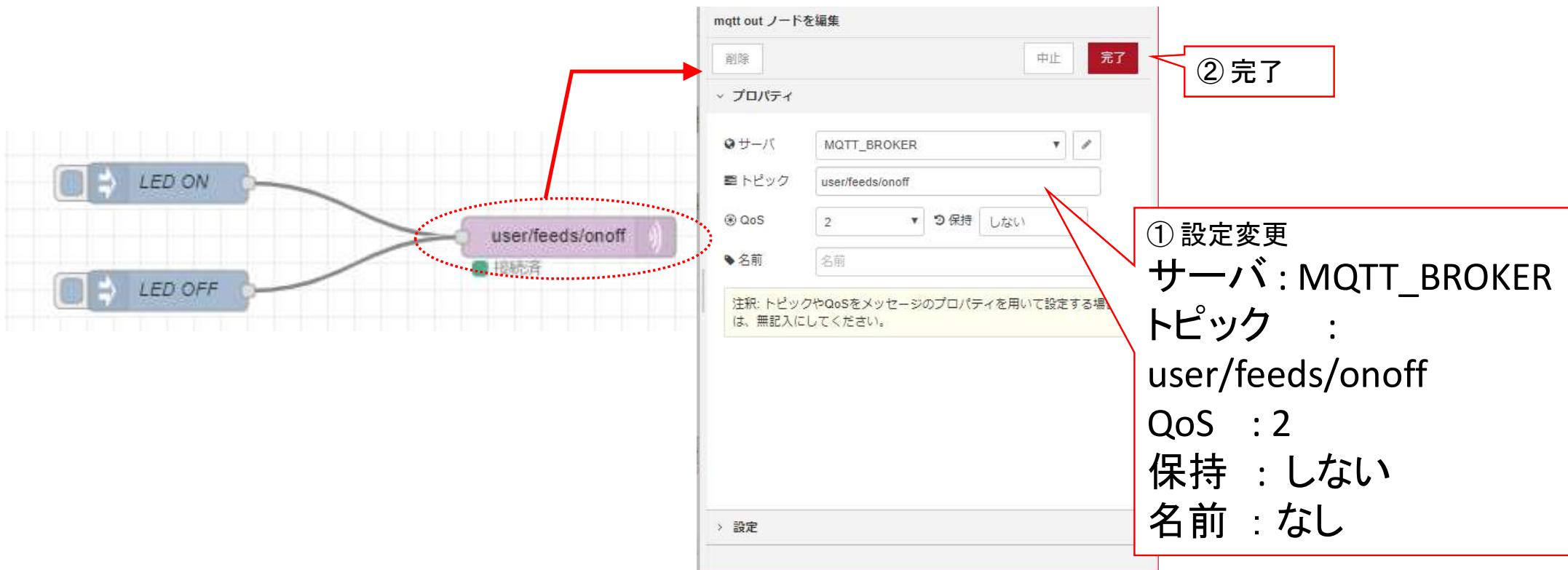
② 完了

① 設定変更
ペイロードタイプ : 文字列(a-z)
ペイロード : OFF
トピック : OFF
繰り返し : なし
名前 : LED OFF

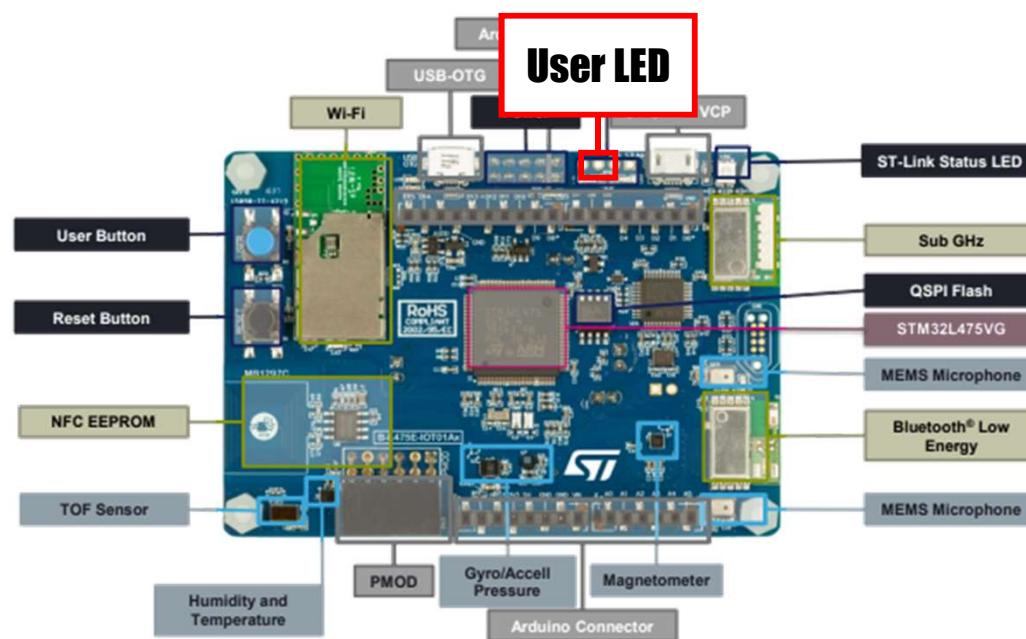
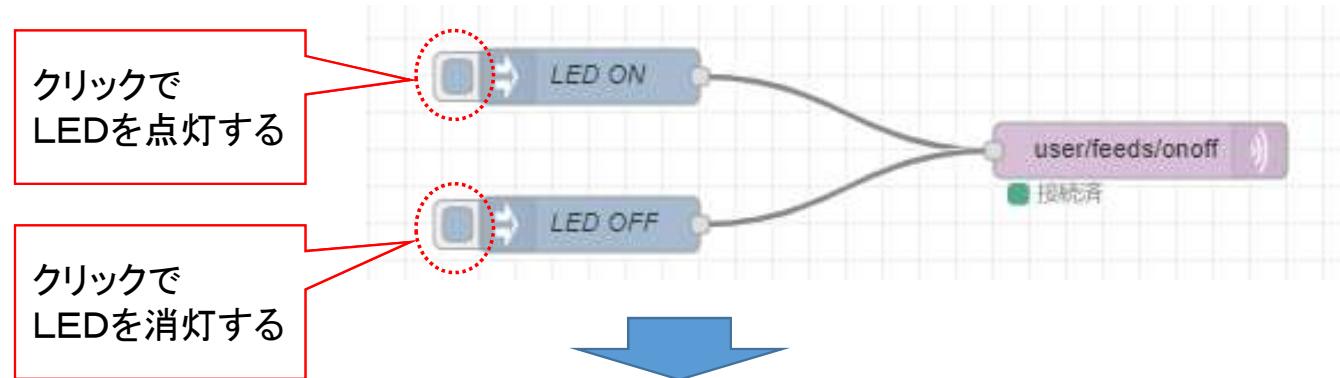
(4) MQTT Publish 設定

注入したメッセージを publishします。

トピックは、MQTT Publish ノードのものとなります。

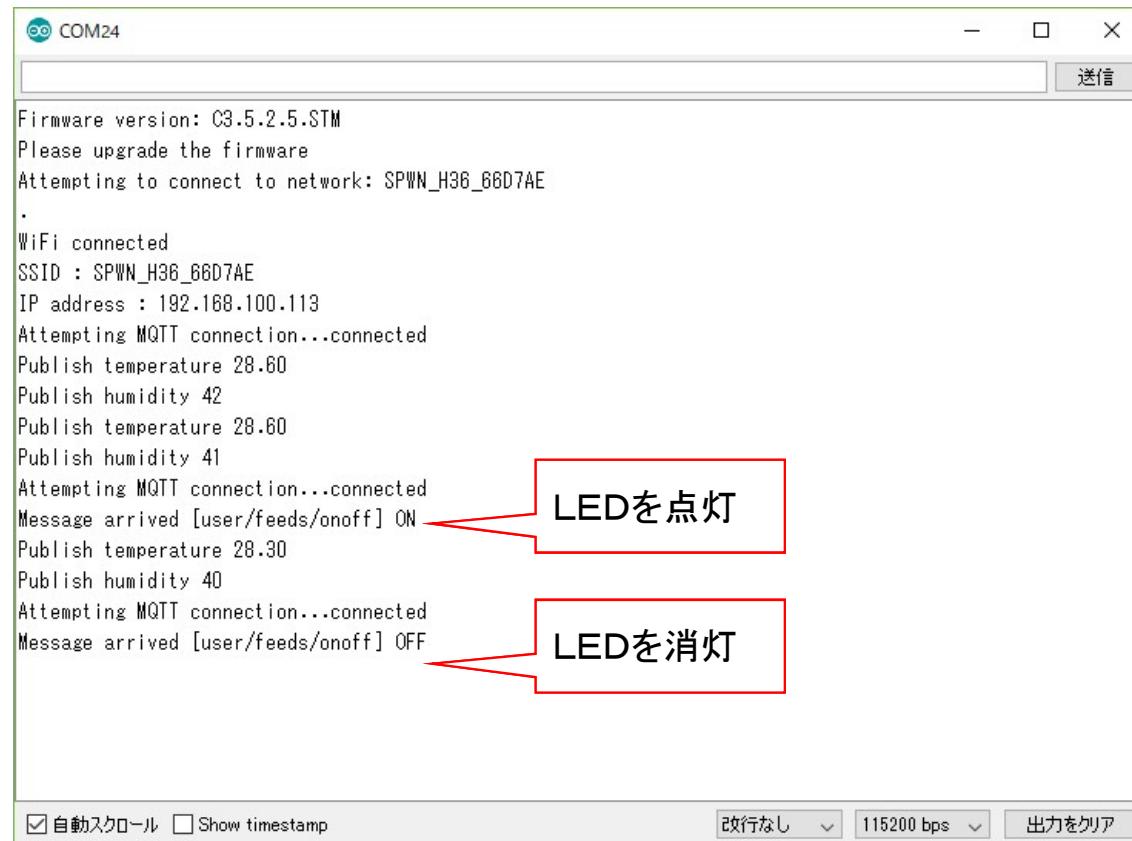


(5) Node-RED で Lチカ



(6) USBを経由して送られてくるデータの確認

受信ループとPublishを同時にやっているので若干のもたつきがあります。



```
COM24
Firmware version: C3.5.2.5.STM
Please upgrade the firmware
Attempting to connect to network: SPWN_H36_B6D7AE
.
WiFi connected
SSID : SPWN_H36_B6D7AE
IP address : 192.168.100.113
Attempting MQTT connection...connected
Publish temperature 28.60
Publish humidity 42
Publish temperature 28.60
Publish humidity 41
Attempting MQTT connection...connected
Message arrived [user/feeds/onoff] ON
Publish temperature 28.30
Publish humidity 40
Attempting MQTT connection...connected
Message arrived [user/feeds/onoff] OFF
```

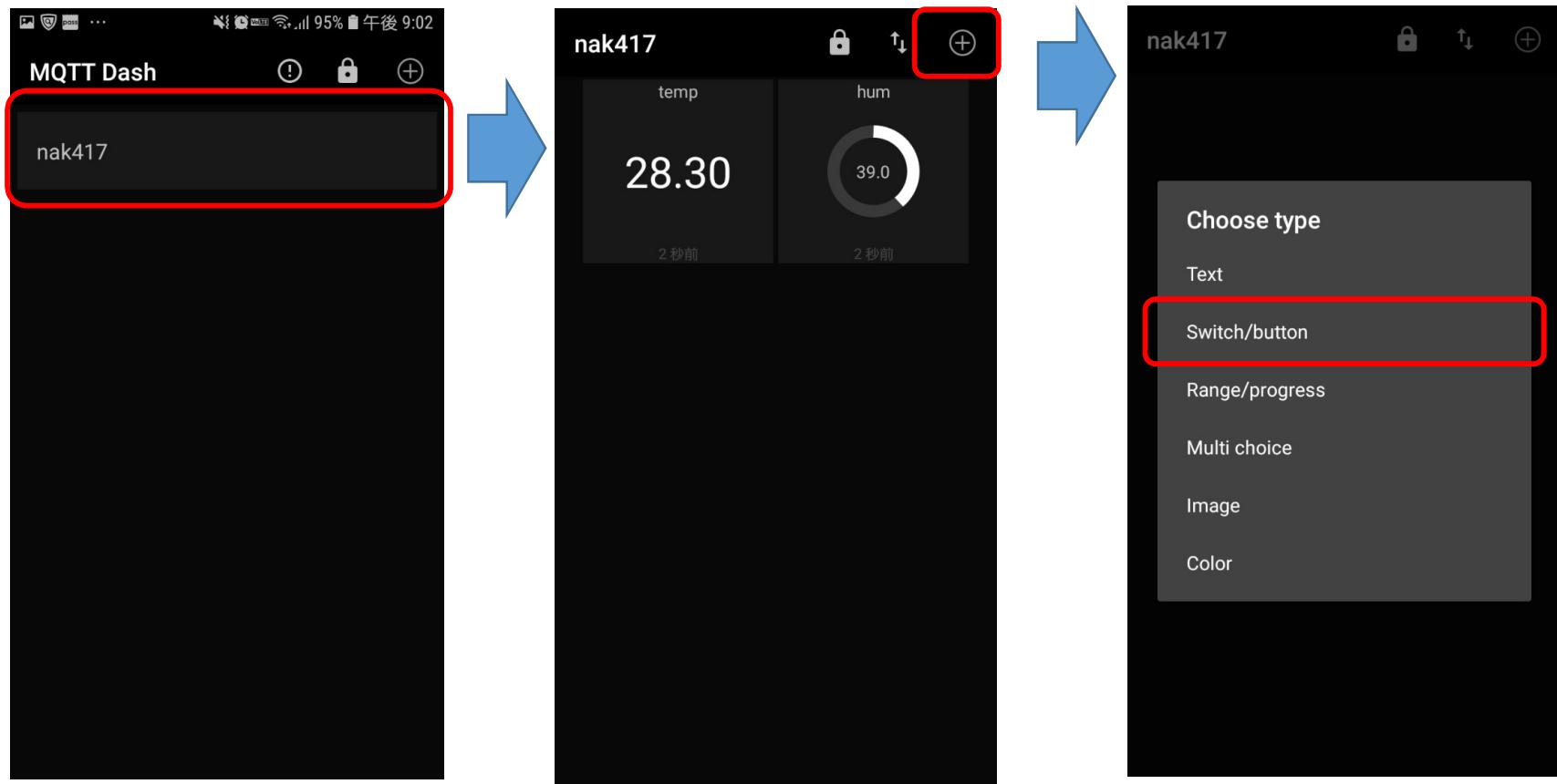
自動スクロール Show timestamp 改行なし 115200 bps 出力をクリア

LEDを点灯

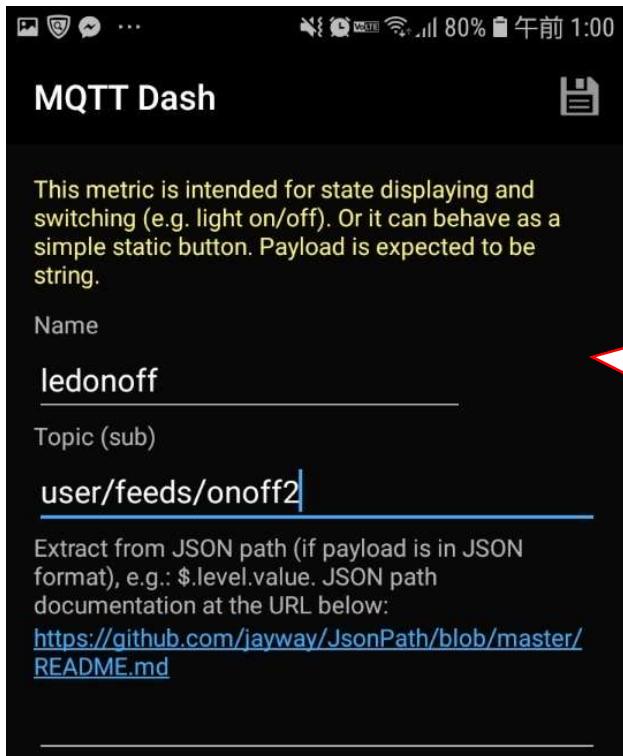
LEDを消灯

8-2 MQTT Dash から Node-RED に LED点灯

(1) Payload を Switch/Buttonで表示する Subscriber を追加



(2) Switch/Button の設定



③ 保存

① Switch/Button設定変更

Name: ledonoff

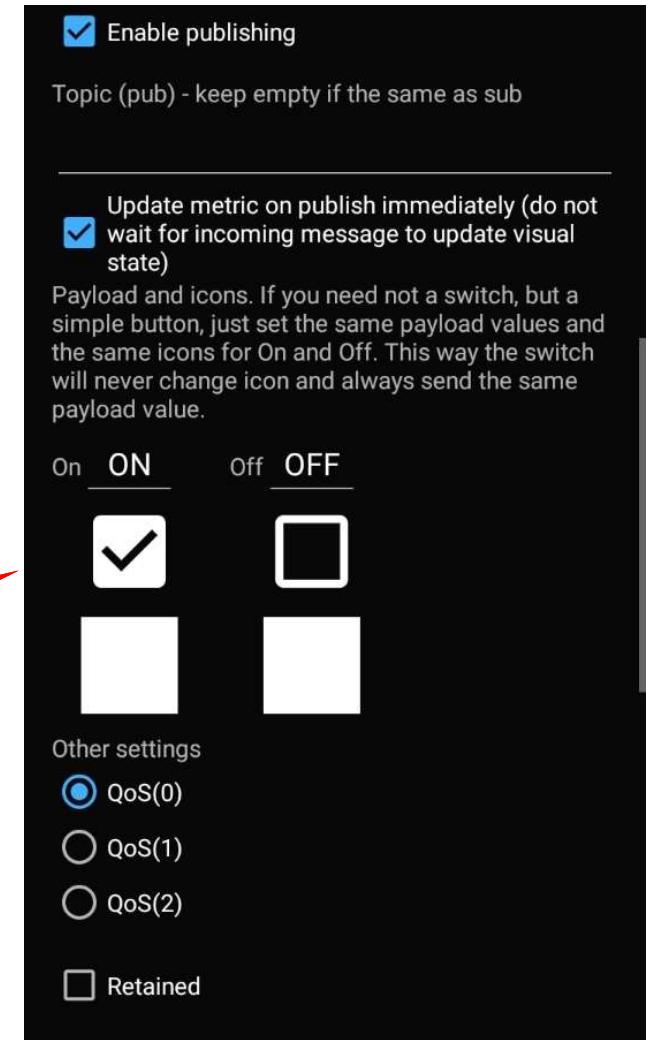
Topic: user/feeds/onoff2
(onoffでないことに注意)

② ボタンを押したときの値の設定

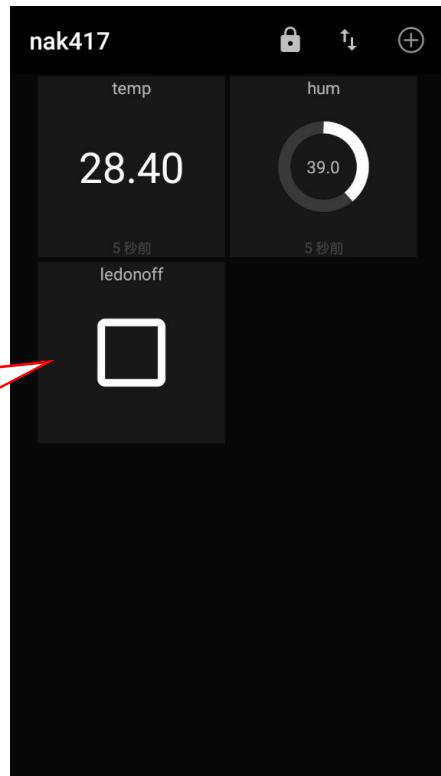
On: ON

Off: OFF

(この値が Payloadになる)

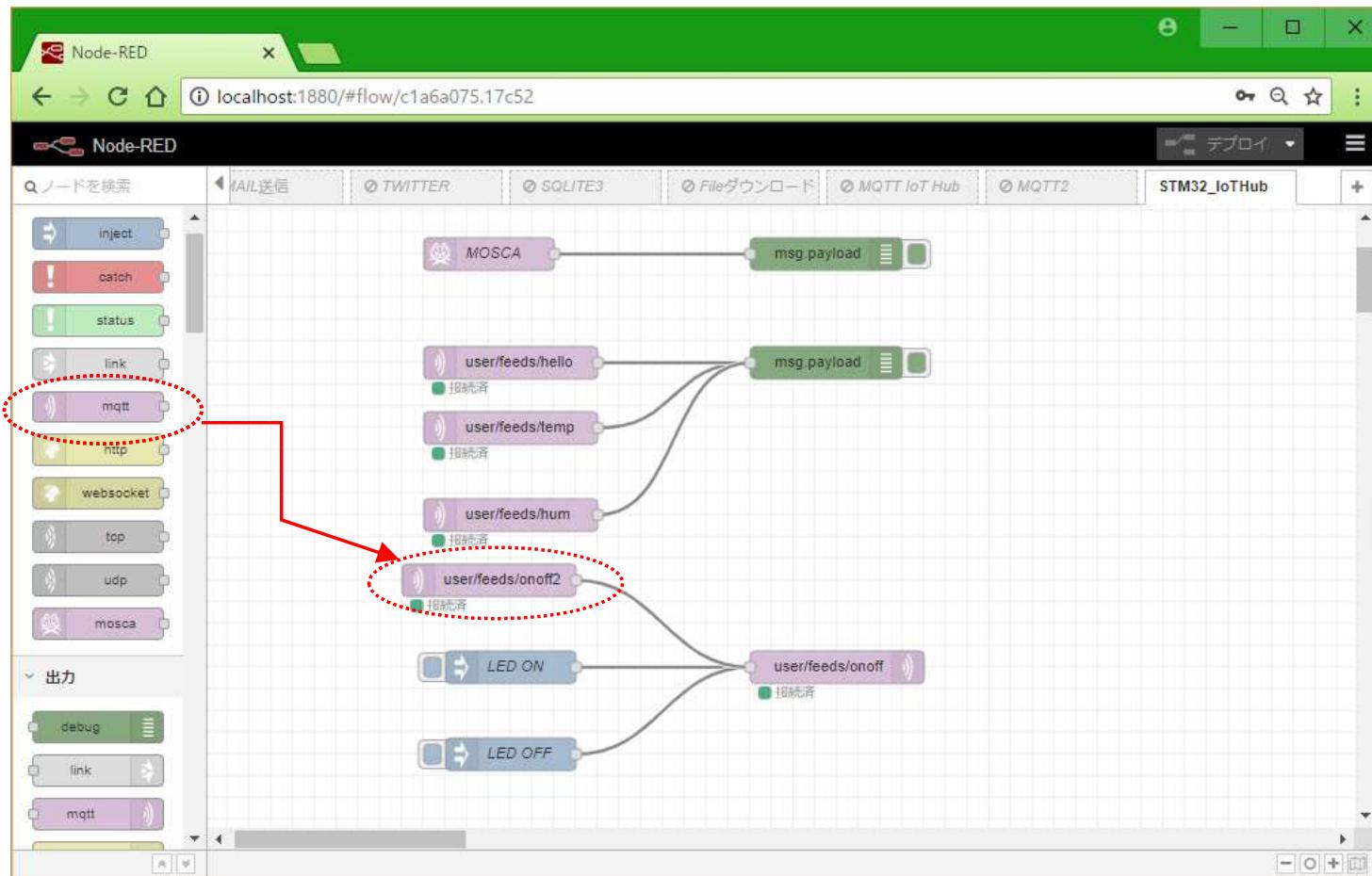


(3) Switch/Button 追加後



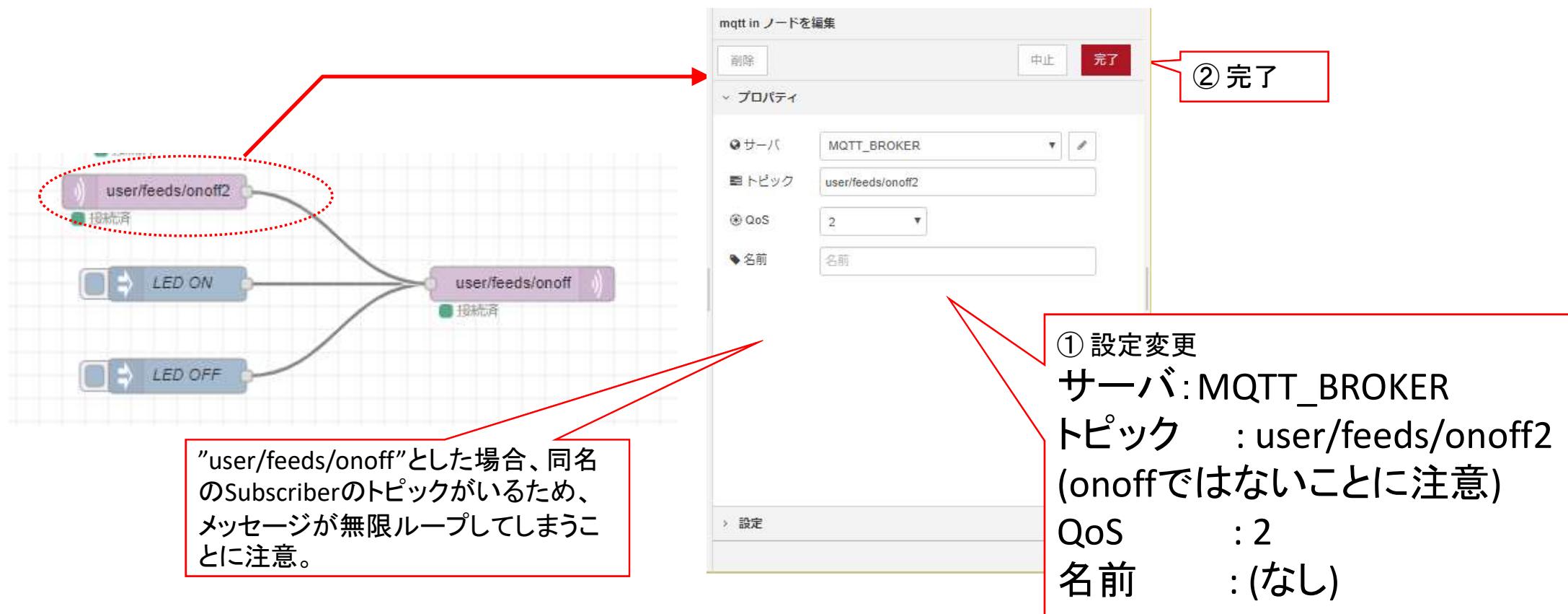
ledonoffボタン
チェックなし: LED消灯
チェックあり: LED点灯

(4) Node-RED で MQTT(入力)ノードを追加

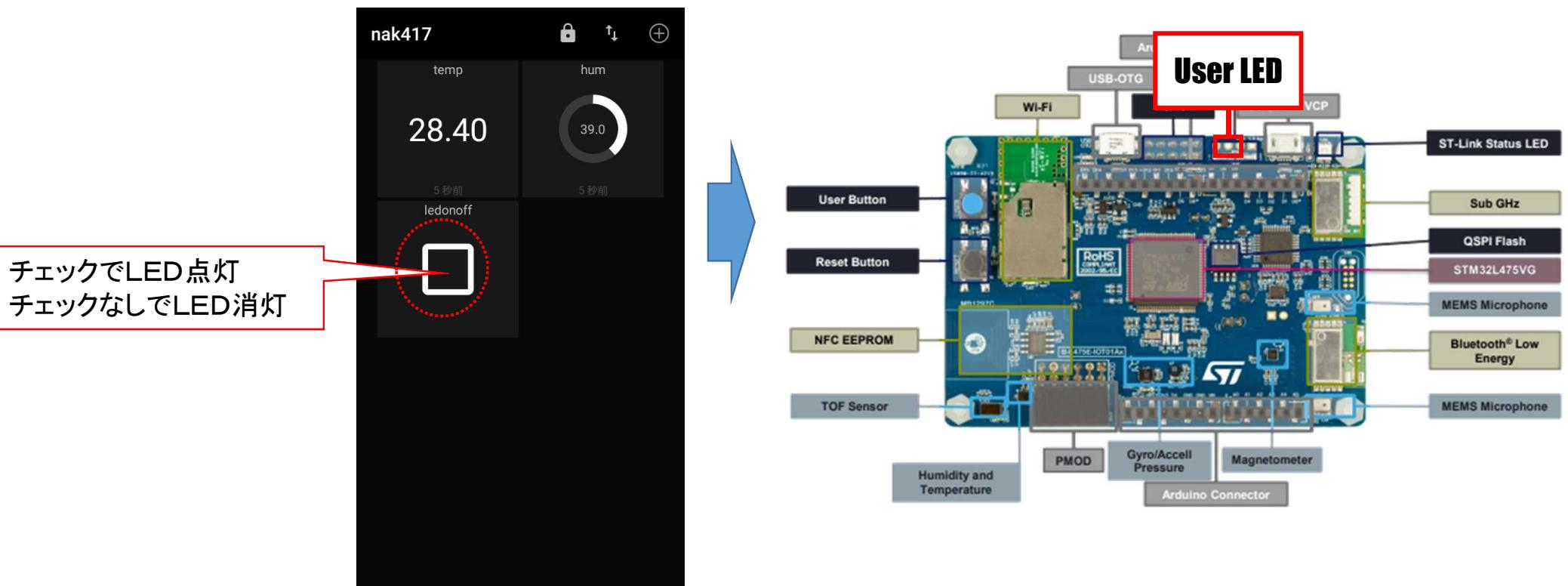


(5) MQTT(入力)ノードの設定

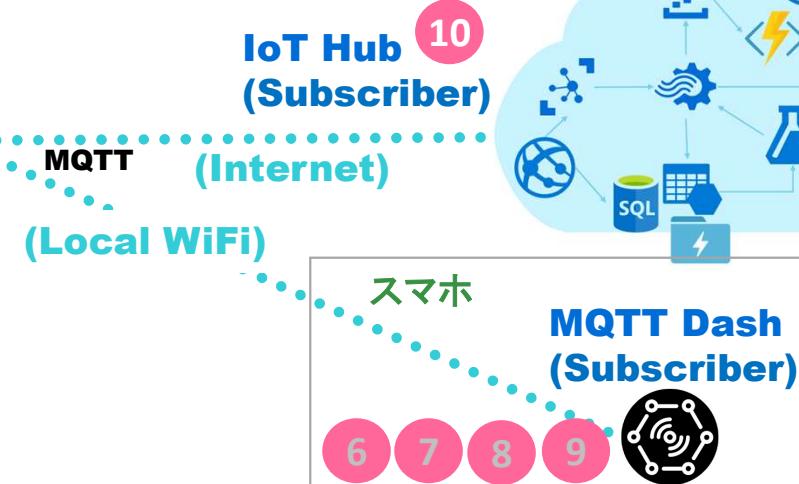
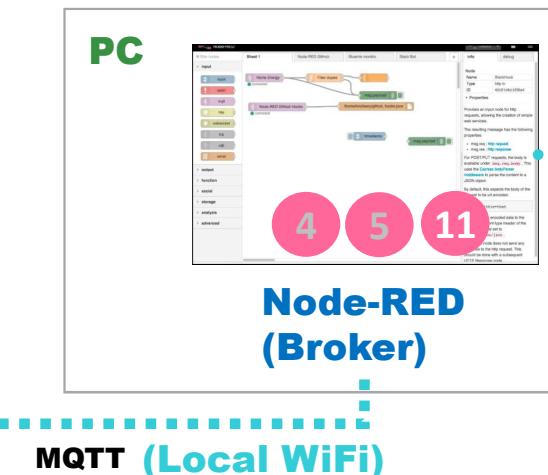
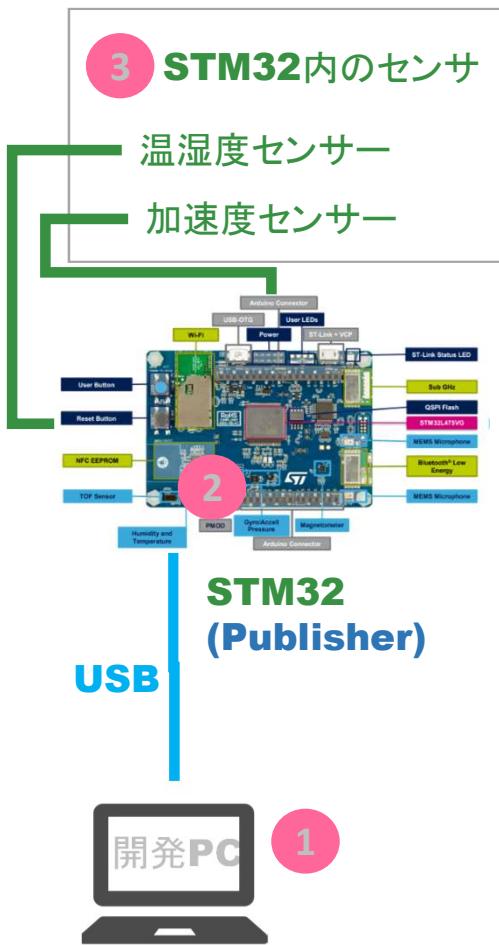
MQTT Dash がPublishするメッセージを購読します。トピック名はMQTT Dash と合わせて、"user/feeds/onoff2" とします。



(6) MQTT Dash で Lチカ



9 Azure IoT Hub の準備



- 10 Azure IoT Hub の準備
- 11 Node-RED と Azure IoT Hub をつなげる

12/2 のハンズオン資料を参照のこと。

ハンズオンクラウド側のダイジェスト版

また、補足資料として以下があります。

[IoTKitHoLV4_IoTKit20181202_Hiroshima\(復習用補足付\).ppt](#)

10-1 Node-RED と Azure IoT Hub をつなげる

10-1 Node-REDにフローの追加

Node-REDに新しいフローを追加します

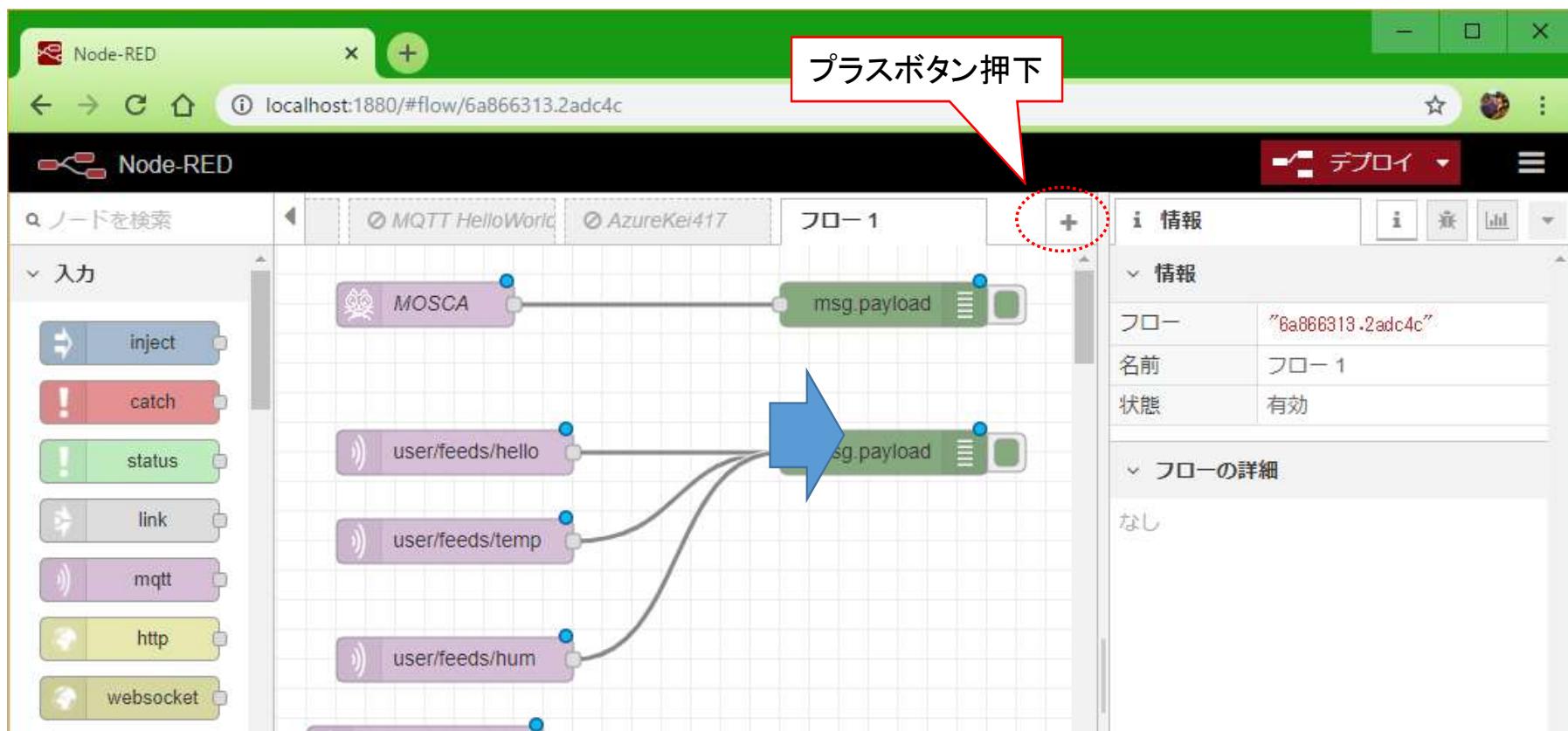
10-2 パレットにAzure-IoT-Hubの追加

10-3 Azure-IoT-Hubの設定

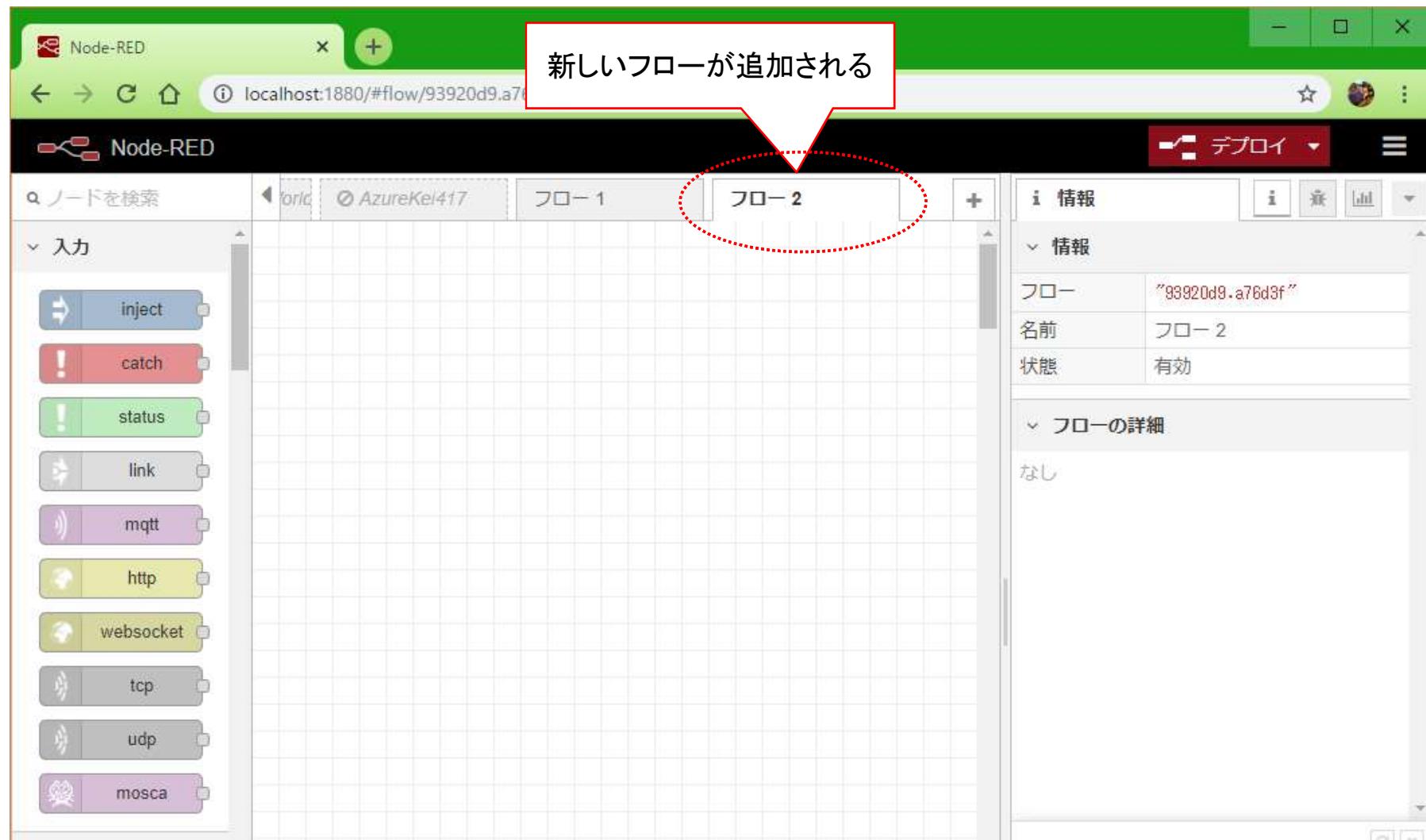
10-4 動作確認

10-1 Node-REDにフローの追加

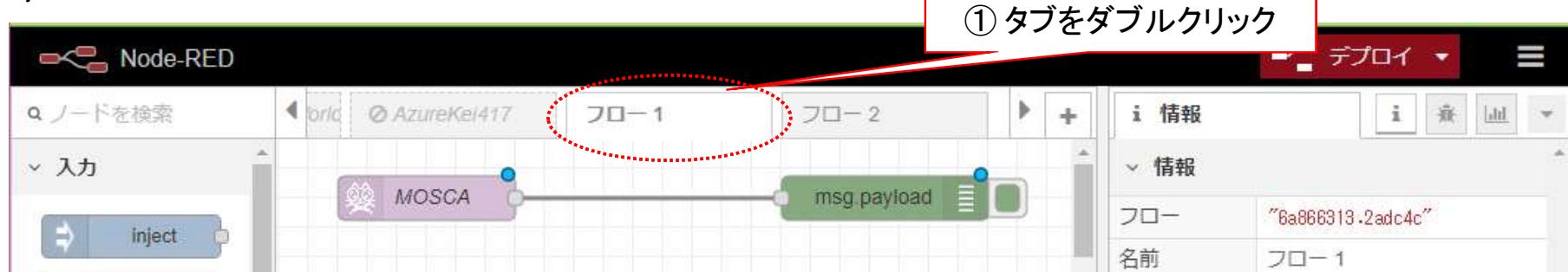
(1) Node-REDのタブの右端にある プラスボタンを押下



(2) 新しいフローが追加される



(3) 前回作成したフローを無効に設定



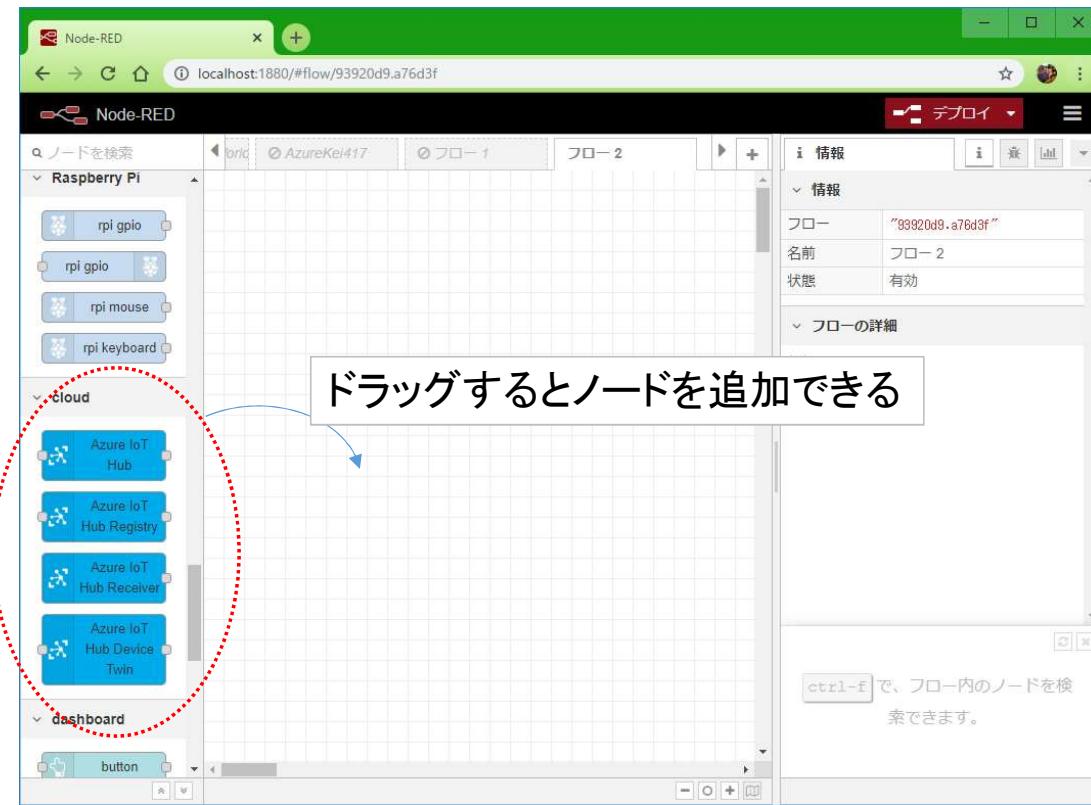
10-2 パレットにAzure-IoT-Hubの追加

- (1) メニューからパレットの管理を開く
- (2) ノード “node-red-contrib-azure-iot-hub” を追加



(3) mqtt , mosca のノードが追加されたことを確認します。

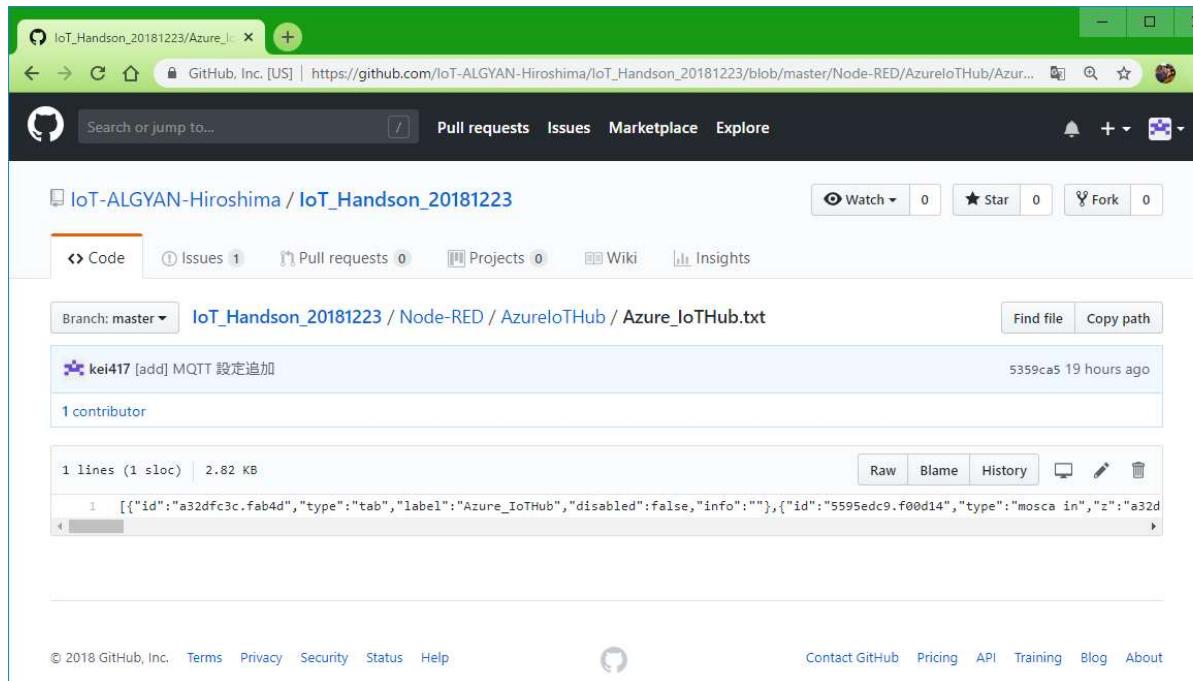
cloud に、Azure IoT Hub XXXX が追加されます



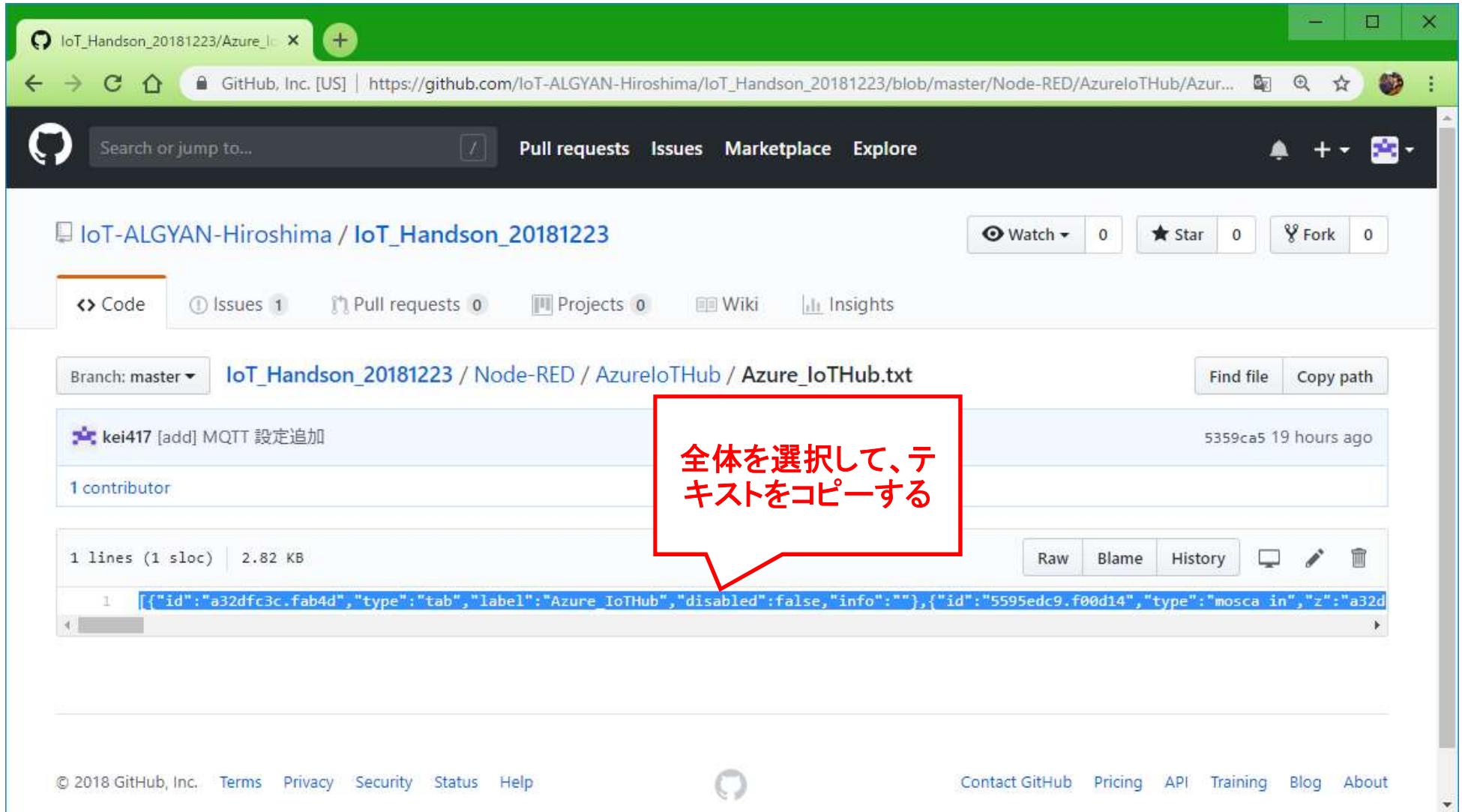
(1) ノード構成のページにアクセス

GitHubにAzure用のNode-REDの構成を用意しました。
以下にアクセスしてください。

https://github.com/IoT-ALGYAN-Hiroshima/IoT_Handson_20181223/blob/master/Node-RED/AzureIoTHub/Azure_IoTHub.txt



(2) ノード構成をクリップボードにコピー

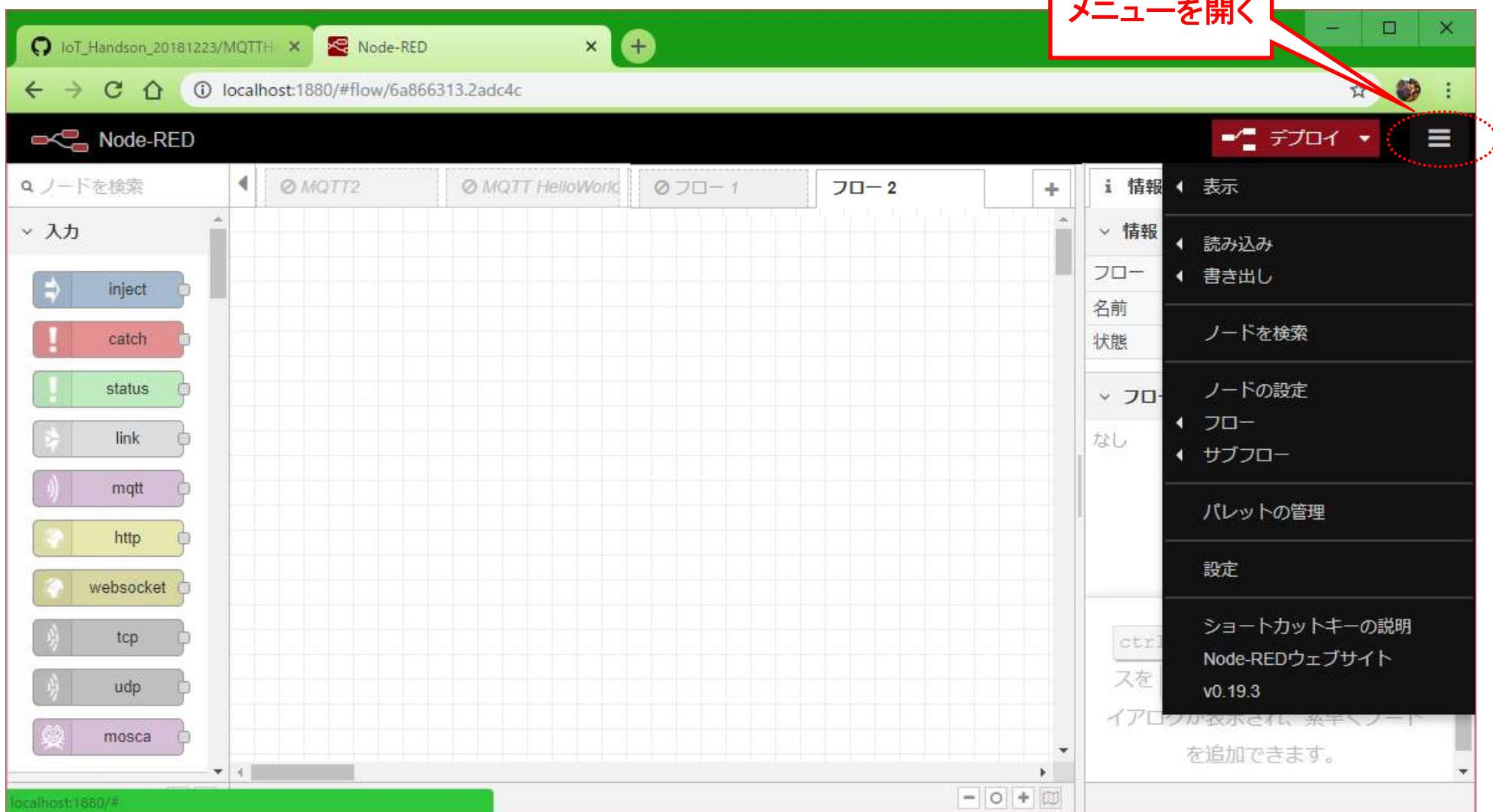


The screenshot shows a GitHub repository page for 'IoT_ALGYAN-Hiroshima/IoT_Handson_20181223'. The repository has 0 stars and 0 forks. The 'Code' tab is selected, showing the file 'Azure_IoTHub.txt'. A red box highlights the code area, and a red callout points to the text '全体を選択して、テキストをコピーする' (Select all and copy the text). The code content is:

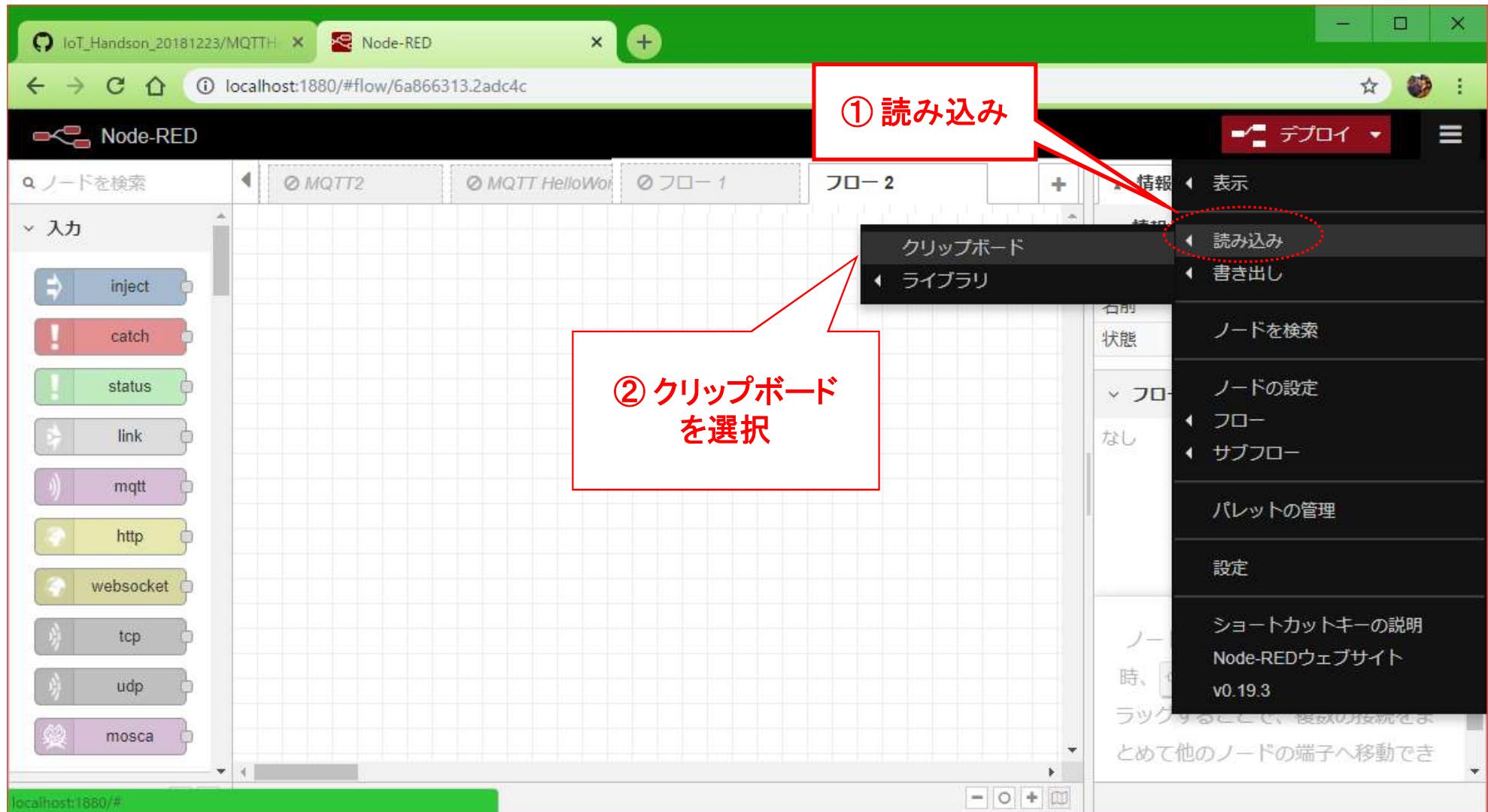
```
[{"id": "a32dfc3c.fab4d", "type": "tab", "label": "Azure IoTHub", "disabled": false, "info": ""}, {"id": "5595edc9.f00d14", "type": "mosca_in", "z": "a32d"}]
```

At the bottom of the page, there are links for GitHub: © 2018 GitHub, Inc. Terms Privacy Security Status Help, Contact GitHub, Pricing, API, Training, Blog, About.

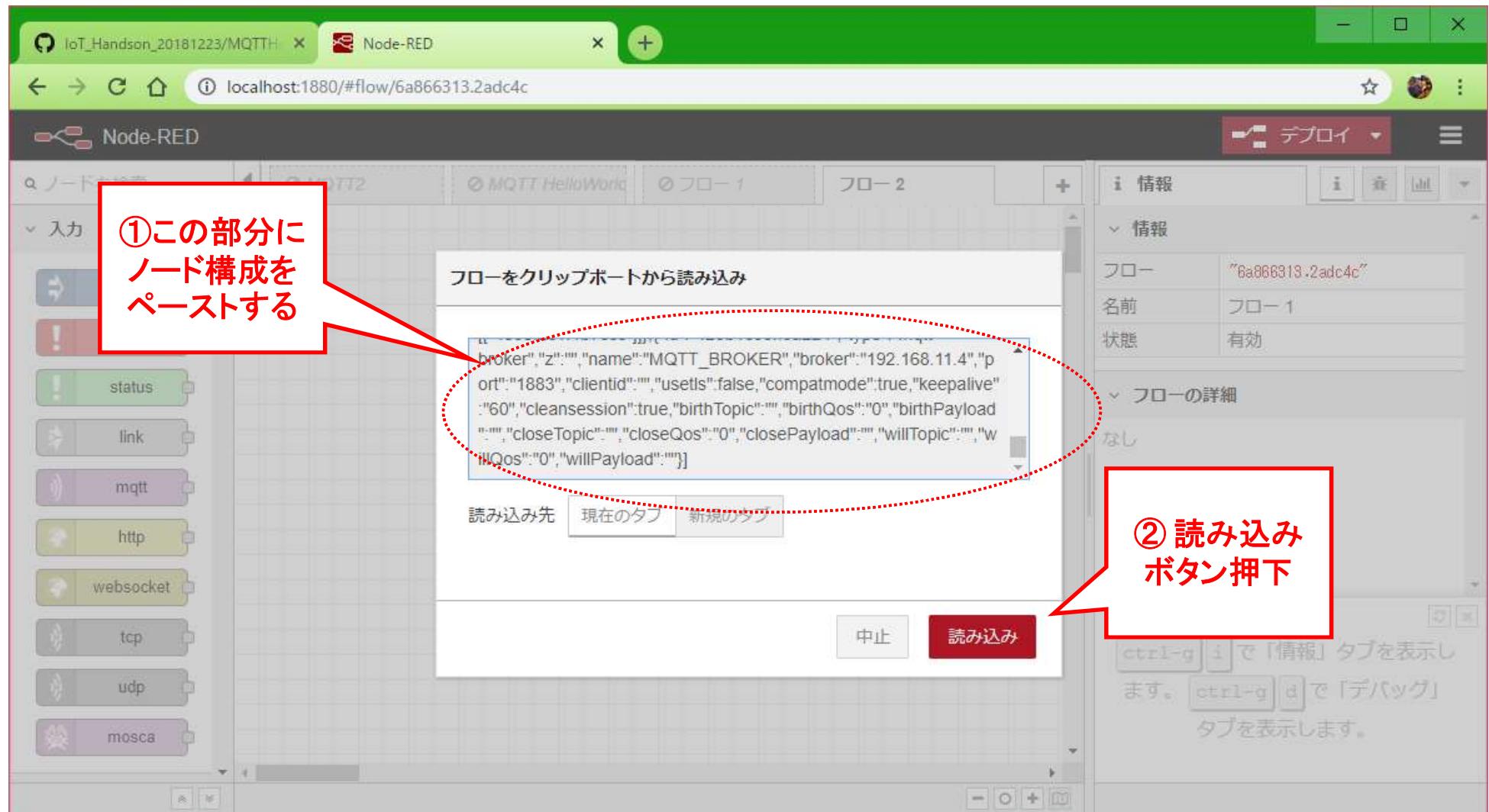
(3) Node-REDのメニューを開く



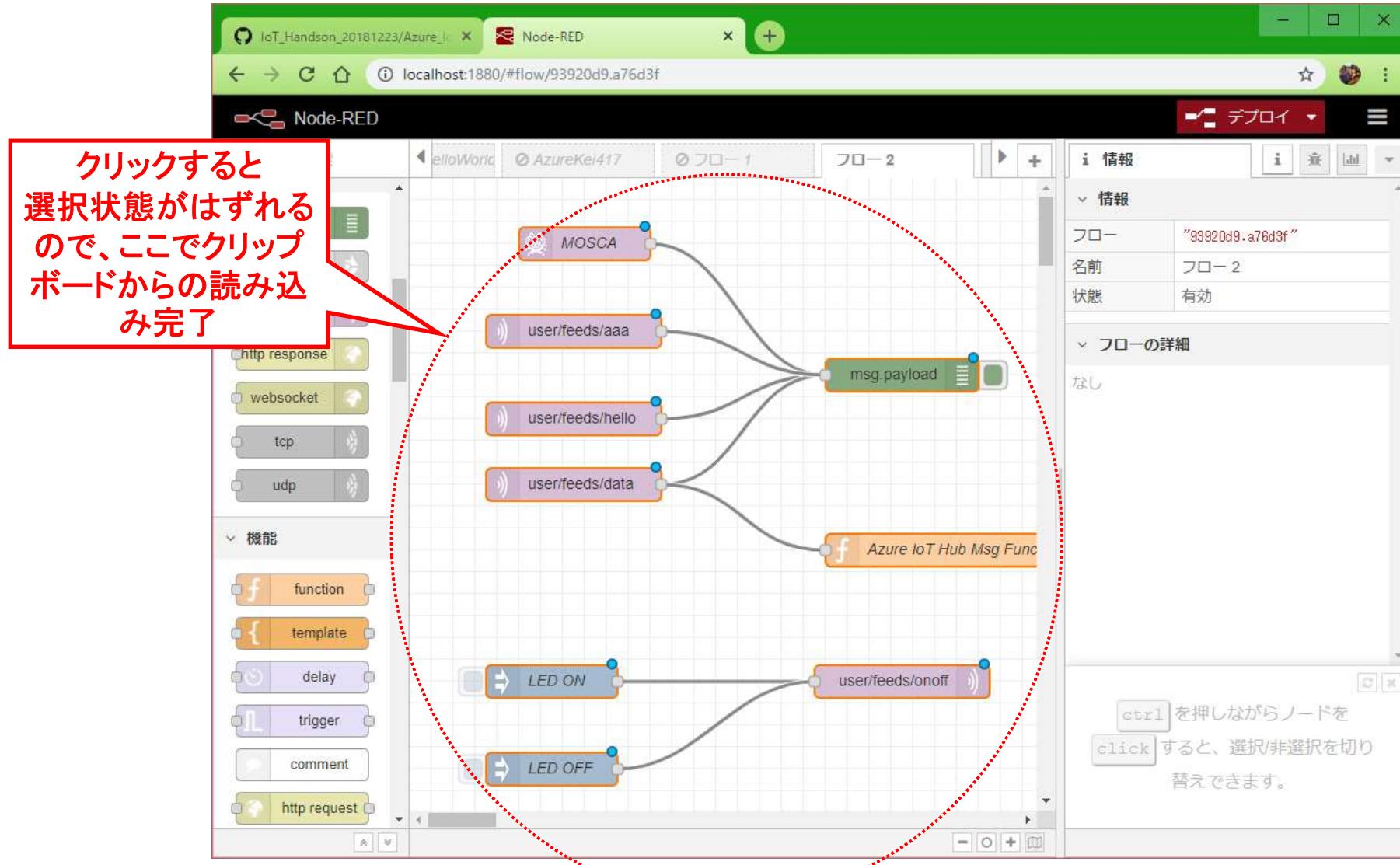
(4)クリップボードからの読み込み



(5)クリップボードからの読み込み



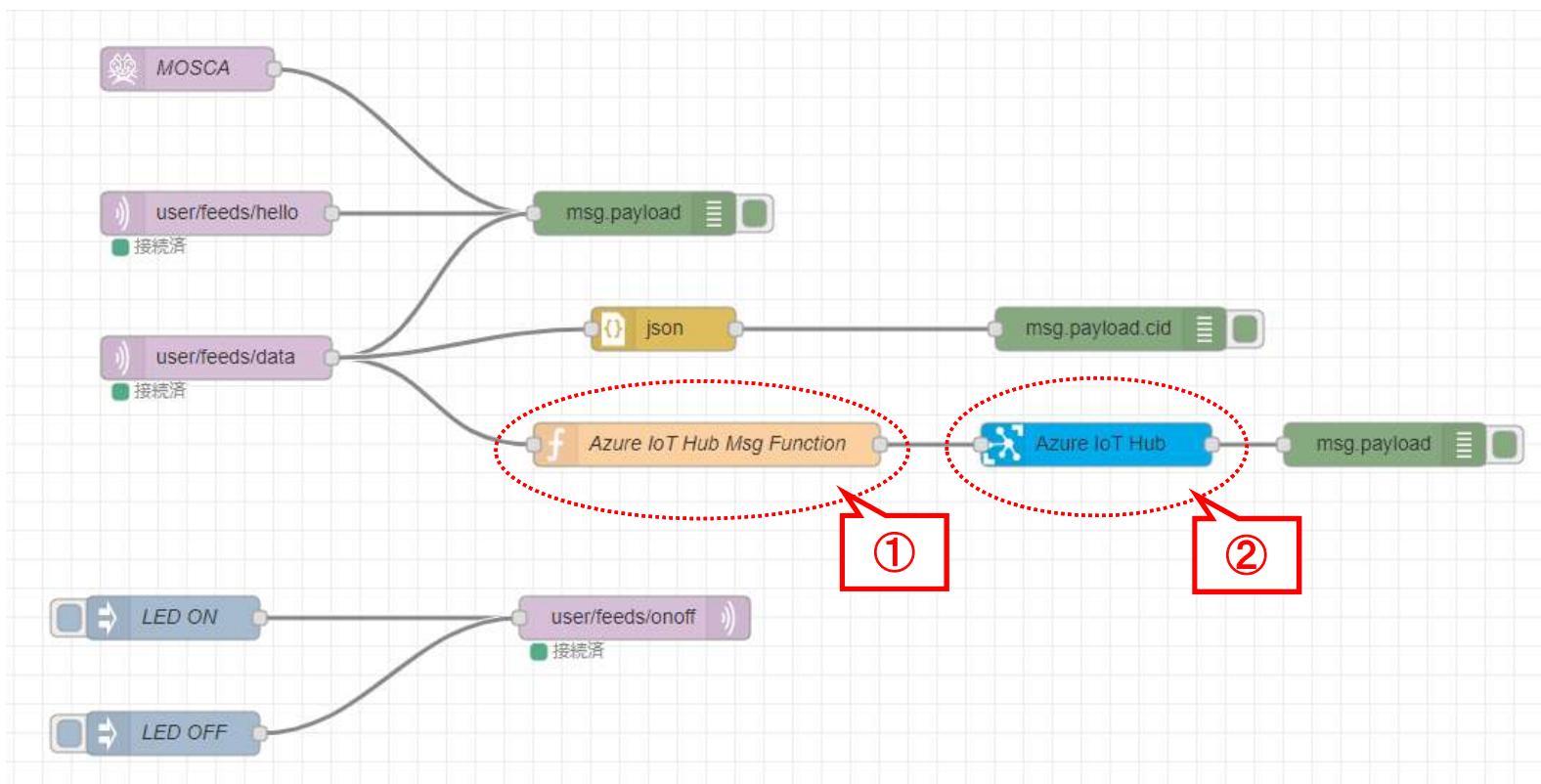
(5) ペースト



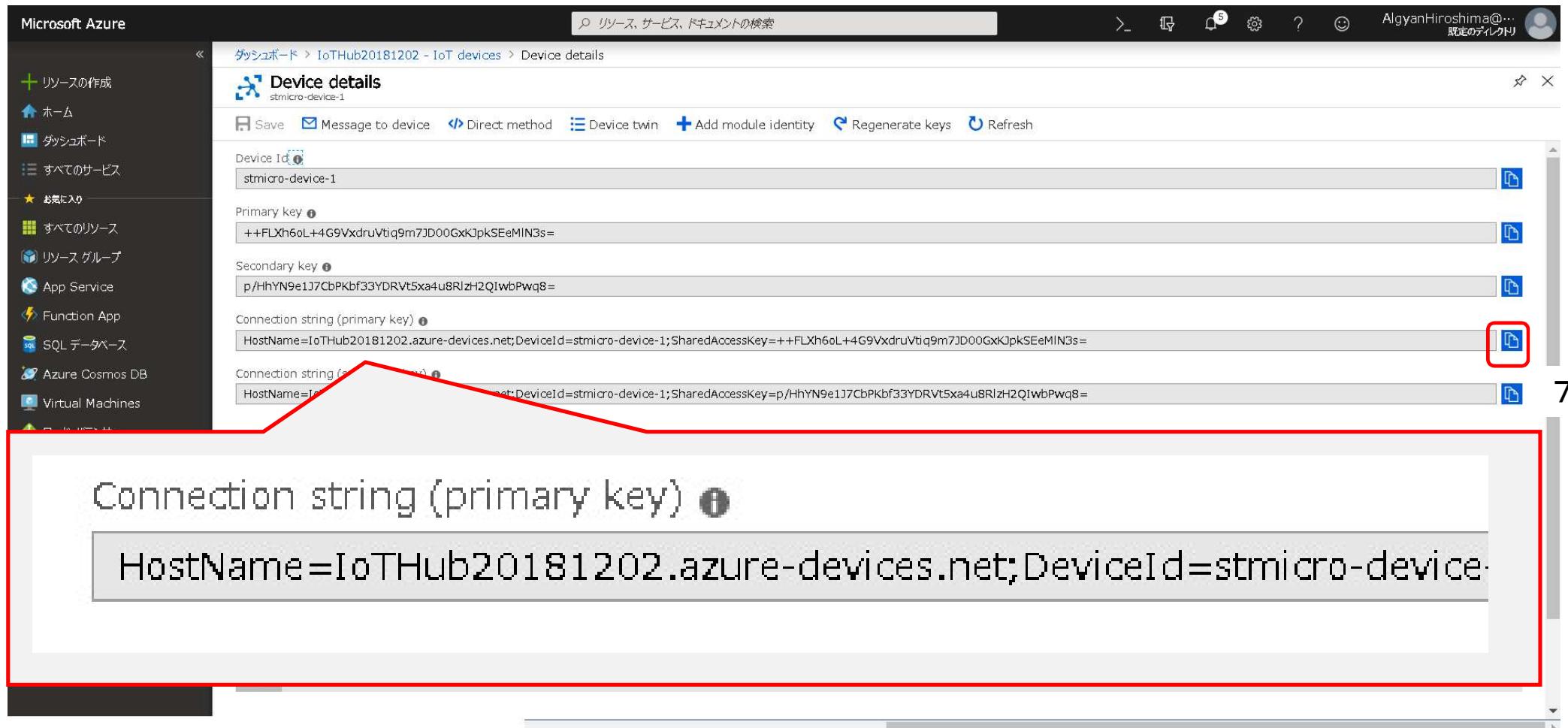
10-3 Azure-IoT-Hubの設定

以下の2つのノードを変更します

- ① Azure IoT Hub Msg Function
- ② Azure IoT Hub



(1) IoT Hub内のIoT devices接続設定文字列をコピーする



The screenshot shows the Microsoft Azure IoT Hub Device details page for a device named "stmicro-device-1". The page includes fields for Device Id (stmicro-device-1), Primary key, Secondary key, Connection string (primary key), and Connection string (secondary key). A red box highlights the "Connection string (primary key)" section, and a red arrow points from it to a larger callout box at the bottom. The callout box contains the connection string: "HostName=IoTHub20181202.azure-devices.net;DeviceId=stmicro-device-1;SharedAccessKey=++FLXh6oL+4G9VxdruVtiq9m7JD00GxKJpkSEeMIN3s=". A red box also surrounds the copy icon next to the primary key connection string.

Microsoft Azure

リソースの作成 リソース、サービス、ドキュメントの検索

ホーム ダッシュボード

すべてのサービス

お気に入り

すべてのリソース

リソース グループ

App Service

Function App

SQL データベース

Azure Cosmos DB

Virtual Machines

IoT Hub

ダッシュボード > IoTHub20181202 - IoT devices > Device details

Device details
stmicro-device-1

Save Message to device Direct method Device twin Add module identity Regenerate keys Refresh

Device Id: stmicro-device-1

Primary key: ++FLXh6oL+4G9VxdruVtiq9m7JD00GxKJpkSEeMIN3s=

Secondary key: p/HhYN9e1J7CbPKbf33YDRVt5xa4u8Rlzh2QIwbPwq8=

Connection string (primary key): HostName=IoTHub20181202.azure-devices.net;DeviceId=stmicro-device-1;SharedAccessKey=++FLXh6oL+4G9VxdruVtiq9m7JD00GxKJpkSEeMIN3s=

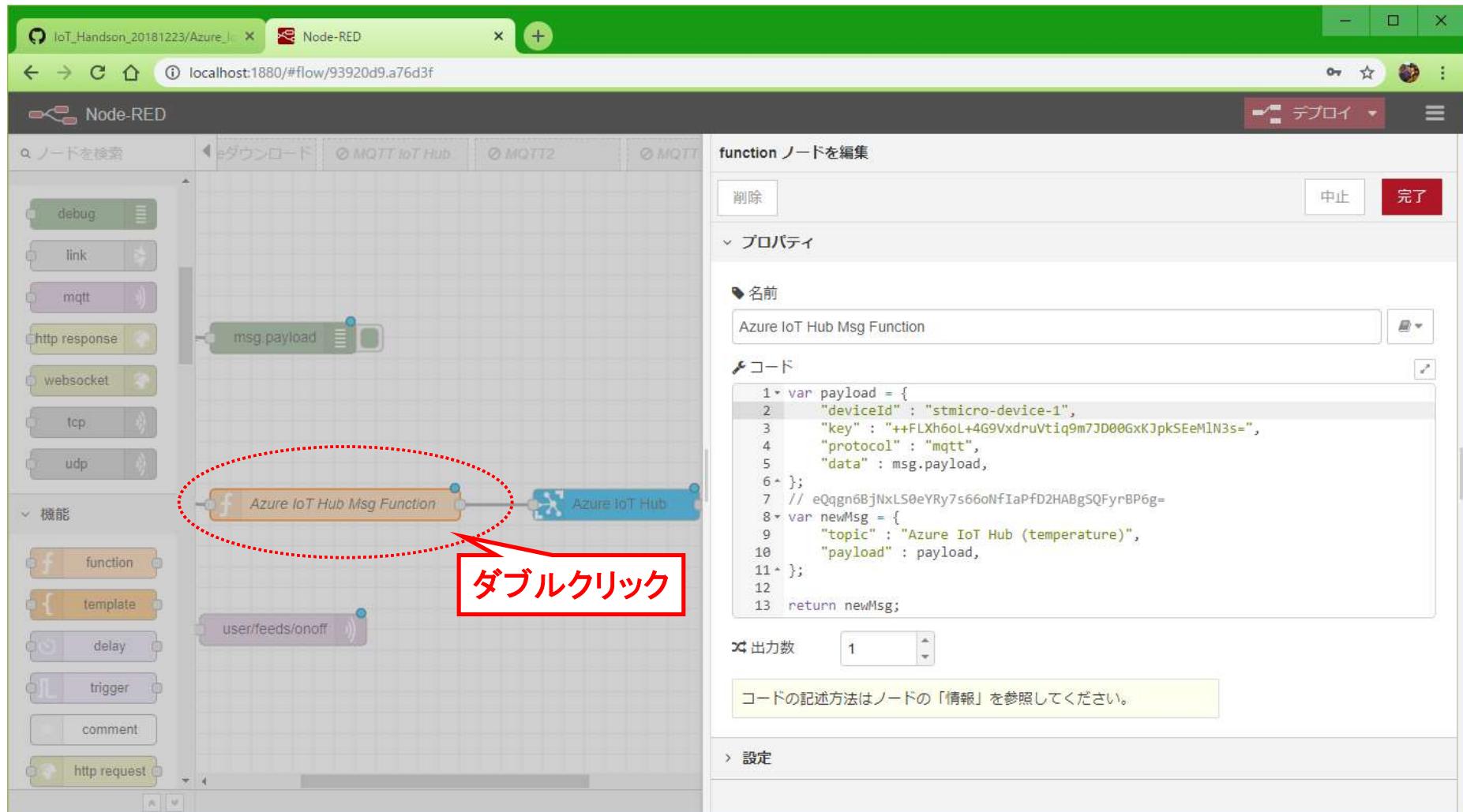
Connection string (secondary key): HostName=IoTHub20181202.azure-devices.net;DeviceId=stmicro-device-1;SharedAccessKey=p/HhYN9e1J7CbPKbf33YDRVt5xa4u8Rlzh2QIwbPwq8=

Connection string (primary key):
HostName=IoTHub20181202.azure-devices.net;DeviceId=stmicro-device-1;SharedAccessKey=++FLXh6oL+4G9VxdruVtiq9m7JD00GxKJpkSEeMIN3s=

Connection string (secondary key):
HostName=IoTHub20181202.azure-devices.net;DeviceId=stmicro-device-1;SharedAccessKey=p/HhYN9e1J7CbPKbf33YDRVt5xa4u8Rlzh2QIwbPwq8=

7

(2) Azure IoT Hub Msg Function の編集



(3) deviceId と key を自分の Azure アカウントの IoT Hub に置き換えます

コード

```
1 var payload = {  
2     "deviceId" : "stmicro-device-1",  
3     "key" : "++FLXh6oL+4G9VxdruVtiq9m7JD00GxKJpkSEeMlN3s=",  
4     "protocol" : "mqtt",  
5     "data" : msg.payload,  
6 };  
7 // eQqgn6BjNxLS0eYRy7s66oNfIaPfd2HABgSQFyrBP6g=  
8 var newMsg = {  
9     "topic" : "Azure IoT Hub (temperature)",  
10    "payload" : payload,  
11};  
12  
13
```

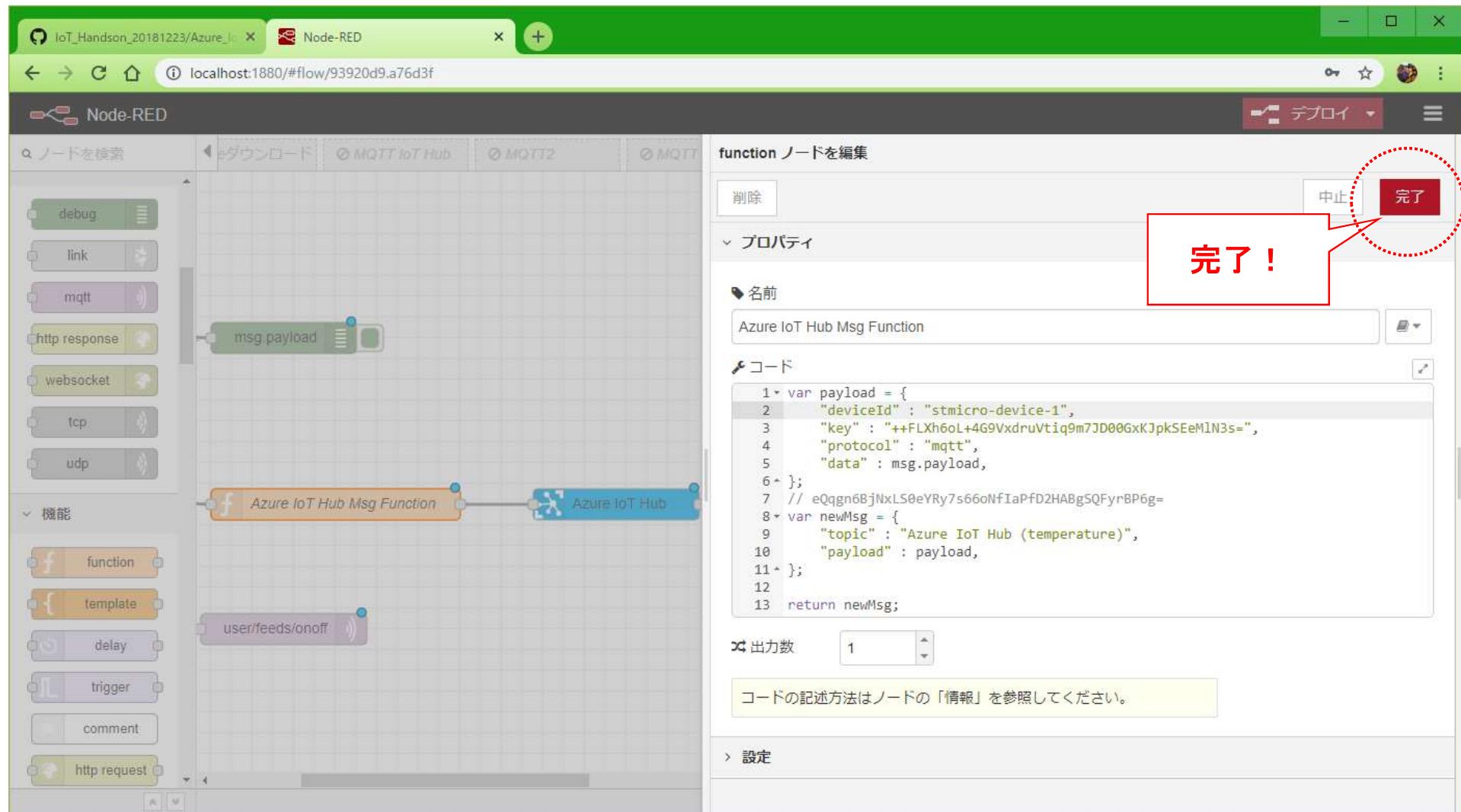
① DeviceId
を設定する

① SharedAccessKey
を設定する

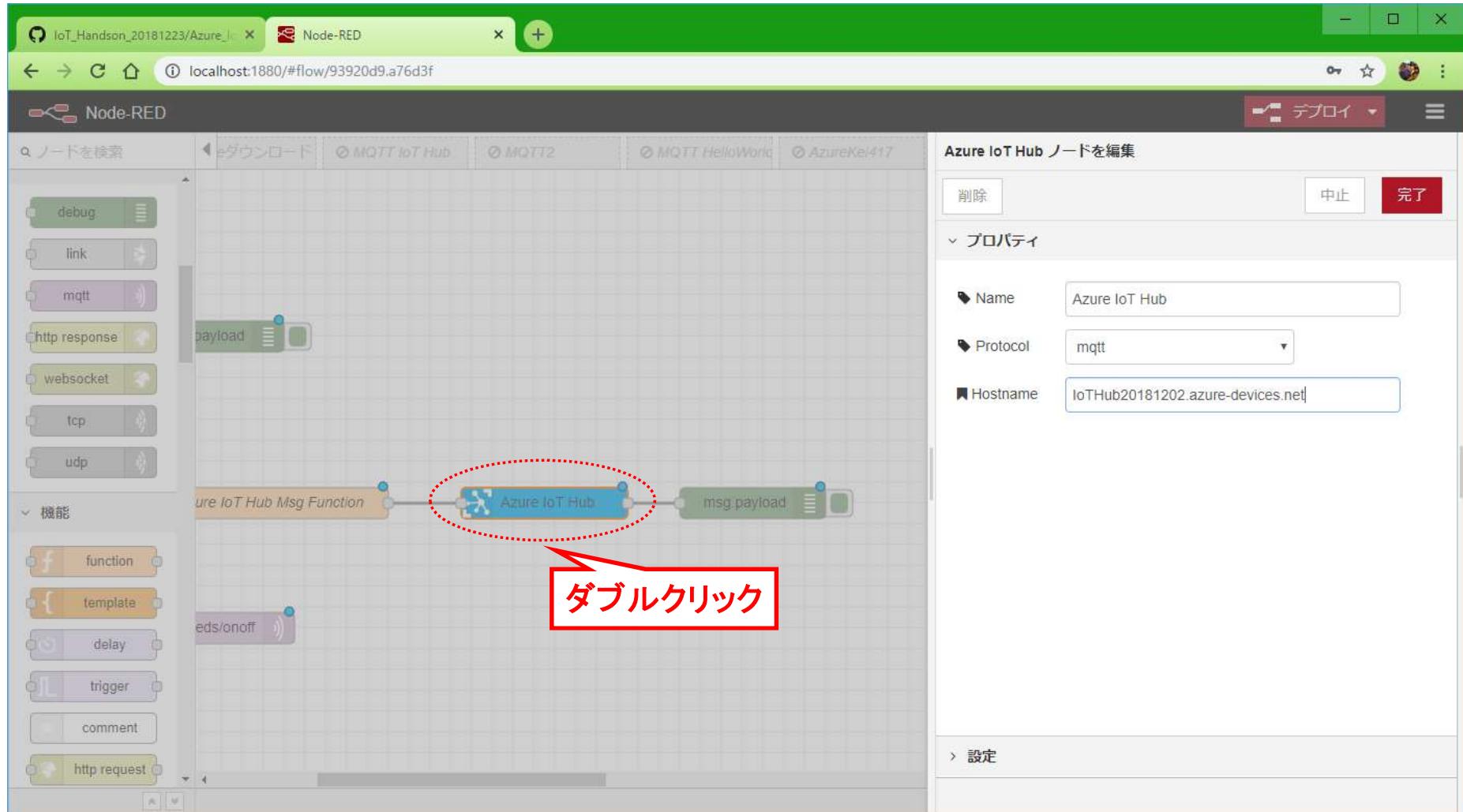
```
DeviceId=stmicro-device-1;SharedAccessKey=++FLXh6oL+4G9VxdruVtiq9m7JD00GxKJpkSEeMlN3s=
```

出

(4) 完了



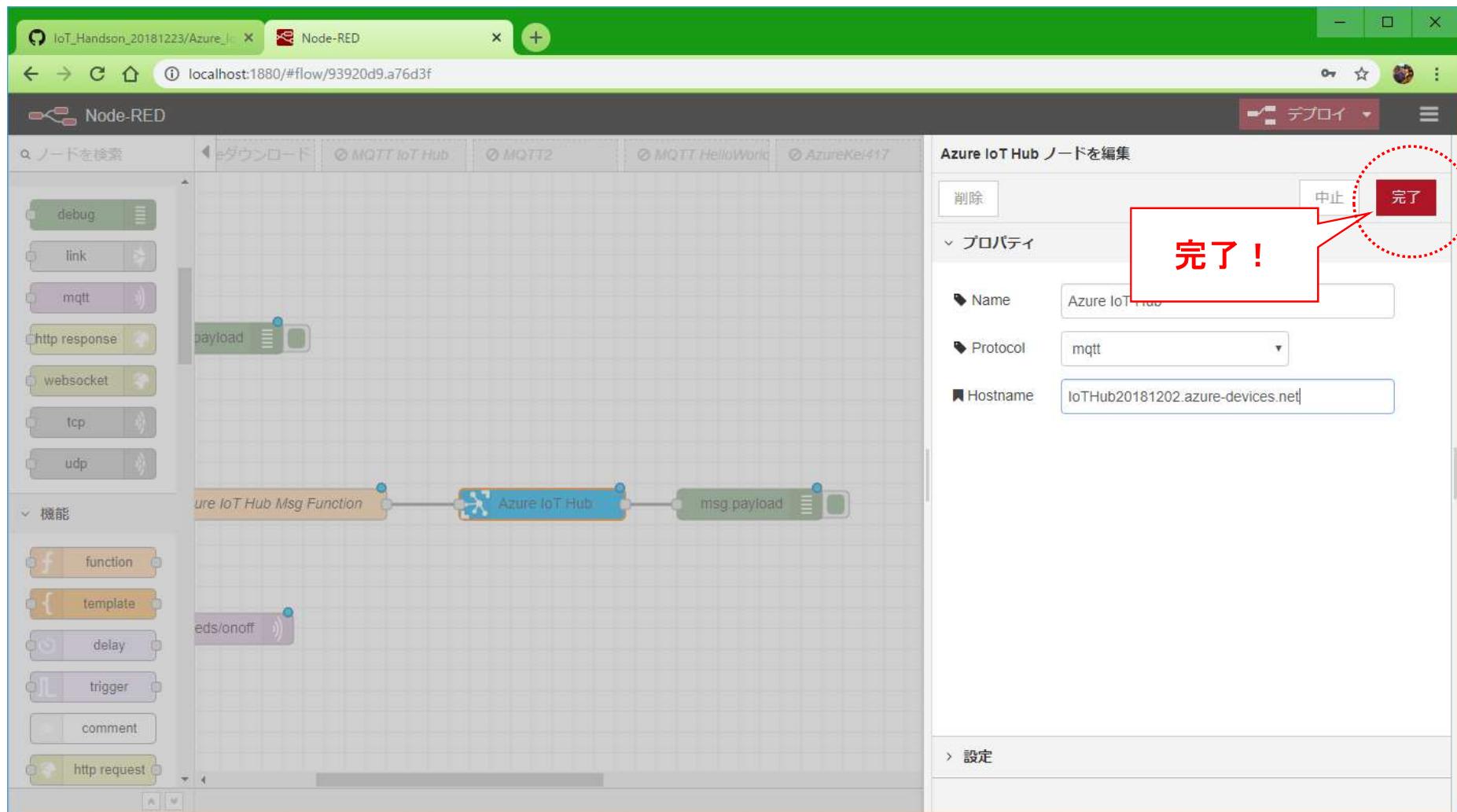
(5) Azure IoT Hub の編集



(6) HostNameを自分の Azure アカウントの IoTHubに置き換えます



(7) 完了



10-4 動作確認

(1) Azure のストレージを確認してみよう！

The screenshot shows the Microsoft Azure portal with the following details:

- Left Sidebar:** Lists various Azure services: リソースの作成, ホーム, ダッシュボード, すべてのサービス, お気に入り, すべてのリソース, リソース グループ, App Service, Function App, SQL データベース, Azure Cosmos DB, Virtual Machines, ロード バランサー, ストレージ アカウント, 仮想ネットワーク, Azure Active Directory, モニター, Advisor, セキュリティセンター, コストの管理と請求, ヘルプとサポート.
- Top Bar:** Includes a search bar (リソース、サービス、ドキュメントの検索), navigation icons (戻る、戻す、ヘルプ), and user information (AlgyanHiroshima@...).
- Current View:** IoT Kit Hands-on 起動中 (IoTHoLRG). The main content area shows the following table of resources:

リソース	種類	場所
IoTHub20181202	IoT Hub	西日本
egstorage20181202	ストレージ アカウント	西日本

(2) Blob の中に日付ごとに格納されます

The screenshot shows the Azure portal interface for managing a storage account named 'egstorage20181202'. The left sidebar lists various services like Home, Dashboard, and Storage Accounts. The main content area displays the storage account's properties. A red box highlights the 'Blob' service section under 'サービス' (Services), which describes it as a REST-based object storage for unstructured data. Below this, there are sections for monitoring and connectivity settings.

リソース、サービス、ドキュメントの検索

AlgyanHiroshima@... 指定のディレクトリ

Microsoft Azure

ダッシュボード > egstorage20181202

egstorage20181202
ストレージ アカウント

リソースの作成

ホーム

ダッシュボード

すべてのサービス

お気に入り

すべてのリソース

リソース グループ

App Service

Function App

SQL データベース

Azure Cosmos DB

Virtual Machines

ロード バランサー

ストレージ アカウント

仮想ネットワーク

Azure Active Directory

モニター

Advisor

セキュリティ センター

コストの管理と請求

ヘルプとサポート

リソースの作成

概要

アクティビティ ログ

アクセス制御 (IAM)

タグ

問題の診断と解決

イベント

Storage Explorer (プレビュー)

設定

アクセス キー

CORS

構成

暗号化

Shared Access Signature

ファイアウォールと仮想ネットワーク

Advanced Threat Protection

オーバーレイ

ロック

Automation スクリプト

Blob service

blob

リソース グループ (変更)
IoTHoLRG

状態
プライマリ: 利用可能、セカンダリ: 利用可能

場所
西日本, 東日本

サブスクリプション (変更)
無料試用版

サブスクリプション ID
dbb2f7a-3923-4343-b937-ef41789a2501

タグ (変更)
タグを追加するにはここをクリック

サービス

BLOB
非構造化データの REST ベースのオブジェクト ストレージ

Azure AD プレビューを使用してデータを探索します

詳細情報

監視

次に指定する直近の期間のデータを表示する:

1 時間	6 時間	12 時間	1 日間	7 日
------	------	-------	------	-----

表示するデータ:
アカウント

送信の合計数

イングレスの合計

(3) データの中を確認してみましょう

The screenshot shows the Microsoft Azure portal interface. On the left, the navigation menu includes options like 'リソースの作成', 'ホーム', 'ダッシュボード', 'すべてのサービス', 'お気に入り', 'すべてのリソース', 'リソース グループ', 'App Service', 'Function App', 'SQL データベース', 'Azure Cosmos DB', 'Virtual Machines', 'ロード バランサー', 'ストレージ アカウント', '仮想ネットワーク', 'Azure Active Directory', 'モニター', 'Advisor', 'セキュリティ センター', 'コストの管理と請求', and 'ヘルプとサポート'.

The main area displays the contents of a blob storage container named 'iot-rawdata'. The container path is 'iot-rawdata / IoTHub20181202 / 01 / 2018 / 12 / 18 / 13'. The blob itself is titled 'IoTHub20181202/01/2018/12/18/13/08'. The blob's properties include 'オブジェクト名: IoTHub20181202/01/2018/12/18/13/08' and 'オブジェクトの拡張子: JSON'.

The blob's content is displayed as a JSON object:

```
1   >r"\\"cid\":26, \"temperature\": 28.60, \"humidity\":46, \"pressure\":1025.26, \"MagX\":100, \"MagY\":100, \"MagZ\":100, \"lat\":37.775, \"lon\":-122.425, \"elevation\":100, \"angle\":100, \"roll\":100, \"pitch\":100, \"yaw\":100, \"ip\":192.168.1.100, \"mac\":\"00:0C:29:AB:CD:EF\", \"model\":\"esp32\", \"firmware\":\"v1.0\", \"status\":\"OK\", \"last_update\":\"2018-12-18T13:08:00Z\""}"
```

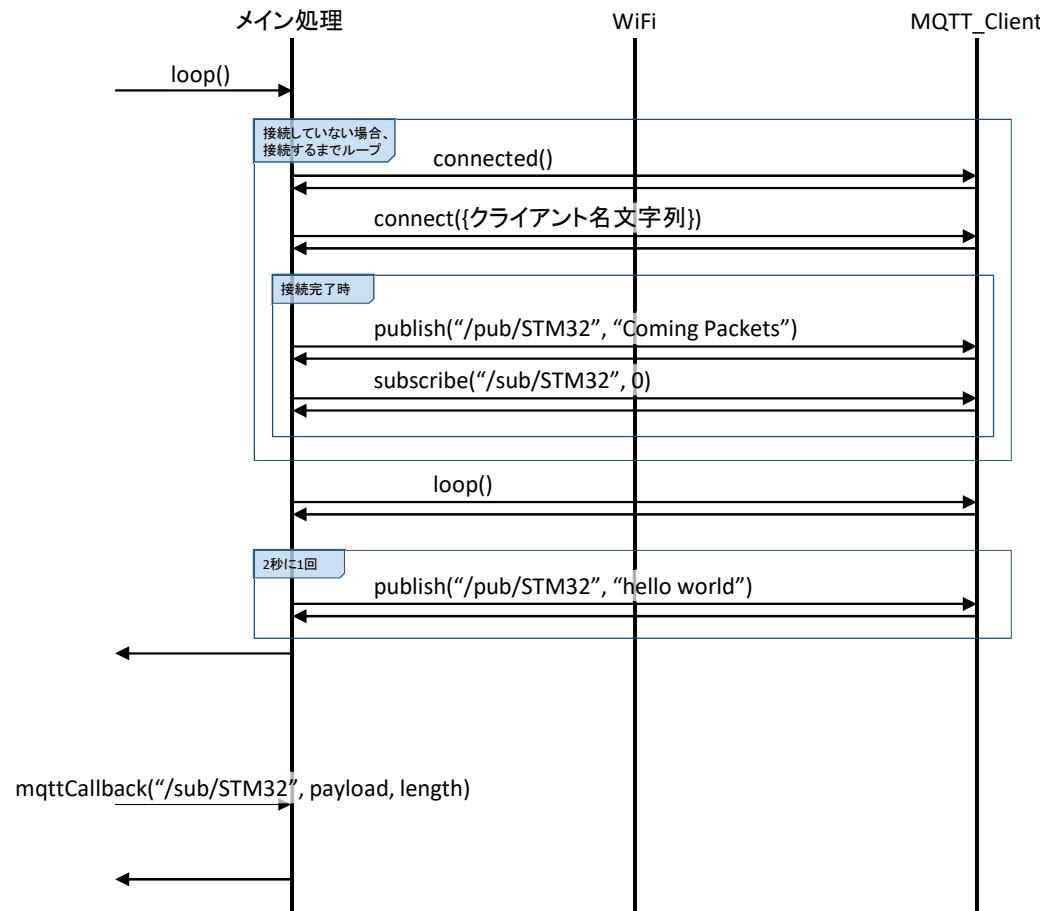
A red box highlights the first line of the JSON content.

おつかれさまでした！

ご清聴ありがとうございました。

備考

STM32側処理シーケンス(loop, callback)

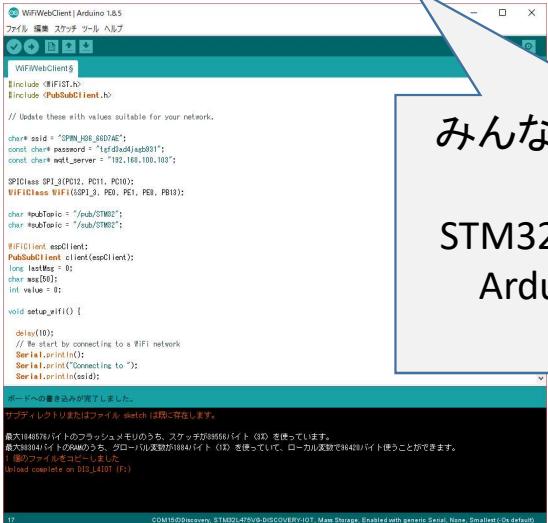


Arduino IDEの設定覚え書き

- C:\Users\{ユーザ名}\AppData\Local\Arduino15\staging\libraries
追加ライブラリが保存されている

STM32 – 開発環境

- Keil MDK-ARM (有料)
- IAR EWARM (有料)
- GCC-based IDEs including free SW4STM32 from AC6 + STM32CubeL4 (無料)
- ARM mbed online (Webベース 無料)
- Arduino Genuino (無料)



```
WiFiWebClient | Arduino 1.8.5
...
#include "WiFi.h"
#include "WiFiClient.h"

// Update these with values suitable for your network.

char ssid = "OPEN-100";
const char password = "1234567890";
const char mqtt_server = "192.168.100.103";

SPIClass SPI_1(PC12, PC11, PC10);
WiFiClass WiFi(SPI_1, PE0, PE1, PE2, PE3);

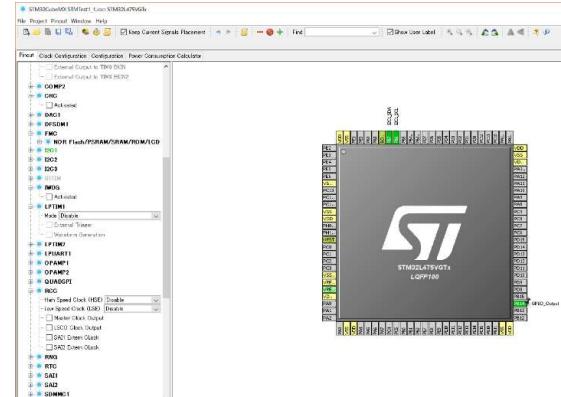
char rootTopic = "/pub/STM32";
char subTopic = "/sub/STM32";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[256];
int value = 0;

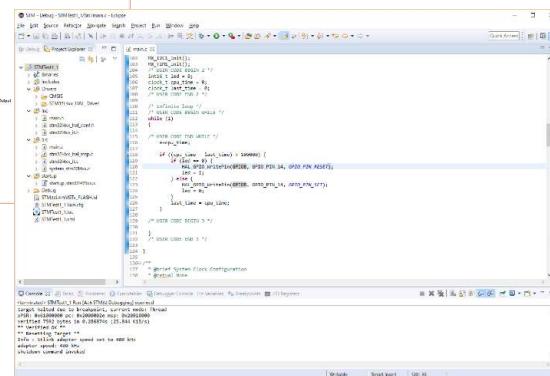
void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.print("Connecting to ");
  Serial.print(ssid);
  Serial.print(" at ");
  Serial.print(subTopic);
  Serial.print(" on ");
  Serial.print(rootTopic);
}

void setup() {
  // Initialize serial
  Serial.begin(115200);
  // Start WiFi
  setup_wifi();
}
```

みんな大好き Arduino !
STM32 Discovery Kitは、
Arduino Uno互換で
機能もMAX



個人的にSW4+STM32Cube
がおすすめ



STM32 - Arduino

- STM32L475 は Arduino互換で動作することができます。
- STM32duinoで公開されているものがあればそちらのライブラリを使用した方が無難です。

https://github.com/stm32duino/Arduino_Core_STM32/tree/master/libraries

- Arduinoでも機能毎に使用するライブラリがいろいろと公開されているので、必要なライブラリをインポートします。

<http://www.musashinodenpa.com/arduino/ref/>

- 機能やPIN配や機能数が純正Arduinoと異なるのでライブラリのピン指定に注意が必要です。

STM32 Discovery Kit

- STM32 Discovery Kit のデータシートをみてみよう！

https://www.st.com/content/ccc/resource/technical/document/user_manual/group0/b1/b8/7a/f2/f7/8d/4b/6b/DM00347848/files/DM00347848.pdf/jcr:content/translations/en.DM00347848.pdf

ページ	内容	見るべきところ
6	機能一覧	STM32の機能を確認しよう
9	ハードウェア・ブロックダイアグラム	マイコン本体と周辺装置(センサー)等のインターフェースを確認しよう
10 – 11	STM32のレイアウト図	STM32を実際に見比べてみよう
29	I2C アドresse一覧	各種センサーのI2Cアドレスを確認しよう。I2Cに関しては別途説明します。
30-31	コネクタ・レイアウト	Arduino シールド互換の威力を確認！
37-40	ピン配	ピン番号と機能を確認
42-52	スケマティック	マイコン本体、ボードの回路とインターフェースのピン配を確認