

○温湿度気圧センサを使う

1. I2C の準備

RaspberryPi では直接のピン操作も可能ですが、より高機能なインターフェースを利用するためには I2C を使うことが有効です。

○I2C とは

I2C は図 11 に示すようにオープンドレインの 2 線をクロックとデータとして利用し、バス構成の接続を実現します。I2C の動作はマスタ動作とスレーブ動作の 2 つに分かれ、それぞれの動作で送信と受信が可能です。基本的な動作は図 12 に示すように、マスタ側が主導権をもち、通信開始、アドレス送信、データ送受信、通信停止の 4 つの状態が順に遷移していきます。スレーブ側はマスタ側の遷移に合わせて、アドレス受信、データ送受信と遷移します。

RaspberryPi で I2C を使用するには `raspi-config` で `i2c` をアクティブにしておきます。

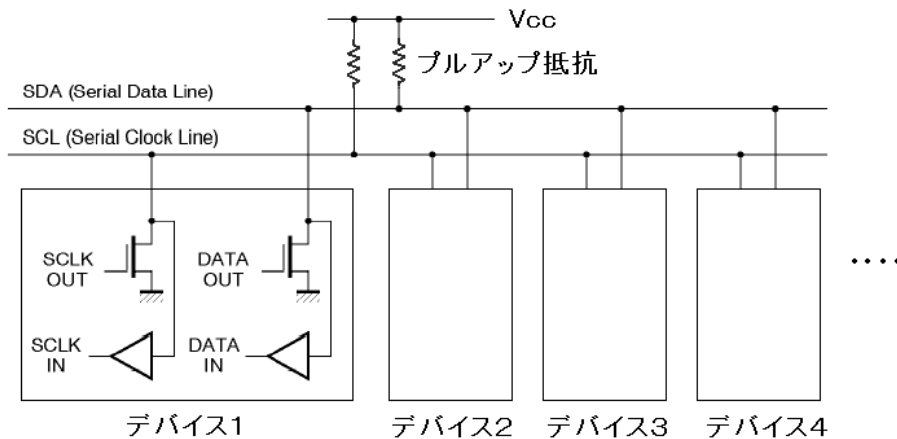


図 11 バス構成をとる I2C

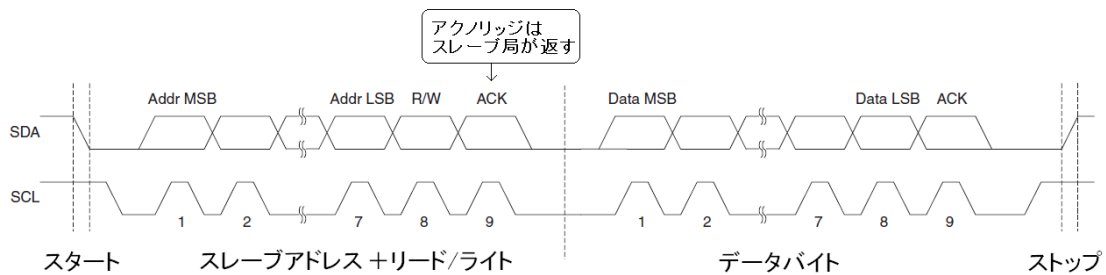


図 12 I2C の信号タイミング

○I2C の状態の確認

RaspberryPi では I2C の状態を `raspi-config` コマンドで確認、切り替えすることができます。

```
sudo raspi-config
```

とタイプして `raspi-config` を起動します。

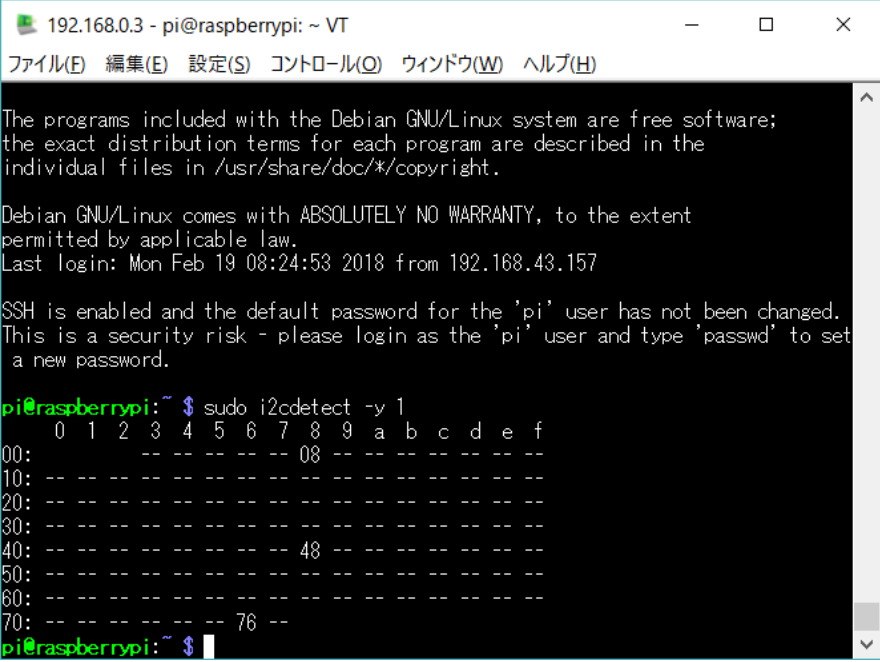
○I2C のツール

I2C の状態の確認には一般的な I2C のツールを使うこともできます。I2C のツールは以下のようにインストールします。

```
sudo apt-get install i2c-tools
```

ツールのインストールのあと以下のコマンドを実行すると、図 18 に示すようなアドレスマップが得られます。

```
sudo i2cdetect -y 1
```



```
192.168.0.3 - pi@raspberrypi: ~ VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Feb 19 08:24:53 2018 from 192.168.43.157

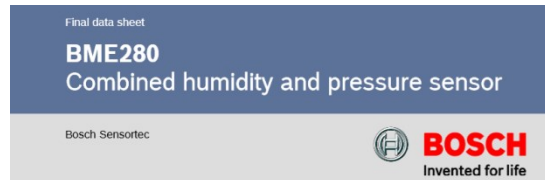
SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

pi@raspberrypi:~$ sudo i2cdetect -y 1
   0 1 2 3 4 5 6 7 8 9 a b c d e f
00: -- -- -- -- -- 08 -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- 48 -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- 76 -- -- -- -- --
pi@raspberrypi:~$
```

BME280 のほかに 2 つのデバイスが接続されている例
図 18 接続されている I2C デバイスのアドレスマップの例

2. BME280 と接続してみる

BME280 はボッシュ製の低消費電力で、温度、湿度、気圧が1つで測ることのできるセンサモジュールです。スペック、内部ブロック図を図14に示します。SPIとI2Cのインターフェースを切り替えて使用できます。



DIGITAL HUMIDITY, PRESSURE AND TEMPERATURE SENSOR

Key features

- Package 2.5 mm x 2.5 mm x 0.93 mm metal lid LGA
- Digital interface I²C (up to 3.4 MHz) and SPI (3 and 4 wire, up to 10 MHz)
- Supply voltage V_{DD} main supply voltage range: 1.71 V to 3.6 V
V_{DDIO} interface voltage range: 1.2 V to 3.6 V
- Current consumption 1.8 μA @ 1 Hz humidity and temperature
2.8 μA @ 1 Hz pressure and temperature
3.6 μA @ 1 Hz humidity, pressure and temperature
0.1 μA in sleep mode
- Operating range -40...+85 °C, 0...100 % rel. humidity, 300...1100 hPa
- Humidity sensor and pressure sensor can be independently enabled / disabled
- Register and performance compatible to Bosch Sensortec BMP280 digital pressure sensor
- RoHS compliant, halogen-free, MSL1

Key parameters for humidity sensor¹

- Response time ($\tau_{63\%}$) 1 s
- Accuracy tolerance ±3 % relative humidity
- Hysteresis ±1% relative humidity

Key parameters for pressure sensor

- RMS Noise 0.2 Pa, equiv. to 1.7 cm
- Offset temperature coefficient ±1.5 Pa/K, equiv. to ±12.6 cm at 1 °C temperature change

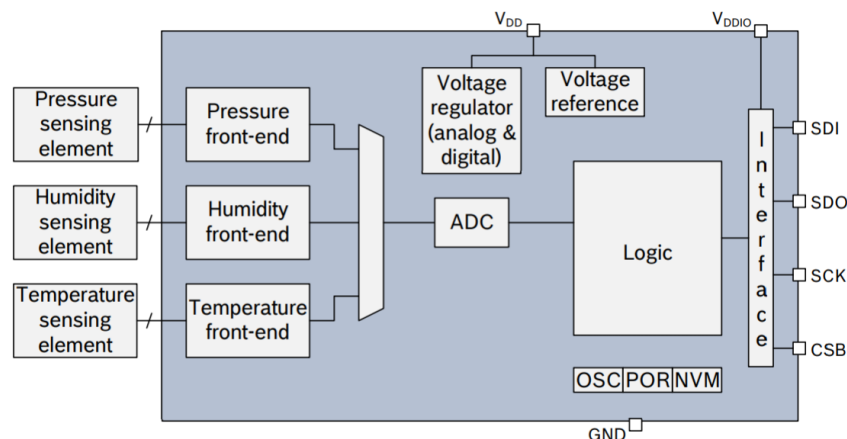


図14 BME-280のスペック、ブロック図

CSB ピンが Vddio に接続されると I2C モードで動作します。I2C のアドレスは SDO 端子の状態で 0x76(SDO を GND)か 0x77(SDO を VDD)を選べます。今回は GND に接続して、アドレス 0x76 で使用します。機能が多い分 I2C のやりとりは少し複雑になります。

しかしながら BME280 も専用のノードが用意されていますので、これを使えば簡単に温度湿度気圧の 3 つ測定値を取り出すことができます。BME-280 のノードは以下のようにインストールします。

```
sudo npm install node-red-contrib-bme280-rpi -g
```

図 15 に BME-280 ノードの使用例を示します。BME-280 ノードはそれ自体でタイマをもっていて一定時間の繰り返し計測が可能になります。その出力は JSON ストリングで 3 つの測定結果をパックにして出力されます。図 16 はその JSON ストリングを function ノードで各要素に分解し Dashboard のグラフに表示させています。function ノードでは JSON.parse()と JSON.stringify()を使って各要素を切り出しています。

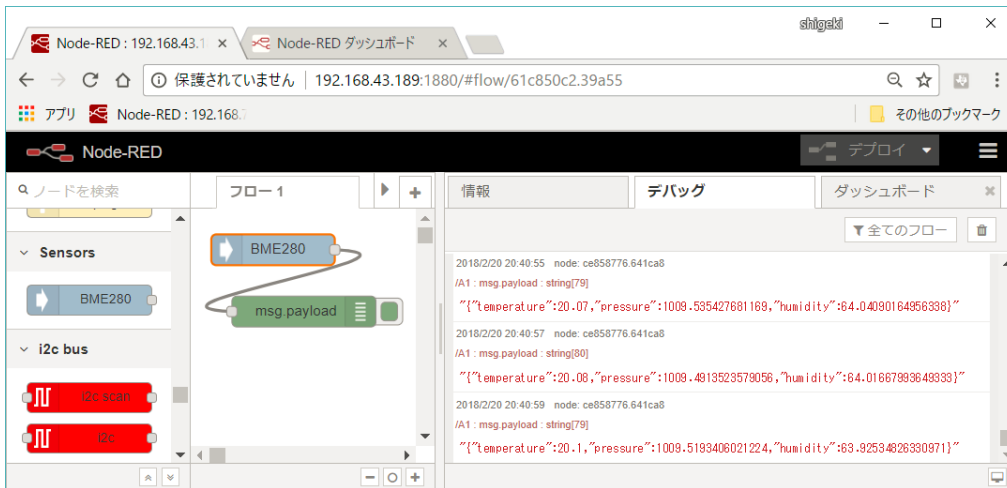


図 15 BME-280 ノードを試す。

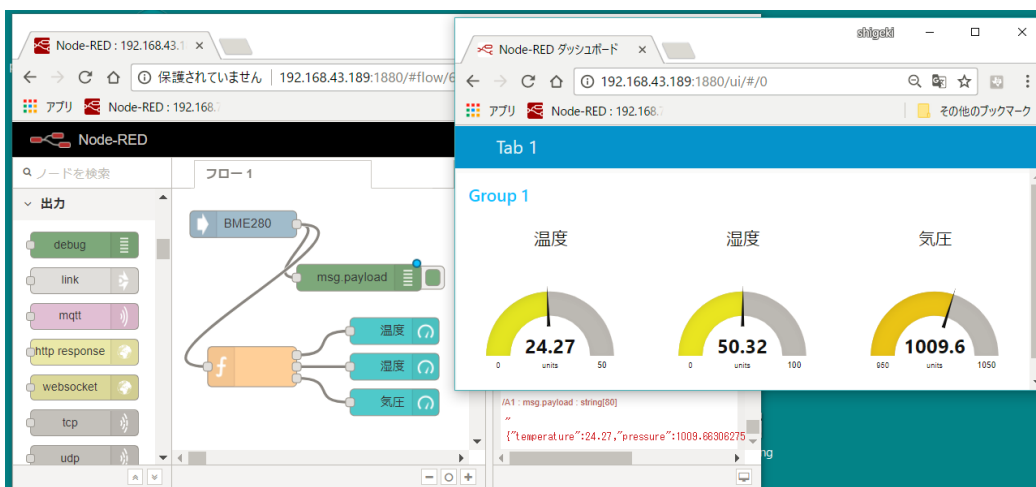


図 16 温度湿度気圧をおのおのグラフ表示させる

リスト1 function ノードの Node.js プログラム

```
obj = JSON.parse(msg.payload);  
var msg0 = new Date(msg.payload);  
var msg1 = new Date(msg.payload);  
var msg2 = new Date(msg.payload);  
msg0.payload=JSON.stringify(obj.temperature);  
msg1.payload=JSON.stringify(obj.humidity).substring(0,5);  
msg2.payload=JSON.stringify(obj.pressure).substring(0,6);  
return [msg0,msg1,msg2];
```

3. クラウド経由

図17に示すように、ローカルネットワーク内に置いたサーバは外部からアクセスできません。そこで、外部サーバ経由のデータ表示を試みます。

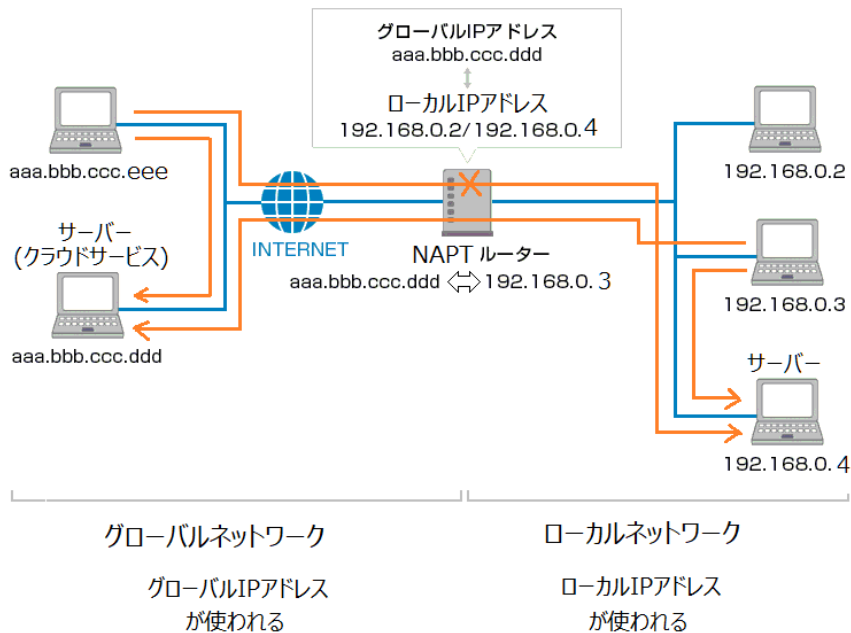
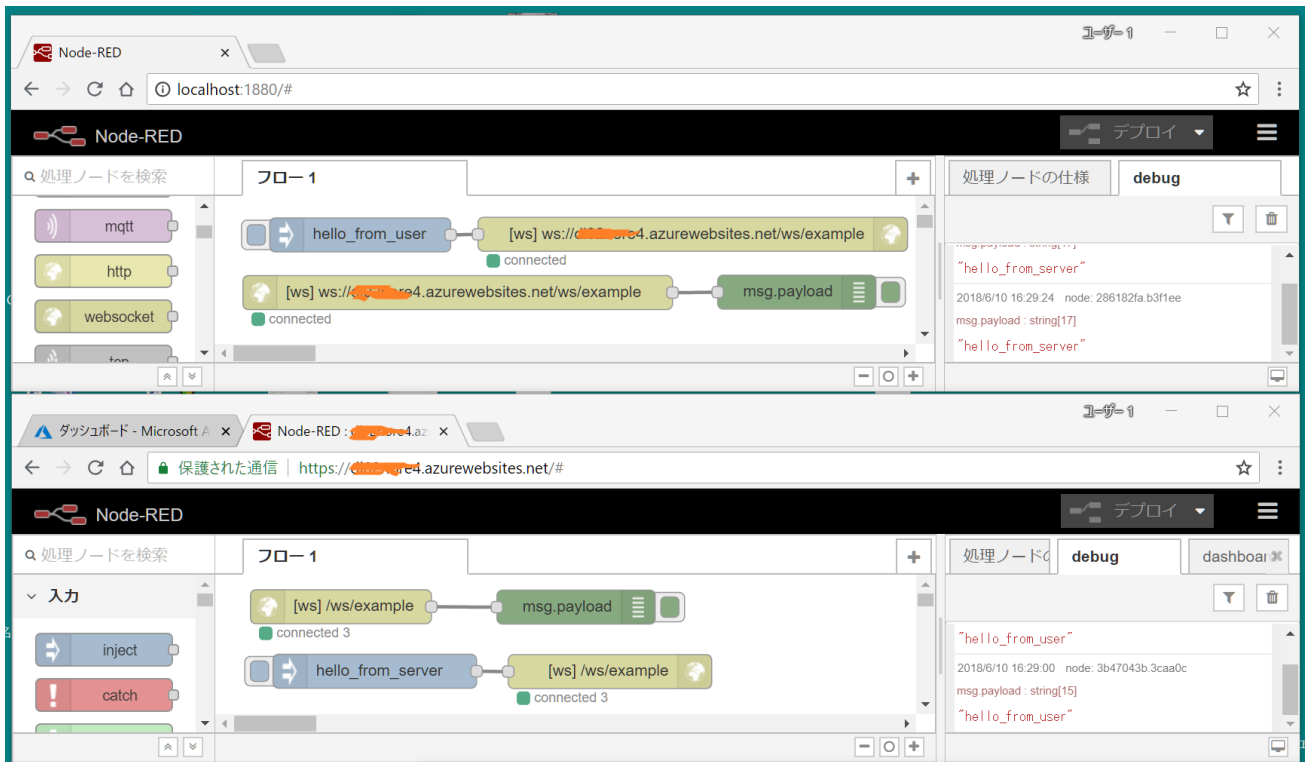


図17 サーバの配置

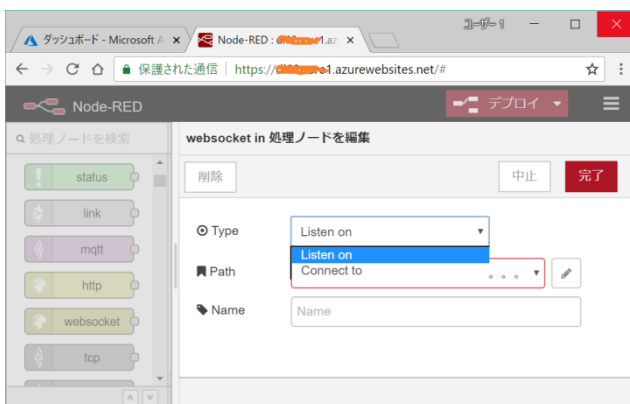
○WebSocket

Web ページで利用されるプロトコル **http** は主にサーバ側からのデータの提供を行います。このプロトコルを拡張してユーザがわからずデータの提供を可能にしたものが **webSocket** です。Node-RED には **websocket** のノードがサーバ側、ユーザ側それぞれの動作が切り替えられる送信ノードと受信ノードが用意されています。

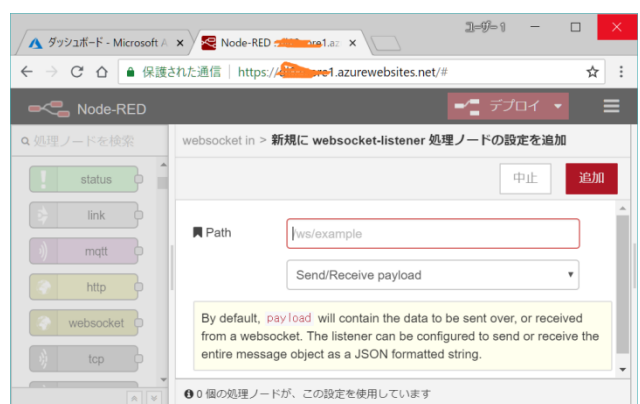


上側が PC、下側がクラウド

図 18 websocket の動作チェック



サーバ/ユーザの設定
 サーバ : Listen on
 ユーザ : Connect to



パスの設定
 サーバの場合 : 内部パスのみ
 ユーザの場合 : URL も必要

図 19 Websocket ノードの設定

ユーザ側、RaspberryPi 側でのパスは、
 ws://xxxx.azurewebsites.net/ws/example
 とします。

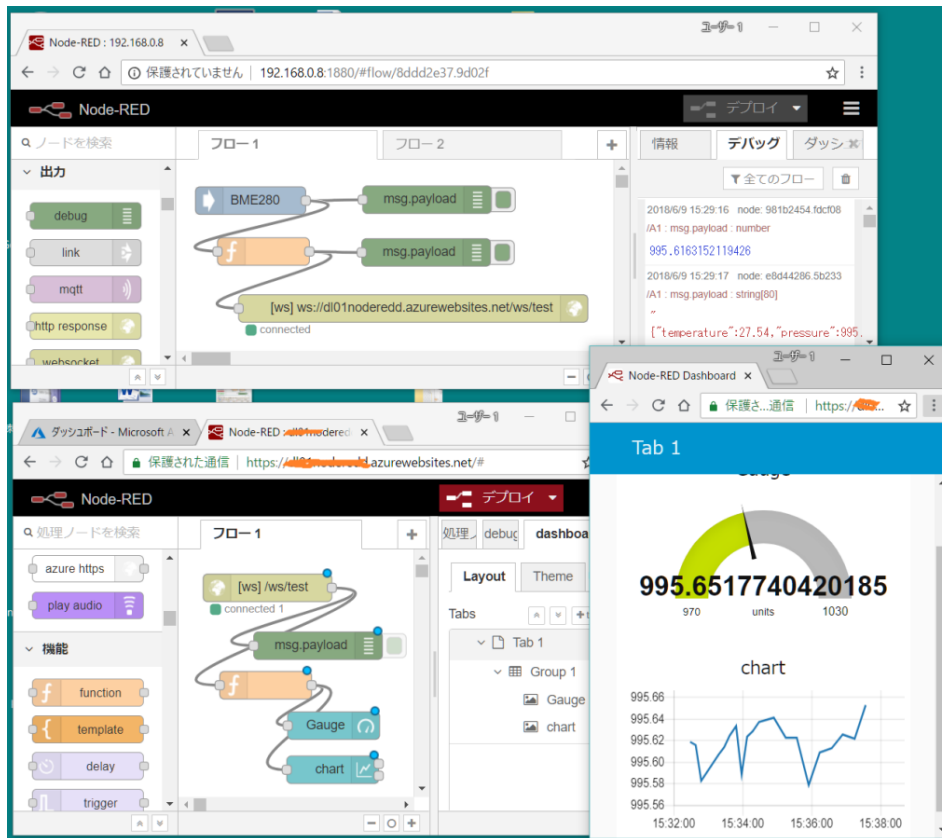


図 20 クラウド経由の温湿度気圧表示（フローは気圧のみの例）

RaspberryPi 側の function ノード

```

a=JSON.parse(msg.payload);
msg.payload=JSON.stringify(a.pressure);
return msg.payload;

```

クラウド側の function ノード

```

msg.payload=Number(msg.payload);
return msg.payload;

```