

# IoTデバイス活用マニュアル

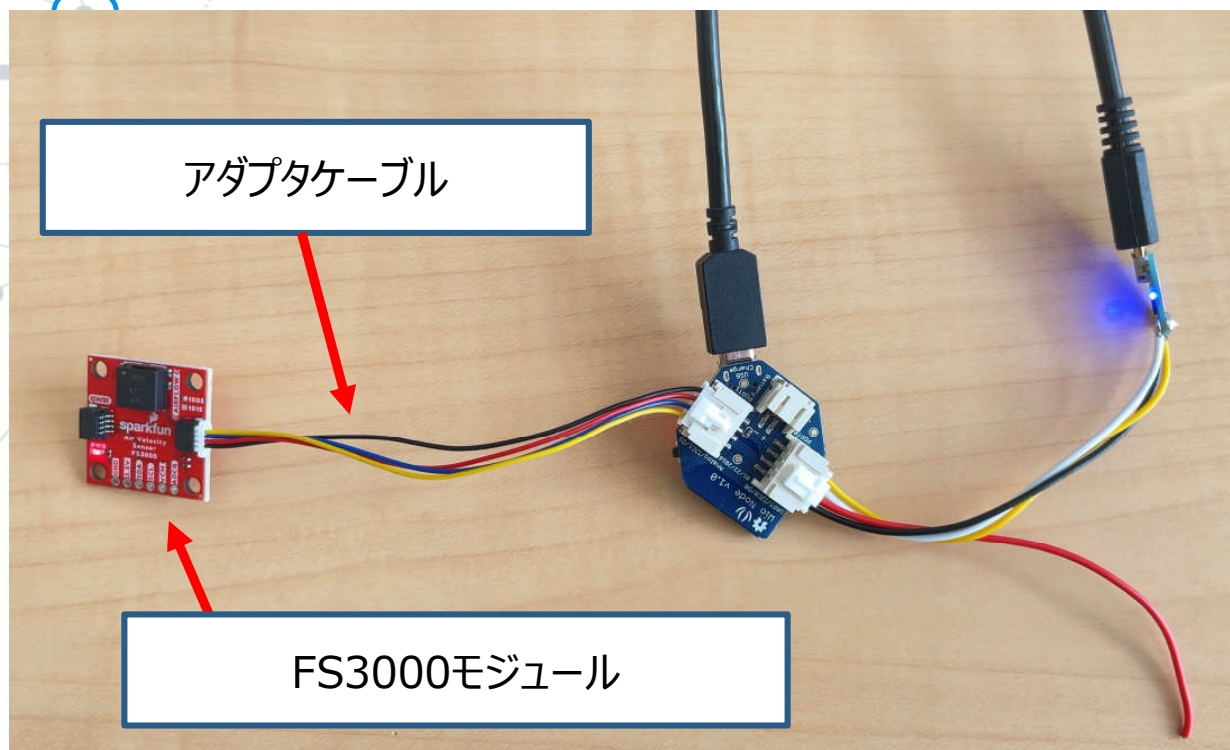
センサ種類：風速計

型番：FS3000

ストーリー：風速が一定以上高まった時アラートメールを送信する



改訂記録：  
2022/2/11 初版 作成 門奈



## Qwiic - FS3000搭載 空気速度センサモジュール

ルネサス製のFS3000を搭載し、Qwiicコネクタを備えた空気速度センサ

### 仕様

- I<sup>2</sup>C アドレス : 0x28
- 風速 : 0 - 7.23 m/秒 (0 - 16.17mph)
- 精度 : フルスケール流量レンジの5 %
- 解像度 : 12 bit
- 入力電圧 : 2.7 - 3.3 V
- 平均消費電流 : 10 mA

### 製品情報

- Qwiic - FS3000搭載 空気速度センサモジュール → <https://www.switch-science.com/catalog/7467/>
- GROVEアダプタケーブル → <https://www.switch-science.com/catalog/5327/>



## 全体の手順

1. Arduino IDEへのプログラムの読み込みと変更
2. FS3000デバイスのライブラリーの読み込みと修正
3. センサーの動作確認
4. Stream Analytics クエリの修正
5. PowerBIの動作確認
6. 風速が所定値よりも高くなったらアラームを発報する



# 1.Arduinoのプログラムの読み込み と変更





## プログラムの読み込み、WiFi 設定、デバイスキーの書き換え

1. 本FS3000用のプログラムをArduinoIDEへ読み込む
2. WiFi の SSID、パスワードを入力する
3. Azure IoT Hub で発行されたデバイスのプライマリ文字列を該当箇所に入力する



## 2. FS3000デバイスのライブラリの 読み込みと修正



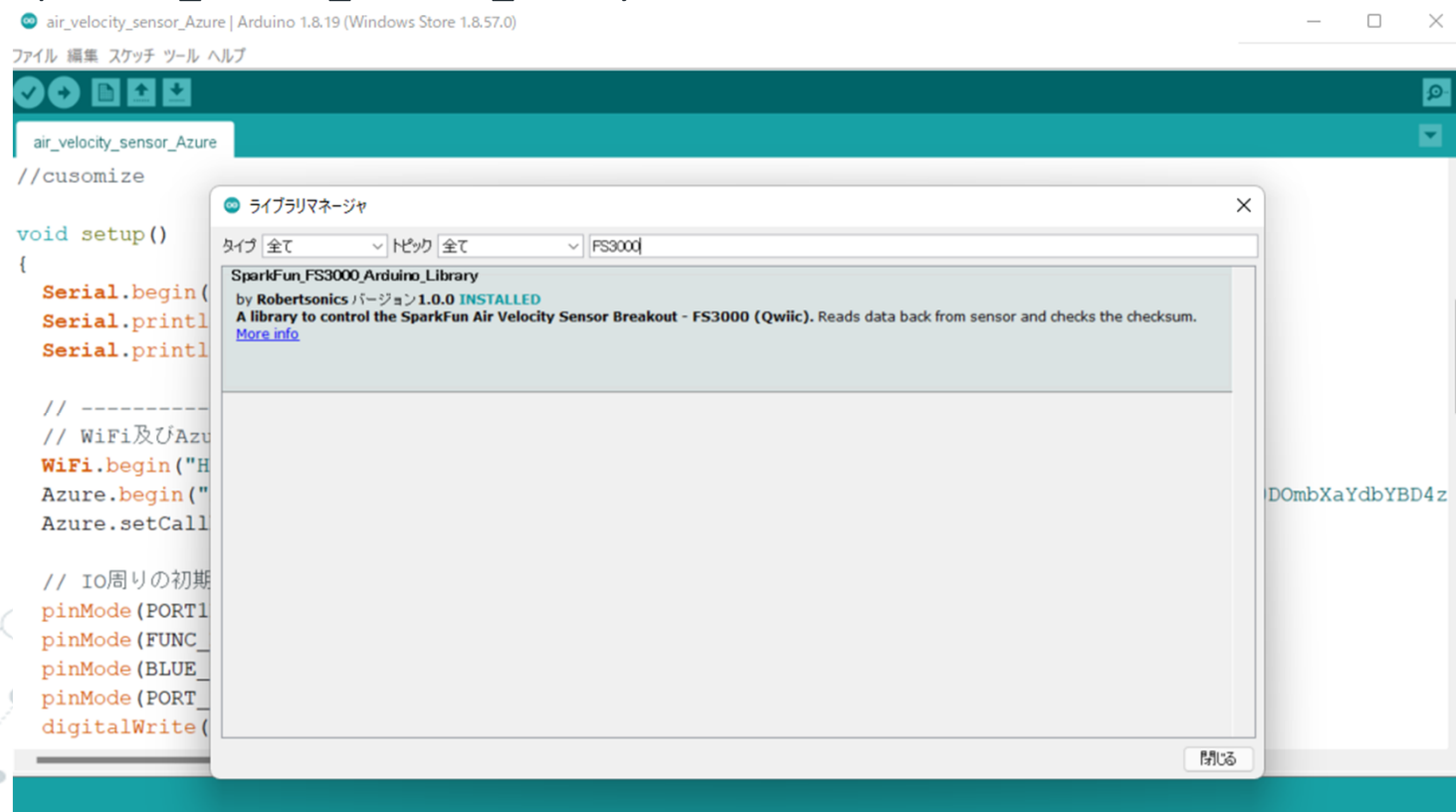
# Arduino IDE へFS3000デバイスのライブラリーを読み込む

- ・ライブラリーを読み込む手順

「ツール」→「ライブラリーを管理」

検索枠で「FS3000」を検索（下図参照）

「SparkFun\_FS3000\_Arduino\_Library」をインストールする

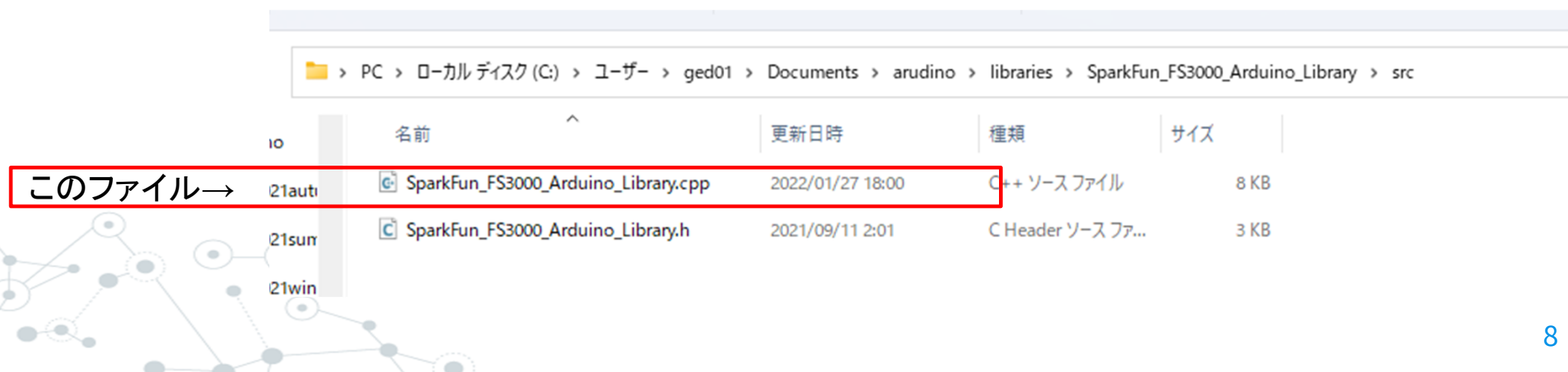


## FS3000デバイスのライブラリーの修正

- Arduinoのライブラリーが保存されているフォルダを探す（下図参照）
- 「libraries」フォルダを開く
- 「SparkFun\_FS3000\_Arduino\_Library」フォルダを開く
- 「src」フォルダを開く
- 「SparkFun\_FS3000\_Arduino\_Library.cpp」ファイルを編集する

ファイルのところで右クリックし、プログラムから開くを選択し、テキストが編集できるアプリを何でも良いので選ぶ

42行目と51行目の「boolean」を「bool」に書き換える（次ページ参照）





but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

\*\*\*\*\*  
\*\*\*/  
\*\*\*\*\*

```
#include "Arduino.h"  
#include "SparkFun_FS3000_Arduino_Library.h"  
#include <Wire.h>
```

```
FS3000::FS3000()  
{  
  
}
```

```
//Initializes the sensor (no settings to adjust)  
//Returns false if sensor is not detected
```

42行目→ `bool FS3000::begin(TwoWire &wirePort)`

```
{  
    i2cPort = &wirePort;  
    if (isConnected() == false) // Check for sensor by verifying ACK response  
        return (false);  
    return (true); //We're all setup!  
}
```

51行目→ `//Returns true if I2C device ack's`

```
bool FS3000::isConnected()  
{  
  
}
```

```
    i2cPort->beginTransmission((uint8_t)FS3000_DEVICE_ADDRESS);  
    if (i2cPort->endTransmission() != 0)  
        return (false); //Sensor did not ACK  
    return (true);  
}
```

「SparkFun\_FS3000\_Arduino\_Library.cpp」ファイルを編集する

42行目と51行目の「boolean」を「bool」に  
書き換える

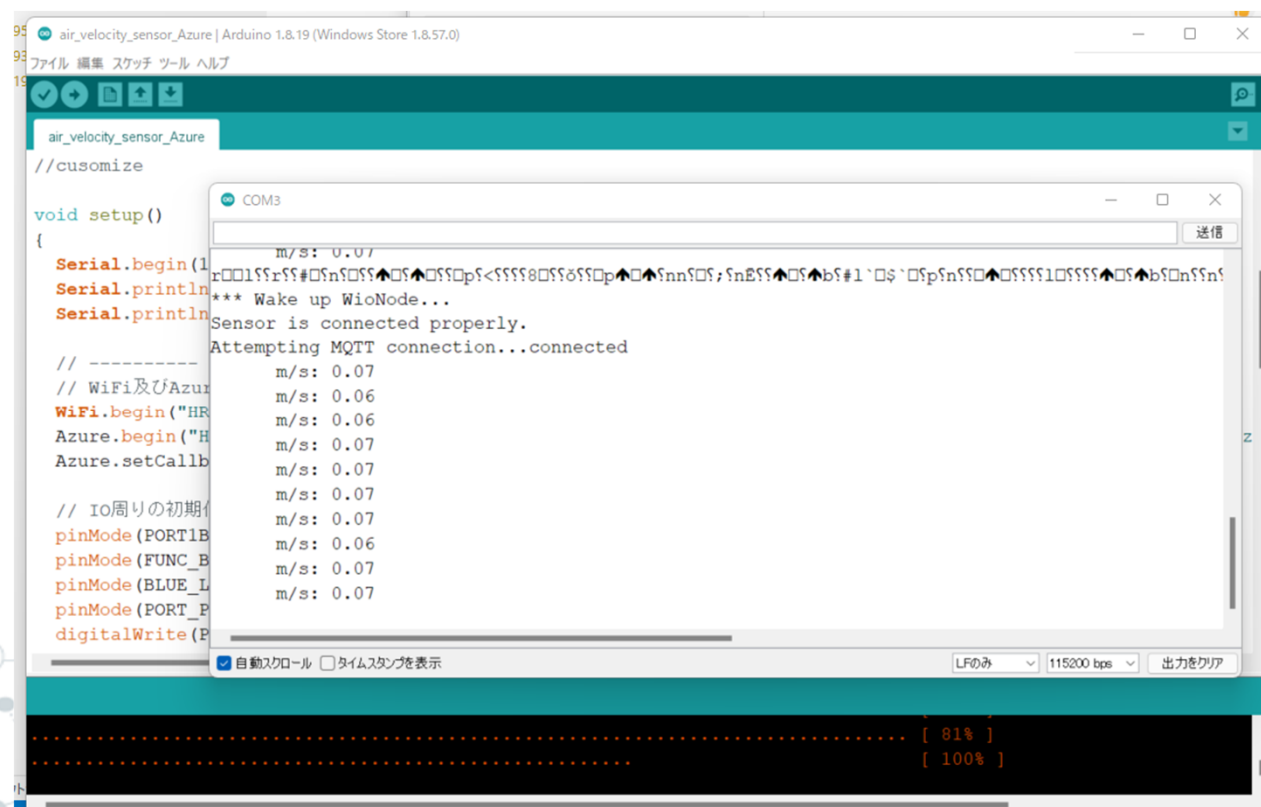


# 3. センサーの動作確認



## センサーの動作確認

- Arduino IDEのシリアルモニタでデータが送付されていることを確認してください
- センサーへ息を吹きかけると数値が変化することを確認してください
- 現在は、1秒（ delay(1000) ）毎の出力になっています。任意の数値へ修正ください
- WioNodeの青ランプがサンプリング時に点灯するようにしています



```
air_velocity_sensor_Azure | Arduino 1.8.19 (Windows Store 1.8.57.0)
//customize

void setup()
{
  Serial.begin(115200);
  Serial.println("*** Wake up WioNode...");
  Serial.println("Sensor is connected properly.");
  Serial.println("Attempting MQTT connection...connected");

  // -----
  // WiFi及びAzure
  WiFi.begin("HR");
  Azure.begin("H");
  Azure.setCallback(callback);

  // IO周りの初期化
  pinMode(PORT1B, OUTPUT);
  pinMode(FUNC_BUTTON, INPUT);
  pinMode(BLUE_LED, OUTPUT);
  pinMode(PORT_P, OUTPUT);
  digitalWrite(PORT_P, LOW);
}

void loop()
{
  m/s: 0.07
  m/s: 0.06
  m/s: 0.06
  m/s: 0.07
  m/s: 0.07
  m/s: 0.07
  m/s: 0.07
  m/s: 0.06
  m/s: 0.07
  m/s: 0.07

  [ 81% ]
  [ 100% ]
```



## 4. Stream Analyticsのクエリ変更



## クエリの説明①

- ・クエリに「params.sensor as sensor,」を追加してください

```
SELECT
  Dev as device,
  DATEADD(hour, 9, EventEnqueuedUtcTime) as time,
  EventEnqueuedUtcTime as utctime,
  params.sensor as sensor,
  params.espvalue as value
INTO
  outputpowerbi
FROM
  inputiothub

SELECT
  Dev as device,
  DATEADD(hour, 9, EventEnqueuedUtcTime) as time,
  EventEnqueuedUtcTime as utctime,
  params.sensor as sensor,
  params.espvalue as value
INTO
  outputcosmosdb
FROM
  inputiothub
```

入力のプレビュー    テスト結果

'outputpowerbi' の 50 行を表示しています。

device	time	utctime	sensor	value
"95monna03"	"2022-02-11T11:17:51.8550000Z"	"2022-02-11T02:17:51.855..."	"air"	0.06132
"95monna03"	"2022-02-11T11:17:50.7610000Z"	"2022-02-11T02:17:50.761..."	"air"	0.06132
"95monna03"	"2022-02-11T11:17:49.6670000Z"	"2022-02-11T02:17:49.667..."	"air"	0.05921
"95monna03"	"2022-02-11T11:17:48.5580000Z"	"2022-02-11T02:17:48.558..."	"air"	0.06132
"95monna03"	"2022-02-11T11:17:47.4480000Z"	"2022-02-11T02:17:47.448..."	"air"	0.06344
"95monna03"	"2022-02-11T11:17:46.3390000Z"	"2022-02-11T02:17:46.339..."	"air"	0.05709

✓ 成功



# 5. PowerBIでのレポートの作成

測定値をモニタする

## データセットを開く

Power BI FactoryScientist受講生サイト

新しい外観をオンにする 検索

ファイル ビュー 閲覧表示 モバイルレイアウト 質問する 探索 テキスト ボックス 図形 ボタン ビジュアル対話 更新 このページを複製する 保存

ホーム  
お気に入り  
最近  
アプリ  
自分と共有  
詳細

ワークスペース  
FactoryScientist受...  
ダッシュボード  
確認ダッシュボード  
レポート  
稼働状況マップ  
ブック  
データセット  
fsdataset  
データフロー  
データフローがありません

データを取得

ページ 1

フィルター

検索

このページでのフィルター ...  
ここにデータ フィールド...

すべてのページでのフィル...  
ここにデータ フィールド...

視覚化

フィールド

検索

fstable  
device  
duration  
time

その他のビジュアルの取得  
ビジュアルをファイルからインポート  
視覚エフェクトの削除  
既定の視覚化の復元

ドリルダウン  
クロス レポート  
オフ  
すべてのフィルターを保持...  
オン  
ドリルスルー フィールド...

fsdatasetを選択する

## 「折れ線グラフ」を選択

The screenshot shows the Power BI 'マイワークスペース' (My Workspace) interface. The main canvas displays a line chart titled 'time および sensor による value' (value by time and sensor). The chart has a vertical axis labeled 'value' and a horizontal axis labeled 'time'. The legend indicates four data series: sensor (blue square), CO2 (green circle), humi (red triangle), and temp (purple diamond). The 'Visualizations' pane on the right shows the 'Line chart' icon selected. The 'Fields' pane on the right shows the 'fstable' table with fields: device, duration, sensor, time, and value. The 'value' field is highlighted with a red circle. A red arrow points from the 'value' field in the 'Fields' pane to the 'value' field in the 'Values' section of the 'Visualizations' pane.

④valueを「値」にドラッグ  
アンドドロップ