

# 製造業ITマイスター指導者育成プログラム 研修テキスト 実習用教材(第2日) 高度IT実装技術の習得1 (ラズパイ+見える化実習)



# 製造業ITマイスター研修教材一覧



| 日  | テーマ                     |    | 教材                               |
|----|-------------------------|----|----------------------------------|
| 1  | 製造業IT導入ワークショップ          | 午前 | IoTとシステムの基礎                      |
|    |                         | 午後 | 製造業IT導入ワークショップ                   |
| 2  | 高度IT実装技術の習得 1           | 午前 | IoTによるシステム開発入門                   |
|    |                         | 午後 | 高度IT実装技術の習得 1 (ラズパイ+見える化実習)      |
| 3  | 高度IT実装技術の習得 2           | 午前 | IoTによる生産管理入門                     |
|    |                         | 午後 | 高度IT実装技術の習得 2 (IoTセンサー実装実習)      |
| 4  | システム構築技術の習得 1           | 午前 | IoTによる在庫管理入門                     |
|    |                         | 午後 | システム構築技術の習得 1 (業務システムの基本パターン)    |
| 5  | システム構築技術の習得 2           | 午前 | IoTによるデータ分析入門                    |
|    |                         | 午後 | システム構築技術の習得 2 (データ分析)            |
| 6  | PBL 1 (事例企業調査)          | 午前 | 事例企業調査                           |
|    |                         | 午前 | 事例企業の課題モデル化実習                    |
| 7  | PBL 2<br>(課題の設定と解決策の提案) | 午後 | システム構築の実際                        |
|    |                         | 午後 | システム構築実習 (1) 課題の設定と解決策の提案        |
| 8  | 高度IT実装技術の適用             | 午前 | IT経営の実践方法                        |
|    |                         | 午後 | システム構築実習 (2) 高度IT実装技術の適用         |
| 9  | システム構築技術の適用             | 午前 | 情報システムセキュリティ基礎<br>知財とオープン&クローズ戦略 |
|    |                         | 午後 | システム構築実習 (3) システム構築技術の適用         |
| 10 | 筆記試験および成果発表会            | 午前 | 個人と組織の発展に繋がるキャリアデザイン講座<br>(筆記試験) |
|    |                         | 午後 | (成果発表会)                          |

# 前半5日間の進め方



## 午後の実習

- 1日目 実習のための環境設定 → 課題発見ワークショップ
- 2日目 デバイス信号のイン/アウト → センサデータの見える化
- 3日目 メールとWebサーバ利活用 → 人感センサとカメラの利用
- 4日目 業務システムの基本パターン → バーコードリーダとNFC
- 5日目 データ分析続き → 工程進捗管理ボード

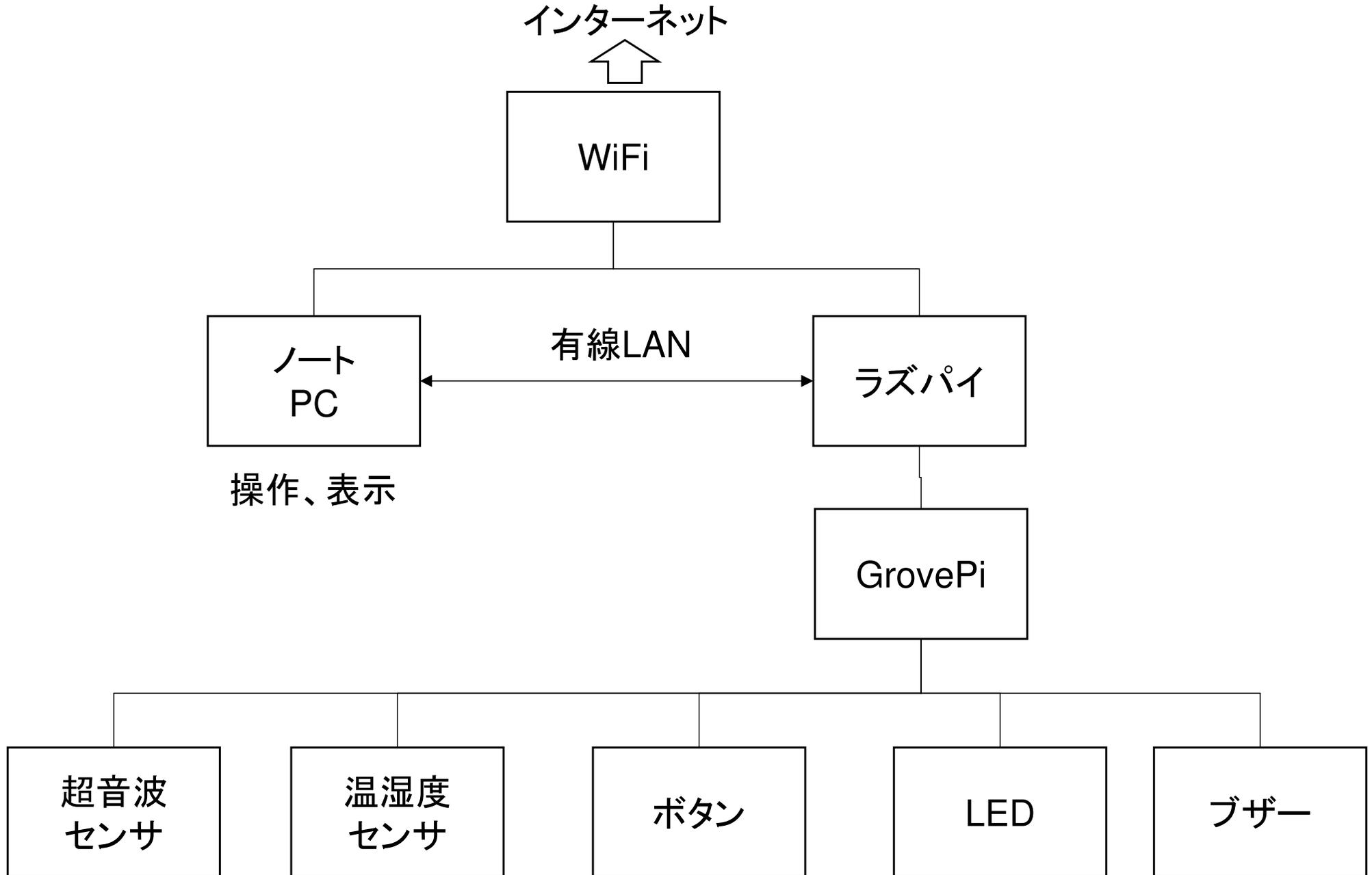
## 1. デバイス信号のイン/アウト

- ① ブザーを鳴らす (Python)
- ② LEDを一定間隔で点滅させる (Python)
- ③ ボタンのONとOFFの表示 (Python)
- ④ 温湿度センサの値を表示する (Python)
- ⑤ Node-REDを使う

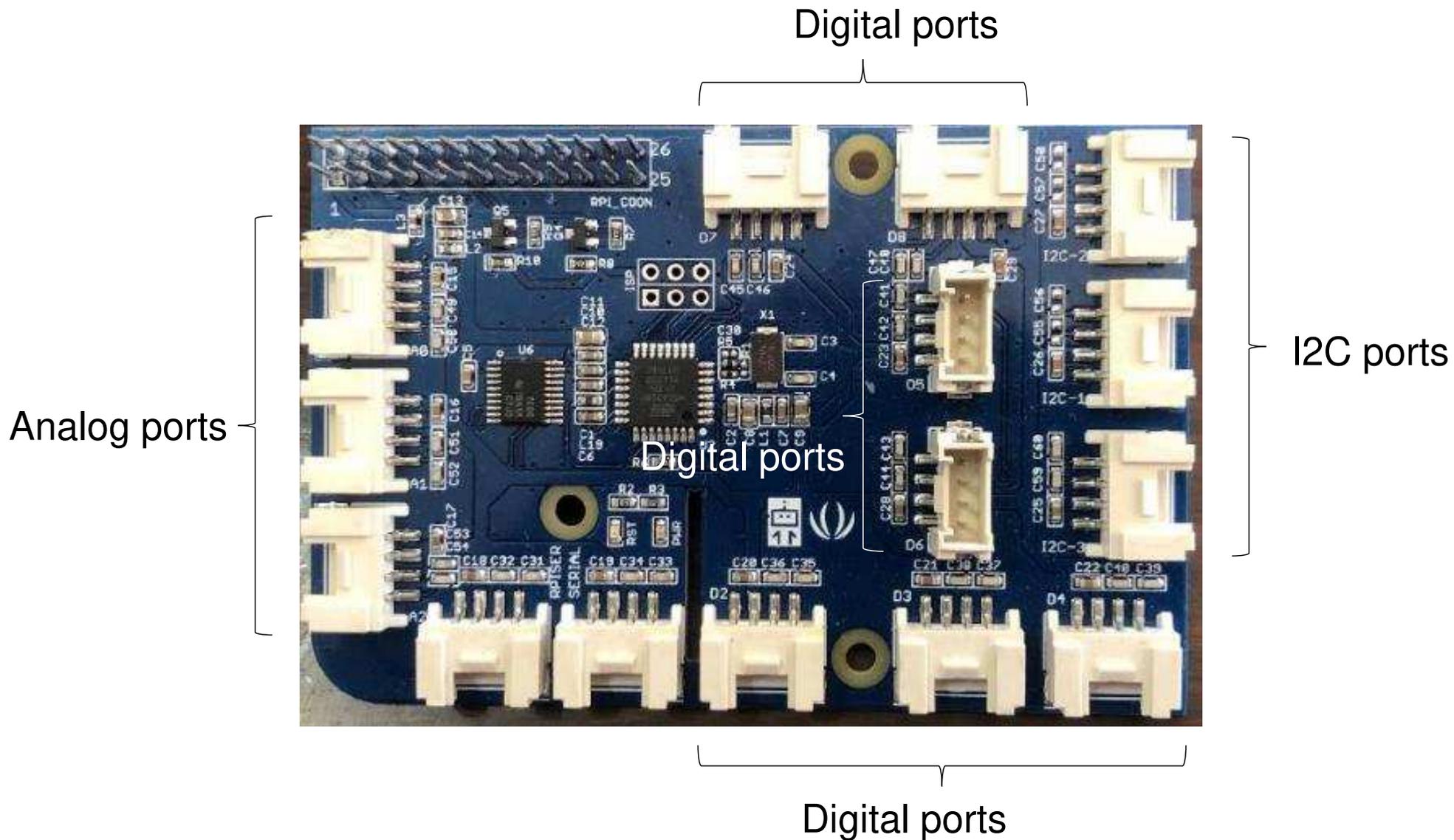
## 2. センサデータの見える化

- ① Node-REDで温湿度センサの値を表示させる
- ② 超音波距離センサの値を表示させる
- ③ ダッシュボードを使う
- ④ クラウドサービスを活用する
- ⑤ 最低限のセキュリティ設定

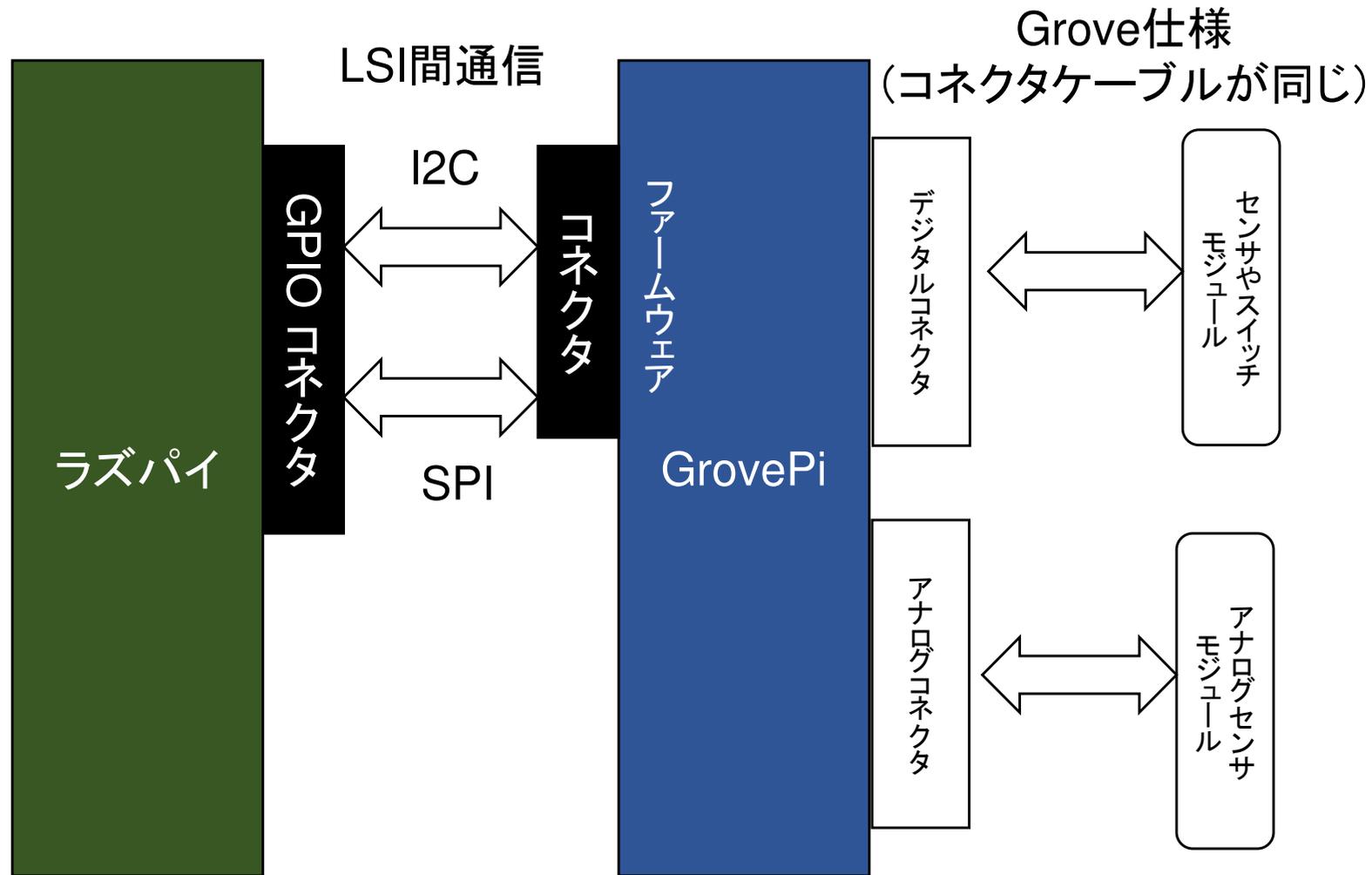
# システム構成



# GrovePi+のコネクタ配置



# GrovePiとラズパイの関係



GPIOのピンを分配しているのではない。  
ファームウェア(ハードウェアに組込まれたソフトウェア)でラズパイと通信している。

## 1. デバイス信号のイン/アウト

- ① ブザーを鳴らす (Python)
- ② LEDを一定間隔で点滅させる (Python)
- ③ ボタンのONとOFFの表示 (Python)
- ④ 温湿度センサの値を表示する (Python)
- ⑤ Node-REDを使う

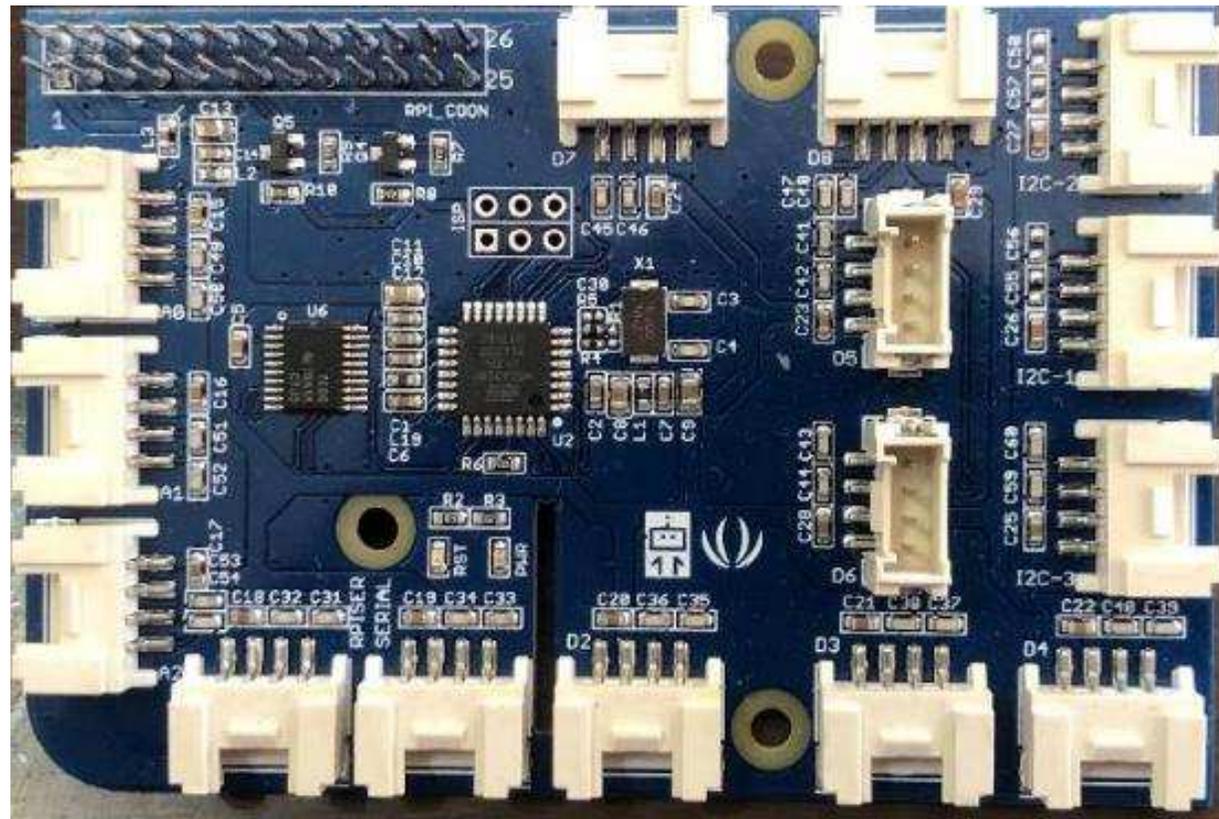
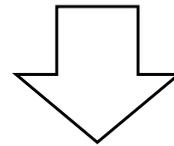
## 2. センサデータの見える化

- ① Node-REDで温湿度センサの値を表示させる
- ② 超音波距離センサの値を表示させる
- ③ ダッシュボードを使う
- ④ クラウドサービスを活用する
- ⑤ 最低限のセキュリティ設定

# Grove Buzzerの取り付け



Digital port 7 に接続する



# Python シェルを立ち上げる



メニュー>プログラミング>Python 3(IDLE)

Python シェル  
画面が立ち上がる



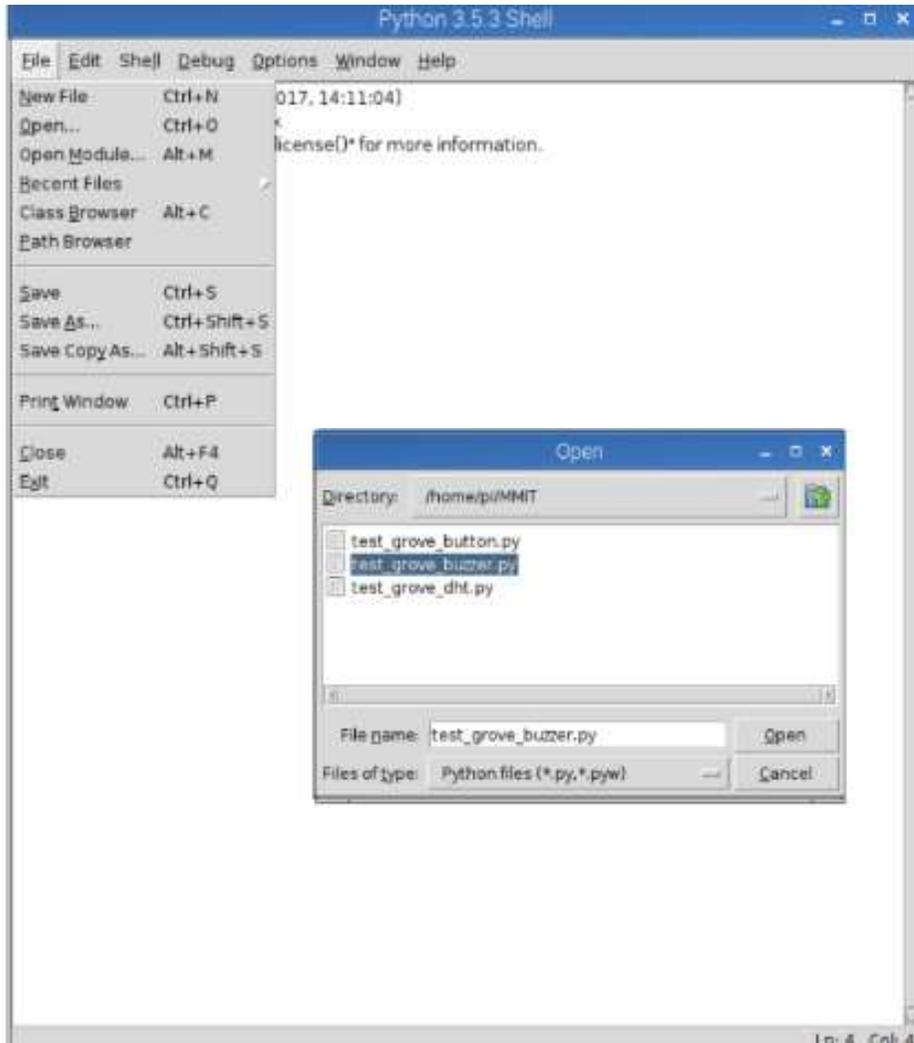
Python IDLEは  
Pythonプログラムの  
開発に使われる  
OSに標準搭載された  
ソフトウェア

# Python プログラムを開く



File > Open > mmit > test\_buzzer.pyを選択 > Open

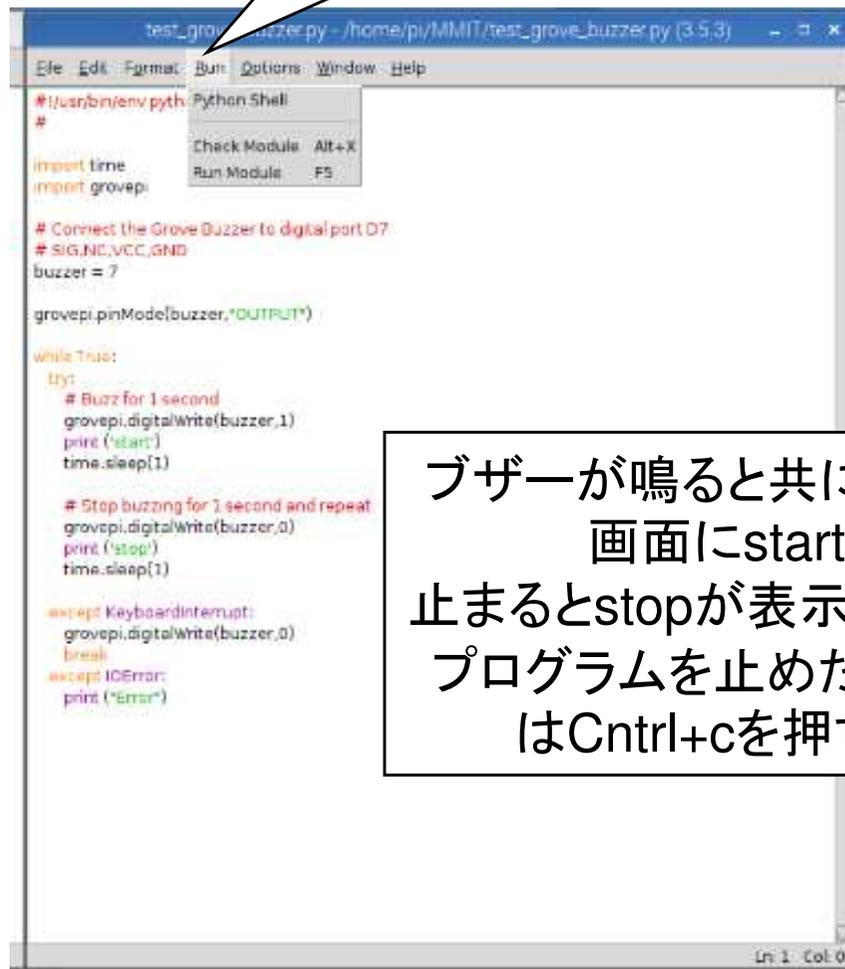
デバッグ可能な  
エディタが開く



# デバッグ(実行)する

エディタのメニュー  
Run > Run Module F5

シェル上で  
プログラムが実行される



```
test_grove_buzzer.py - /home/pi/MMIT/test_grove_buzzer.py (3.5.3)
File Edit Format Run Options Window Help
Python Shell
Check Module Alt+X
Run Module F5

#!/usr/bin/env python
#
import time
import grovepi

# Connect the Grove Buzzer to digital port D7:
# SIG,NE,VCC,GND
buzzer = 7

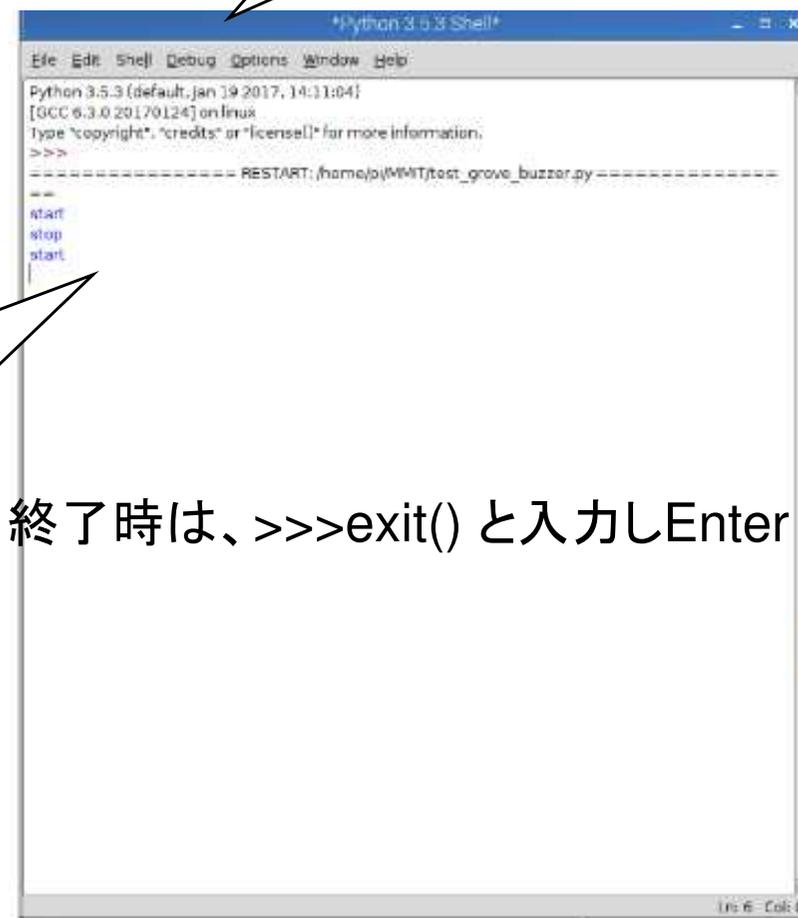
grovepi.pinMode(buzzer,"OUTPUT")

while True:
    try:
        # Buzz for 1 second
        grovepi.digitalWrite(buzzer,1)
        print('start')
        time.sleep(1)

        # Stop buzzing for 1 second and repeat
        grovepi.digitalWrite(buzzer,0)
        print('stop')
        time.sleep(1)

    except KeyboardInterrupt:
        grovepi.digitalWrite(buzzer,0)
        break
    except IOError:
        print("Error")
```

ブザーが鳴ると共にシェル  
画面にstart  
止まるとstopが表示される。  
プログラムを止めたいとき  
はCntrl+cを押す。

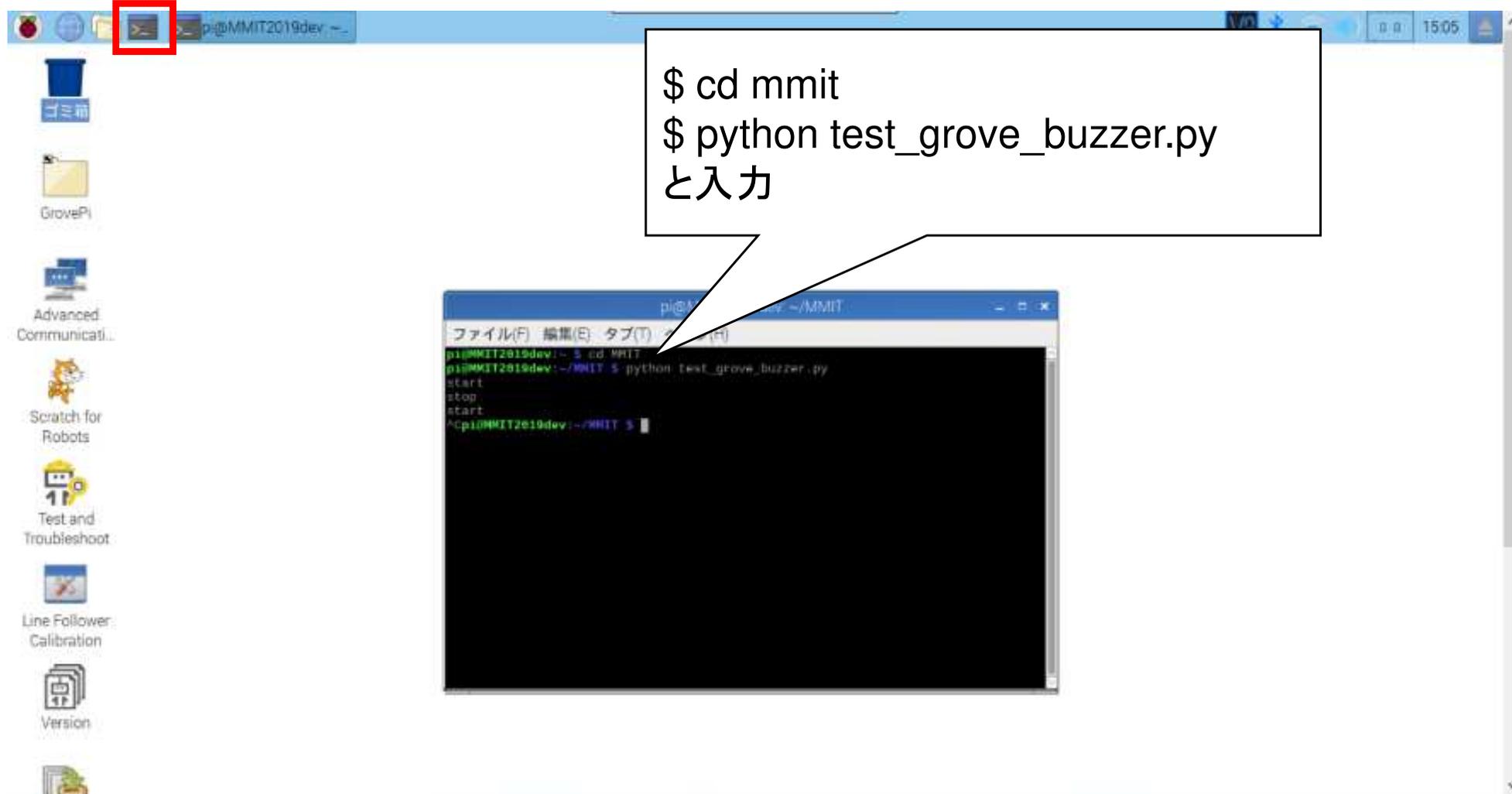


```
Python 3.5.3 Shell
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
----- RESTART: /home/pi/MMIT/test_grove_buzzer.py -----
>>>
start
stop
start
```

終了時は、>>>exit() と入力しEnter

# コンソールでの実行

コンソールアイコンを  
クリック



# ラズパイの標準エディタ



ラズパイに標準でインストールされているエディタを使ってプログラムを開きます。  
\$ leafpad test\_grove\_buzzer.py[enter]

```
pi@mmit19126: ~/mmit
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
pi@mmit19126:~$ cd mmit
pi@mmit19126:~/mmit$ python test_grove_buzzer.py
start
stop
start
stop
start
stop
start
stop
start
stop
^Cpi@mmit19126:~/mmit$ leafpad test_grove_buzzer.py
```

leafpadというGUIで動くエディタでプログラムが開きます。

```
test_grove_buzzer.py
ファイル(F) 編集(E) 検索(S) オプション(O) ヘルプ(H)
#!/usr/bin/env python
#
import time
import grovepi

# Connect the Grove Buzzer to digital port D8
# SIG, NC, VCC, GND
buzzer = 7

grovepi.pinMode(buzzer, "OUTPUT")

while True:
    try:
        # Buzz for 1 second
        grovepi.digitalWrite(buzzer, 1)
        print ('start')
        time.sleep(1)

        # Stop buzzing for 1 second and repeat
        grovepi.digitalWrite(buzzer, 0)
        print ('stop')
        time.sleep(1)

    except KeyboardInterrupt:
        grovepi.digitalWrite(buzzer, 0)
        break
    except IOError:
        print ("Error")
```

# プログラム (test\_grove\_buzzer.py)

```
#!/usr/bin/env python  
#
```

```
import time  
import grovepi
```

GrovePiを使うためのライブラリ

```
# Connect the Grove Buzzer to digital port D7  
# SIG,NC,VCC,GND  
buzzer = 7
```

GrovePiのデジタルポートの番号

```
grovepi.pinMode(buzzer,"OUTPUT")
```

ポート7を出力として設定

```
for n in range(5):
```

繰り返し回数

```
# Buzz for 1 second
```

```
grovepi.digitalWrite(buzzer,1)
```

ブザーON

```
print ('start')
```

```
time.sleep(1)
```

1秒待つ

繰り返しループ

```
# Stop buzzing for 1 second and repeat
```

```
grovepi.digitalWrite(buzzer,0)
```

ブザーOFF

```
print ('stop')
```

```
time.sleep(1)
```

# ■ 実習でのPythonプログラムについて

- GPIOを直接コントロールするプログラムではありません
- GrovePiを使って配線やプログラムをより簡単にしたものを実習に使っています

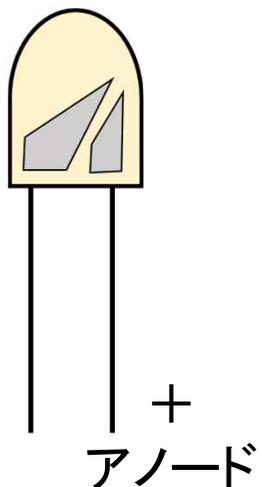
## 1. デバイス信号のイン/アウト

- ① ブザーを鳴らす (Python)
- ② LEDを一定間隔で点滅させる (Python)
- ③ ボタンのONとOFFの表示 (Python)
- ④ 温湿度センサの値を表示する (Python)
- ⑤ Node-REDを使う

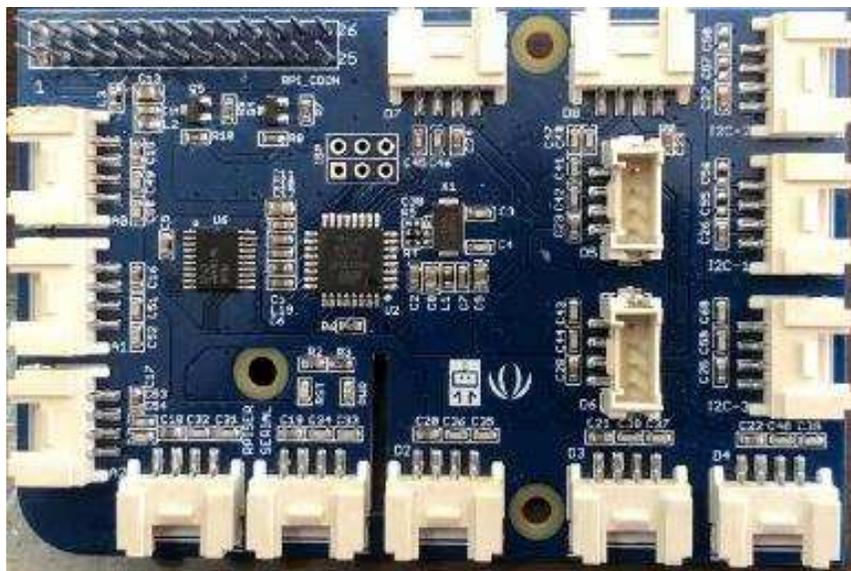
## 2. センサデータの見える化

- ① Node-REDで温湿度センサの値を表示させる
- ② 超音波距離センサの値を表示させる
- ③ ダッシュボードを使う
- ④ クラウドサービスを活用する
- ⑤ 最低限のセキュリティ設定

# Grove White LEDの組立、取付



電極の形状で見分けて  
モジュール基板に挿し込む



Digital port 5 に接続する

# プログラム (test\_grove\_led.py)



```
#!/usr/bin/env python

import time
import grovepi

# Connect the Grove LED to digital port D5
led = 5

grovepi.pinMode(led,"OUTPUT")
time.sleep(1)

while True:
    try:
        #Blink the LED
        grovepi.digitalWrite(led,1)          # Send HIGH to switch on LED
        print ("LED ON!")
        time.sleep(1)

        grovepi.digitalWrite(led,0)          # Send LOW to switch off LED
        print ("LED OFF!")
        time.sleep(1)

    except KeyboardInterrupt: # Turn LED off before stopping
        grovepi.digitalWrite(led,0)
        break
    except IOError:          # Print "Error" if communication error encountered
        print ("Error")
```

GrovePiを使うためのライブラリ

以下の処理の無限ループ

Ctrl+c でループを抜ける

# Pythonプログラムの基本構成



**【import 文】** Pythonで使用するプログラムライブラリを組込みます。  
GrovePiを使う時も、GrovePi用のライブラリを組込んでいます

**【初期化処理】**  
(GPIOを使う場合はここでGPIOの初期化を行います)

**【プログラムメインの処理】**  
(ここでループするケースが多くなります)

**【プログラム終了時の処理】**

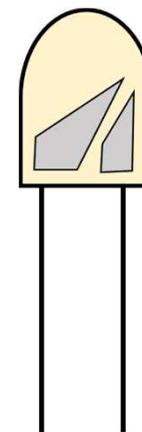
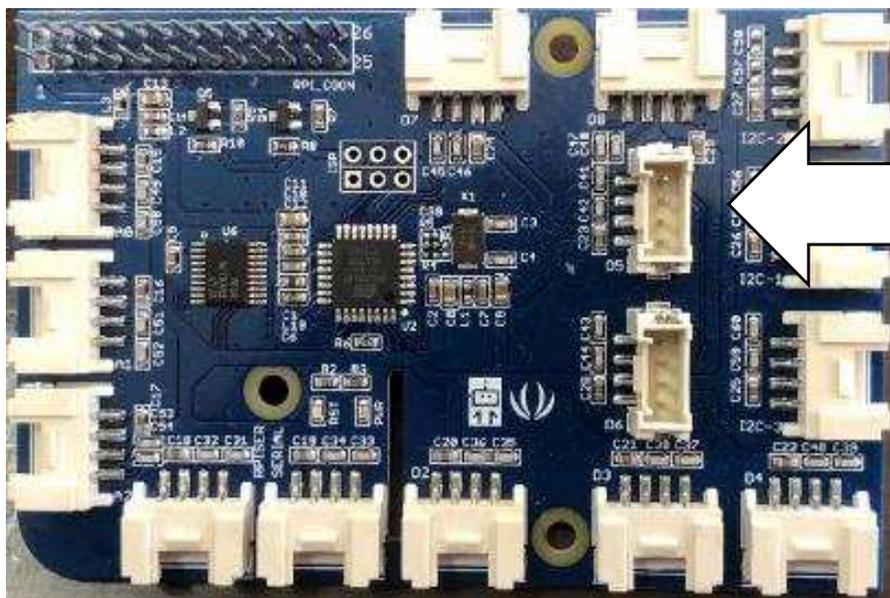
## 1. デバイス信号のイン/アウト

- ① ブザーを鳴らす (Python)
- ② LEDを一定間隔で点滅させる (Python)
- ③ ボタンのONとOFFの表示 (Python)
- ④ 温湿度センサの値を表示する (Python)
- ⑤ Node-REDを使う

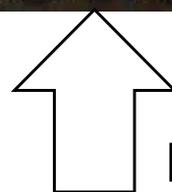
## 2. センサデータの見える化

- ① Node-REDで温湿度センサの値を表示させる
- ② 超音波距離センサの値を表示させる
- ③ ダッシュボードを使う
- ④ クラウドサービスを活用する
- ⑤ 最低限のセキュリティ設定

# Grove Buttonの取付



Digital port 5 に接続したまま



Digital port 2 に接続する



# プログラム (test\_grove\_button1.py)



```
#!/usr/bin/env python
#
#
import time
import grovepi

# Connect the Grove Button to digital port D3
# SIG,NC,VCC,GND
button = 2

grovepi.pinMode(button,'INPUT')

while True:
    try:
        print(grovepi.digitalRead(button))
        time.sleep(.5)

    except IOError:
        print ("Error")
```

# プログラムを修正する



```
#!/usr/bin/env python  
#
```

```
import time  
import grovepi
```

```
# Connect the Grove Button to digital port D2  
# SIG,NC,VCC,GND
```

```
button = 2
```

```
led = 5
```

```
grovepi.pinMode(button,"INPUT")  
grovepi.pinMode(led,"OUTPUT")
```

```
> while True:
```

```
    try:
```

```
        mode = grovepi.digitalRead(button)
```

```
        grovepi.digitalWrite(led,mode)
```

```
        print (mode)
```

```
        time.sleep(.5)
```

```
    except IOError:
```

```
        print ("Error")
```

GrovePiのポート番号

ボタンの状態を取得して  
ボタンの状態をLEDを点滅させる

# test\_grove\_button2.pyで保存



The image shows a Python IDE window with the title bar '\*test\_grove\_button1.py - /home/pi/mmit/test\_grove'. The 'File' menu is open, and the 'Save As...' option is highlighted with a white arrow. The 'Save As' dialog box is open, showing the directory '/home/pi/mmit' and a list of files: test\_grove\_button1.py, test\_grove\_buzzer.py, test\_grove\_dht11.py, and test\_grove\_led.py. The 'File name' field contains 'test\_grove\_button2.py' and the 'Files of type' dropdown is set to 'Python files (\*.py, \*.pyw)'. A white arrow points to the 'Save' button in the dialog box.

ボタンを押すとLEDがOFF(消灯)  
離すとON(点灯)するようなプログラムを  
作成してください。

# 作業



- test\_button2.pyをエディタで開く
- save as を選びファイル名をtest\_button3.pyとする
- 次のページの様に改変しsaveする

# test\_button3.py



```
button = 2
led = 5

grovepi.pinMode(button,"INPUT")
grovepi.pinMode(led,"OUTPUT")

while True:
    try:
        mode = grovepi.digitalRead(button)
        print (mode)
        if mode == 1 :
            grovepi.digitalWrite(led,0)
        else :
            grovepi.digitalWrite(led,1)
            time.sleep(.5)

    except IOError:
        print ("Error")
```

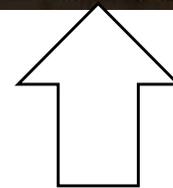
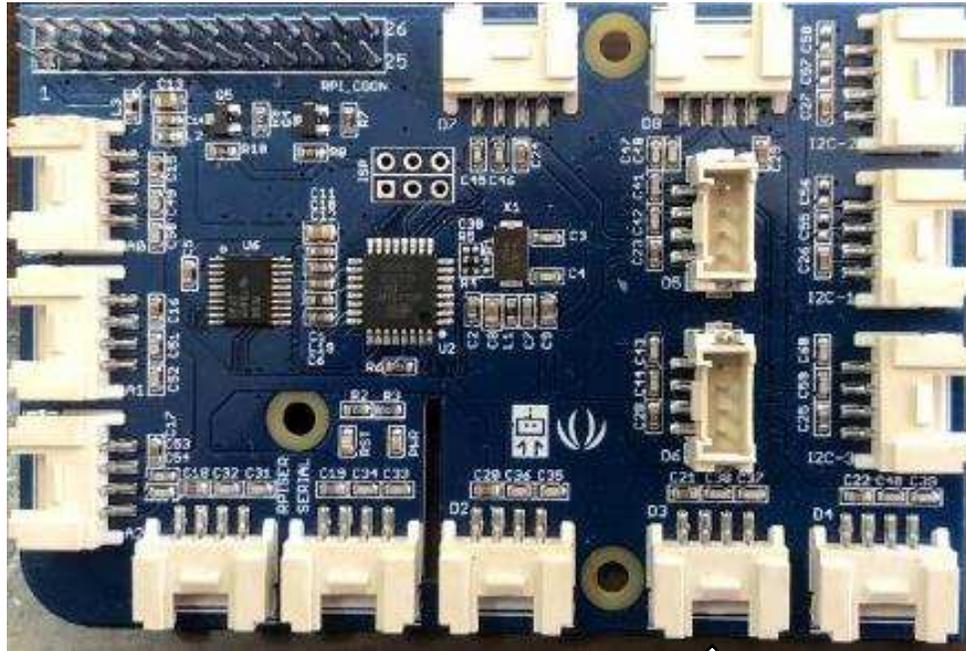
## 1. デバイス信号のイン/アウト

- ① ブザーを鳴らす (Python)
- ② LEDを一定間隔で点滅させる (Python)
- ③ ボタンのONとOFFの表示 (Python)
- ④ 温湿度センサの値を表示する (Python)
- ⑤ Node-REDを使う

## 2. センサデータの見える化

- ① Node-REDで温湿度センサの値を表示させる
- ② 超音波距離センサの値を表示させる
- ③ ダッシュボードを使う
- ④ クラウドサービスを活用する
- ⑤ 最低限のセキュリティ設定

# Grove Temp&Humiの取付



Digital port 3 に接続する

# test\_grove\_dht11.py



```
import grovepi
import math
import time
# Connect the Grove Temperature & Humidity Sensor Pro to digital port D3
# SIG,NC,VCC,GND
sensor = 3 # The Sensor goes on digital port 3.

blue = 0 # The Blue colored sensor.
white = 1 # The White colored sensor.

while True:
    try:
        # This example uses the blue colored sensor.
        # The first parameter is the port, the second parameter is the type of sensor.
        [temp,humidity] = grovepi.dht(sensor,blue)
        if math.isnan(temp) == False and math.isnan(humidity) == False:
            print("temp = %.02f C humidity =%.02f%%"%(temp, humidity))
            time.sleep(5)

    except IOError:
        print ("Error")
```

grovepi.dht(sensor.blue)でセンサーから温度と湿度を取得  
mathライブラリを使って、温度と湿度をセンサーデータから計算  
print()で計算した温度と湿度をコンソールに表示

## 1. デバイス信号のイン/アウト

- ① ブザーを鳴らす (Python)
- ② LEDを一定間隔で点滅させる (Python)
- ③ ボタンのONとOFFの表示 (Python)
- ④ 温湿度センサの値を表示する (Python)

⑤ Node-REDを使う

## 2. センサデータの見える化

- ① Node-REDで温湿度センサの値を表示させる
- ② 超音波距離センサの値を表示させる
- ③ ダッシュボードを使う
- ④ クラウドサービスを活用する
- ⑤ 最低限のセキュリティ設定

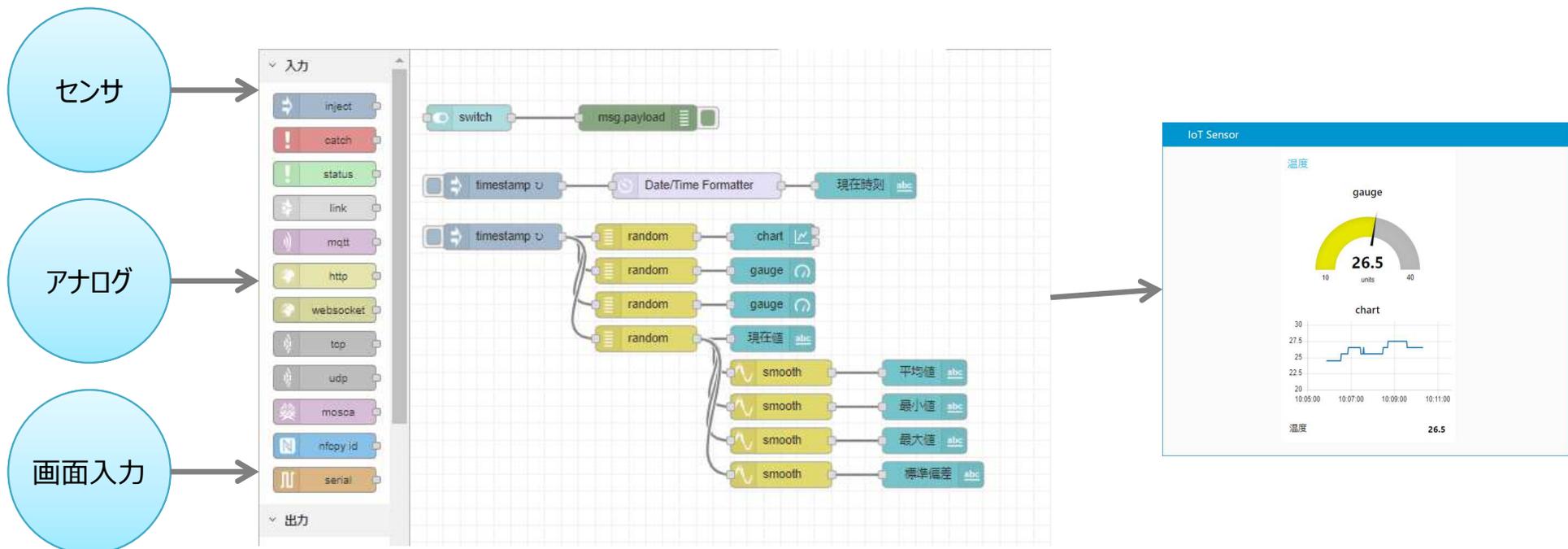
# Node-REDについて



Node-REDは、特別なプログラミング技術が不要で、IoTアプリケーションを簡単に作ることができるオープンソースソフトウェアです。

Node-REDはハードウェアや様々なデバイスを、Webサービスや様々なソフトウェアと接続する為に、IBMのオープンソースプロジェクトとして、2013年に開発が始まりました。

その後、Node-REDが急速に進展し、IoT全般で使用することができるツールになりました。またNode-REDが特に優れているのは、全世界の開発者が参加するコミュニティーが成長し続けており、新しいノードがどんどん追加され、様々なタスクを処理することが可能になっている事です。



# PCのブラウザから接続

ブラウザにURLを直に入力

http://

新しいタブ

×

+

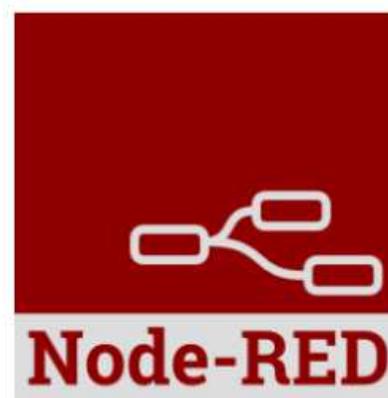


http://192.168.130:1880

ログイン

ユーザID

パスワード



ユーザ名:

admin

パスワード:

.....

ログイン

# GrovePi対応ノードをインストール



The screenshot shows the Node-RED web interface in a browser window. The browser address bar shows 'Node-RED : 192.168.130.10'. The interface includes a search bar for nodes, a palette of nodes (catch, status, link, mqtt, http, websocket, tcp, udp, Watson IoT), and an output section (debug, link). On the right side, a menu is open, showing options like '表示', '読み込み', '書き出し', 'ノードを検索', 'ノードの設定', 'フロー', 'サブフロー', 'パレットの管理', '設定', 'ショートカットキーの説明', 'Node-RED Pi Website', and 'v0.19.4'. The 'パレットの管理' (Manage Palettes) option is highlighted with a red box. The browser's address bar also shows a warning icon and the text '保護されていない通信'.

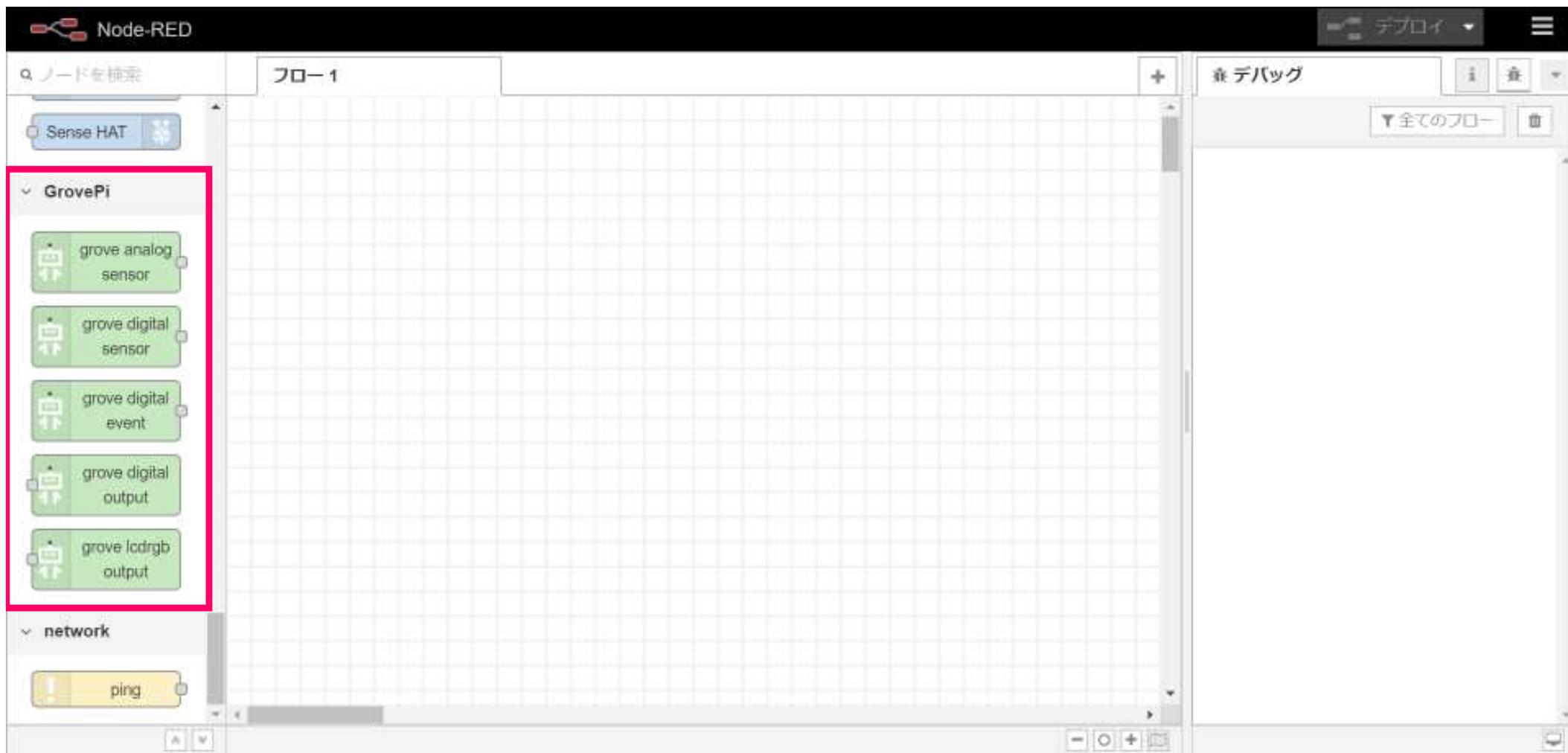
サイドバーの三本ラインアイコンクリック>パレットの管理クリック

# GrovePi対応ノードをインストール



The screenshot shows the Node-RED interface with the 'User Settings' panel open. The search bar in the 'Palettes' section contains 'grovepi', and the search results list three nodes. A white arrow points to the 'Add Node' button for the first result, 'node-red-contrib-grovepi'.

ノードの追加タブ>grovepiで検索>node-red-contrib-grovepiの[ノードの追加]クリック



GrovePi対応のノードが現れる＞念のため再起動

# 時刻ノード追加



時刻を取得するノードは標準では入っていないため、ノード追加する必要があります。

右上の「パレットの管理」→「ノードを追加」→「node-red-contrib-moment」と入力「ノード追加」を押す。



# NFCノード追加



Node-REDで使用できるNFCリーダーのノードは、元々Pythonで作られたライブラリをNode-REDで活用できる形に変換されたものです。

NFCリーダーのノードを使用する場合は、まずpipをインストールする必要があります。

pipとは・・・Pythonのパッケージを管理するためのツール

パッケージ: 公式が配布もしくはサードパーティが配布するもの。

## <作業>

1. `sudo apt-get install python-usb python-pip -y`
2. `sudo pip install -U nfcpy-id-reader`
3. `cat << EOF | sudo tee /etc/udev/rules.d/nfcdev.rules`  
`SUBSYSTEM=="usb", ACTION=="add", ATTRS{idVendor}=="054c",`  
`ATTRS{idProduct}=="06c3", GROUP="plugdev" EOF`
4. `sudo reboot`
5. パレット管理より「node-red-contrib-nfcpy-id」を検索し「ノードを追加をクリック」

※3は一般ユーザーでNode-REDを立ち上げたときにも実行できるようにする方法。

# NFCノード追加



NFCリーダーのノードは標準では入っていないため、ノード追加する必要があります。

右上の「パレットの管理」→「ノードを追加」→「node-red-contrib-nfcpy」と入力  
「node-red-contrib-nfcpy-id」の「ノード追加」を押す。



# SQLiteノード追加



取得した時間データなどを格納するための箱として、SQLiteを使用します。

右上の「パレットの管理」→「ノードを追加」→「node-red-node-sqlite」と入力  
「node-red-node-sqlite」の「ノード追加」を押す。



# ■ 前半5日間の進め方



## 午後の実習

- 1日目 実習のための環境設定 → 課題発見ワークショップ
- 2日目 デバイス信号のイン/アウト → センサデータの見える化
- 3日目 メールとWebサーバ利活用 → 人感センサとカメラの利用
- 4日目 業務システムの基本パターン → バーコードリーダとNFC
- 5日目 データ分析続き → 工程進捗管理ボード

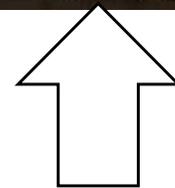
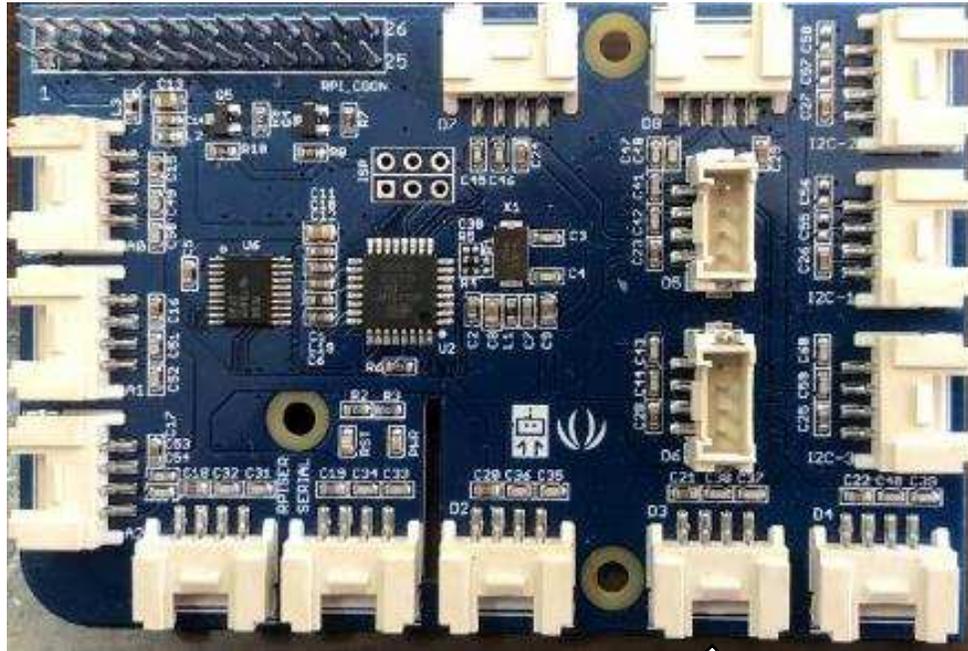
## 1. デバイス信号のイン/アウト

- ① ブザーを鳴らす (Python)
- ② LEDを一定間隔で点滅させる (Python)
- ③ ボタンのONとOFFの表示 (Python)
- ④ 温湿度センサの値を表示する (Python)
- ⑤ Node-REDを使う

## 2. センサデータの見える化

- ① Node-REDで温湿度センサの値を表示させる
- ② 超音波距離センサの値を表示させる
- ③ ダッシュボードを使う
- ④ クラウドサービスを活用する
- ⑤ 最低限のセキュリティ設定

# Grove Temp&Humiの取付



Digital port 3 に接続する

# ノードを配置しプロパティ編集



Board  
ペンアイコンクリック> GrovePi

Sensor Type  
Temperature/Humidity DHT11

Digital Pin  
Digital 3

Interval  
5 Seconds

サイドバーにデバッグ  
メッセージ表示

## 1. デバイス信号のイン/アウト

- ① ブザーを鳴らす (Python)
- ② LEDを一定間隔で点滅させる (Python)
- ③ ボタンのONとOFFの表示 (Python)
- ④ 温湿度センサの値を表示する (Python)
- ⑤ Node-REDを使う

## 2. センサデータの見える化

- ① Node-REDで温湿度センサの値を表示させる
- ② 超音波距離センサの値を表示させる
- ③ ダッシュボードを使う
- ④ ボタンを押すとON、再度押すとOFF
- ⑤ 最低限のセキュリティ設定

# ノードを配置しプロパティ編集



The screenshot shows the Node-RED web interface in a browser. The main workspace contains a flow with two nodes: 'grove digital sensor' and 'msg.payload'. The 'grove digital sensor' node is selected, and its configuration dialog is open. The configuration is as follows:

- Board: GrovePi
- Sensor Type: Ultrasonic Range
- Digital Pin: Digital 6
- Interval: 1 Seconds
- Name: Name

The debug console on the right shows a series of messages with the payload 'number' and values 168, 167, 5, 4, and 4.

Sensor Type  
Ultrasonic Range

Digital Pin  
Digital 6

Interval  
1 Seconds

サイドバーにデバッグ  
メッセージ表示

## 1. デバイス信号のイン/アウト

- ① ブザーを鳴らす (Python)
- ② LEDを一定間隔で点滅させる (Python)
- ③ ボタンのONとOFFの表示 (Python)
- ④ 温湿度センサの値を表示する (Python)
- ⑤ Node-REDを使う

## 2. センサデータの見える化

- ① Node-REDで温湿度センサの値を表示させる
- ② 超音波距離センサの値を表示させる
- ③ ダッシュボードを使う
- ④ クラウドサービスを活用する
- ⑤ 最低限のセキュリティ設定

# 実習の概要



- Node-REDにnode-red-dashboardを組み込む
- 温度センサーのデータを受信するフローを作成する
- 温度データをダッシュボードのtextノードに表示する
- ダッシュボードにGaugeノード、chartノードを追加する
- 湿度データ、物体温度データもダッシュボードに表示する

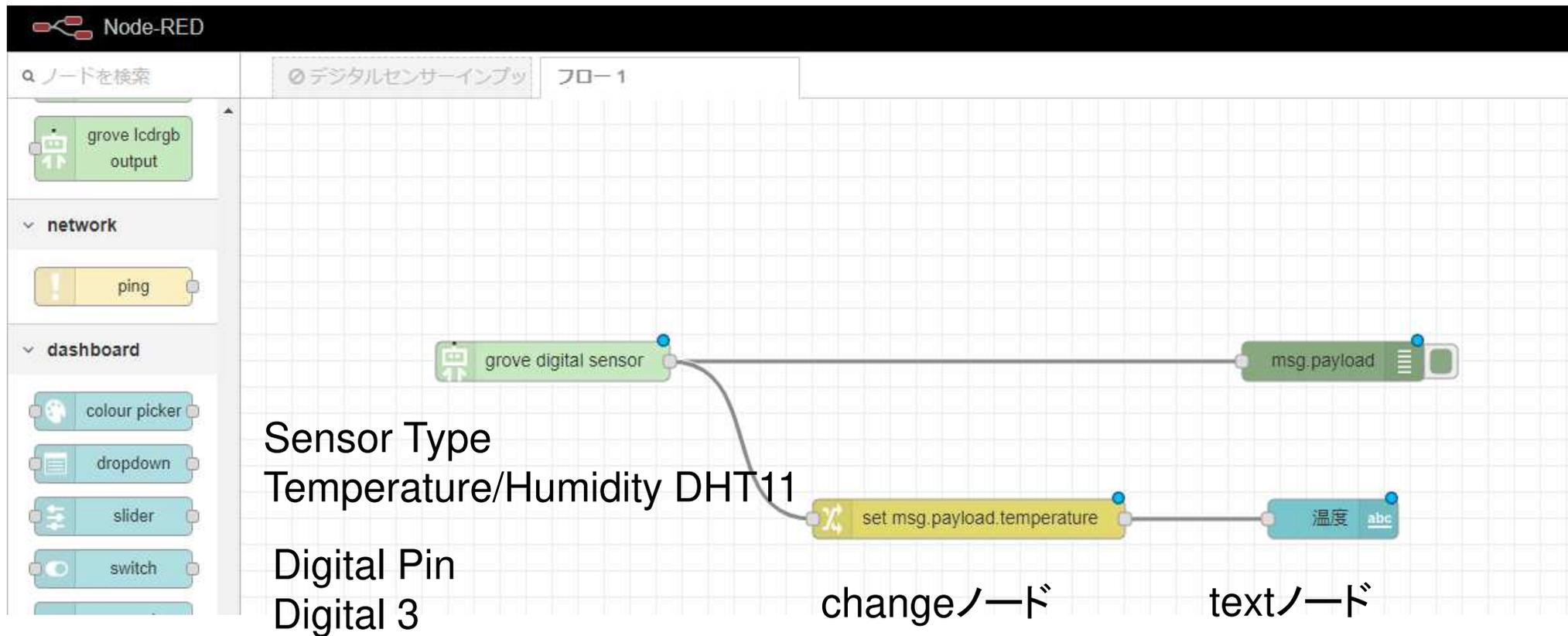
# ダッシュボードノードをインストール



dashboard

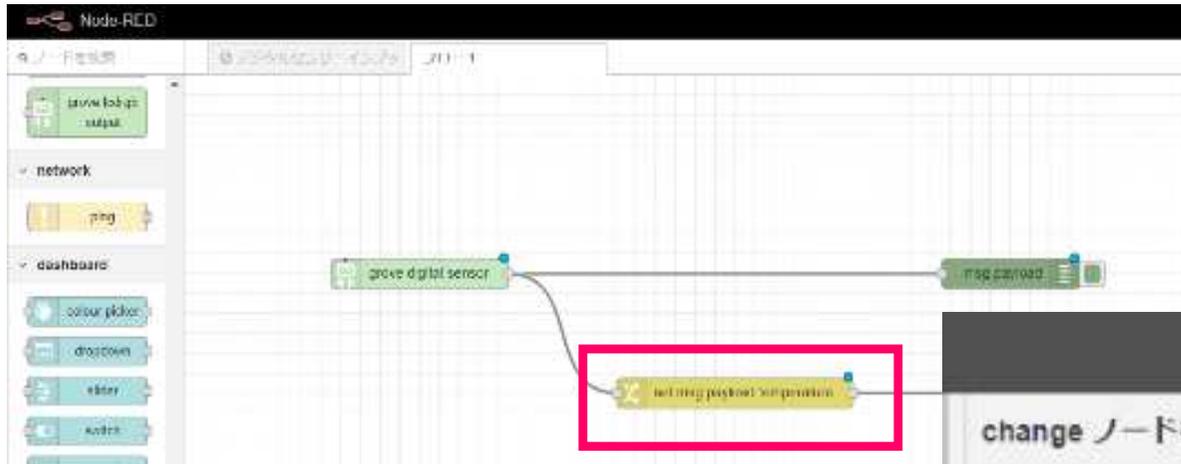
node-red-dashboardを選択

# フローを書く



Interval  
5 Seconds

# changeノードの設定



change ノードを編集

削除 中止 完了

プロパティ

名前 名前

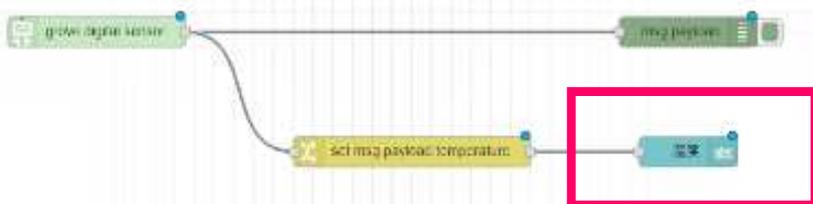
ルール

値の代入 msg.payload

対象の値 msg.payload.temperature

値の代入 msg.payload  
対象の値 msg.payload.temperature

最後にデプロイ



Textノードをダブルクリック

名前 IoT Sensor>[追加]

Group ペンアイコンクリック

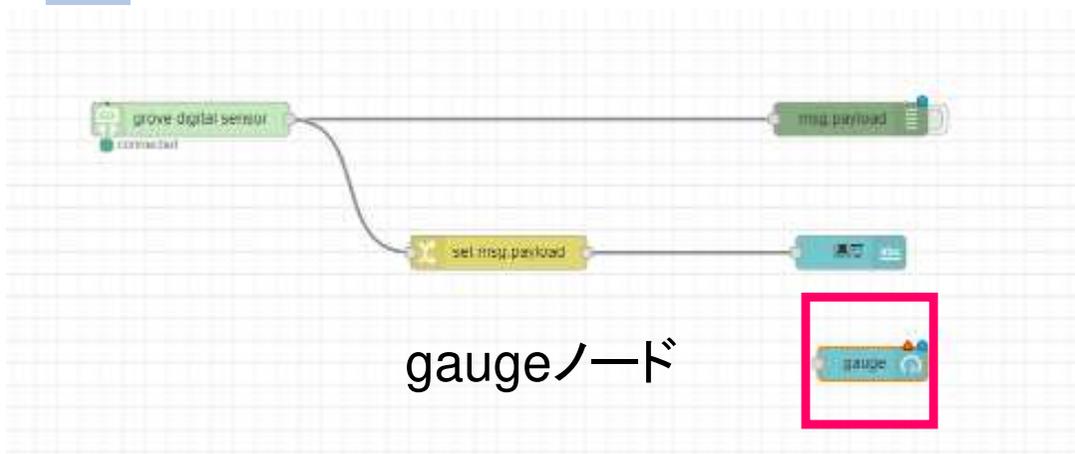


完了

Label 温度







### gauge ノードを編集

削除 中止 完了

▼ プロパティ

Group [IoT Sensor] 温度

Size 自動

Type Gauge

Label gauge

Value format {{value}}

Units units

Range min 10 max 40

Colour gradient

Sectors 10 ... 20 ... 30 ... 40

Name



チャート ノードを編集

削除 中止 完了

▼ プロパティ

Group [IoT Sensor] 温度

Size 自動

Label chart

Type Line chart enlarge points

X-axis last 1 hours OR 1000 points

X-axis Label HH:mm:ss

Y-axis min 20 max 30

Legend None Interpolate linear

Series Colours

Blank label display this text before valid data arrives

Use deprecated (pre 2.5.0) data format.

Name

grove digital sensor connected

msg.payload

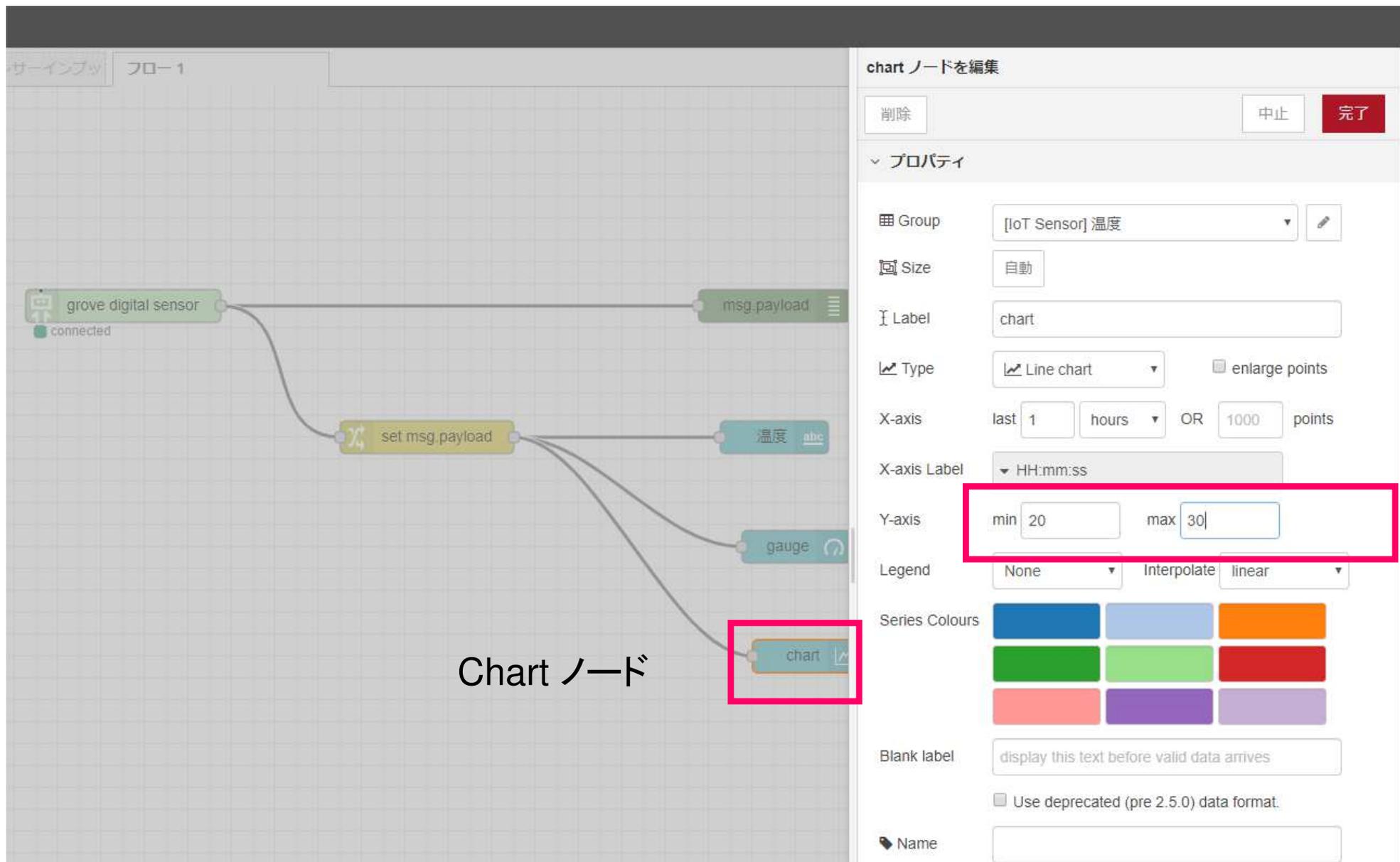
set msg.payload

温度 abc

gauge

chart

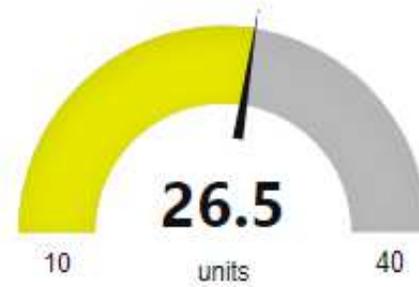
Chart ノード



# IoT Sensor

温度

gauge



chart



温度

26.5

温度だけではなく、湿度もダッシュボードに表示できるようにしてください。

余裕があれば、センサーを超音波距離センサーに変えたり、違うパネルを使った表示にも挑戦してください。

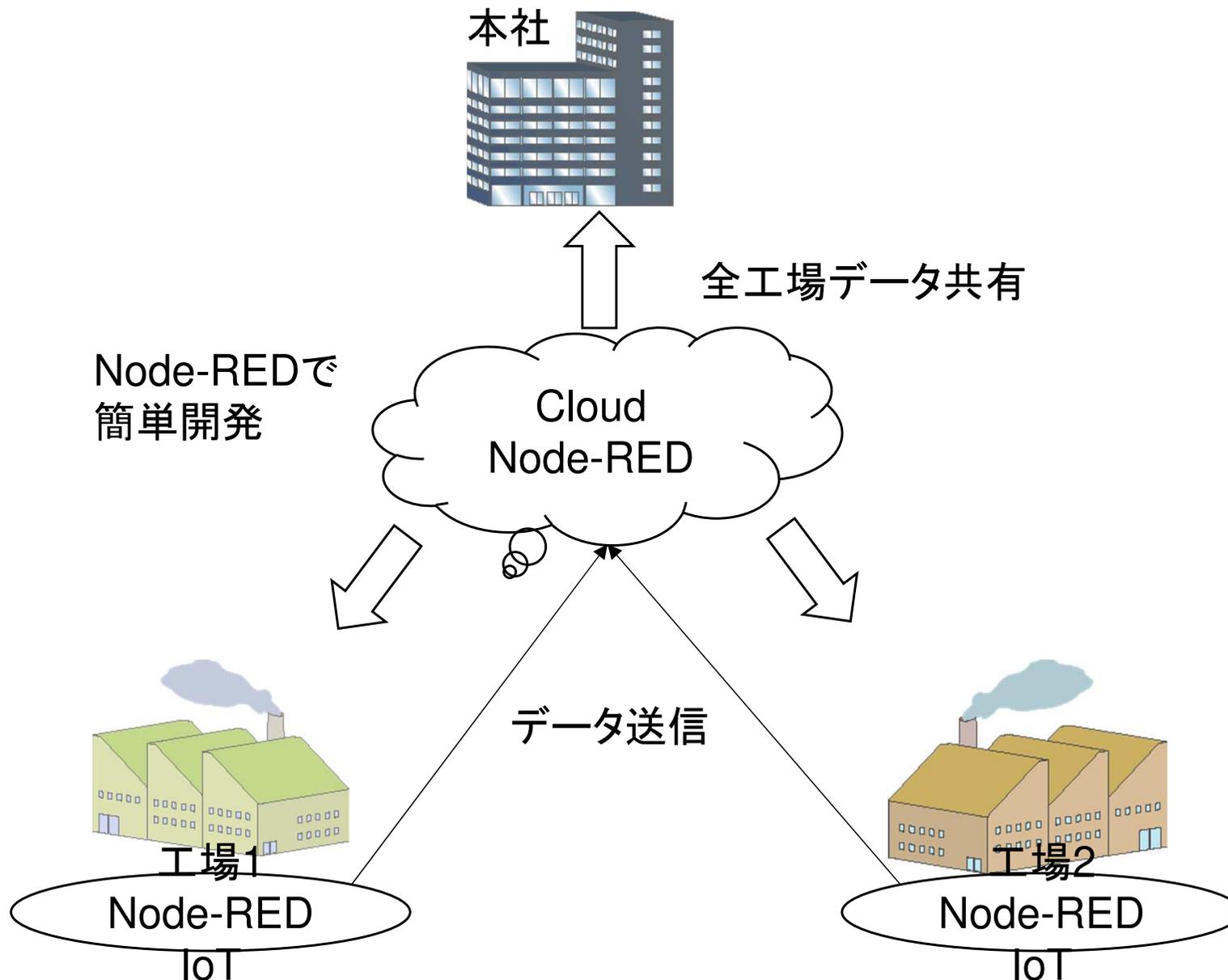
## 1. デバイス信号のイン/アウト

- ① ブザーを鳴らす (Python)
- ② LEDを一定間隔で点滅させる (Python)
- ③ ボタンのONとOFFの表示 (Python)
- ④ 温湿度センサの値を表示する (Python)
- ⑤ Node-REDを使う

## 2. センサデータの見える化

- ① Node-REDで温湿度センサの値を表示させる
- ② 超音波距離センサの値を表示させる
- ③ ダッシュボードを使う
- ④ クラウドサービスを活用する
- ⑤ 最低限のセキュリティ設定

# クラウドサービスを使うメリット



## 1. デバイス信号のイン/アウト

- ① ブザーを鳴らす (Python)
- ② LEDを一定間隔で点滅させる (Python)
- ③ ボタンのONとOFFの表示 (Python)
- ④ 温湿度センサの値を表示する (Python)
- ⑤ Node-REDを使う

## 2. センサデータの見える化

- ① Node-REDで温湿度センサの値を表示させる
- ② 超音波距離センサの値を表示させる
- ③ ダッシュボードを使う
- ④ クラウドサービスを活用する

- ⑤ 最低限のセキュリティ設定

# ログインパスワードをハッシュ化する

以下のコマンドを入力

```
$ cd /usr/lib/node_modules/node-red
```

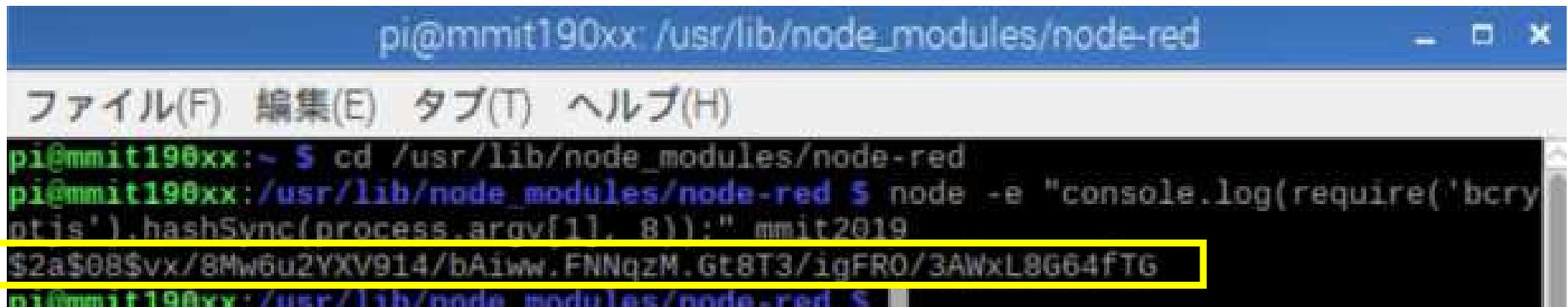


```
pi@mmit190xx: /usr/lib/node_modules/n
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
pi@mmit190xx:~ $ cd /usr/lib/node_modules/node-red
```

設定パスワード

以下のコマンドと設定パスワードを入力

```
$ node -e "console.log(require('bcryptjs').hashSync(process.argv[1], 8))" 
```



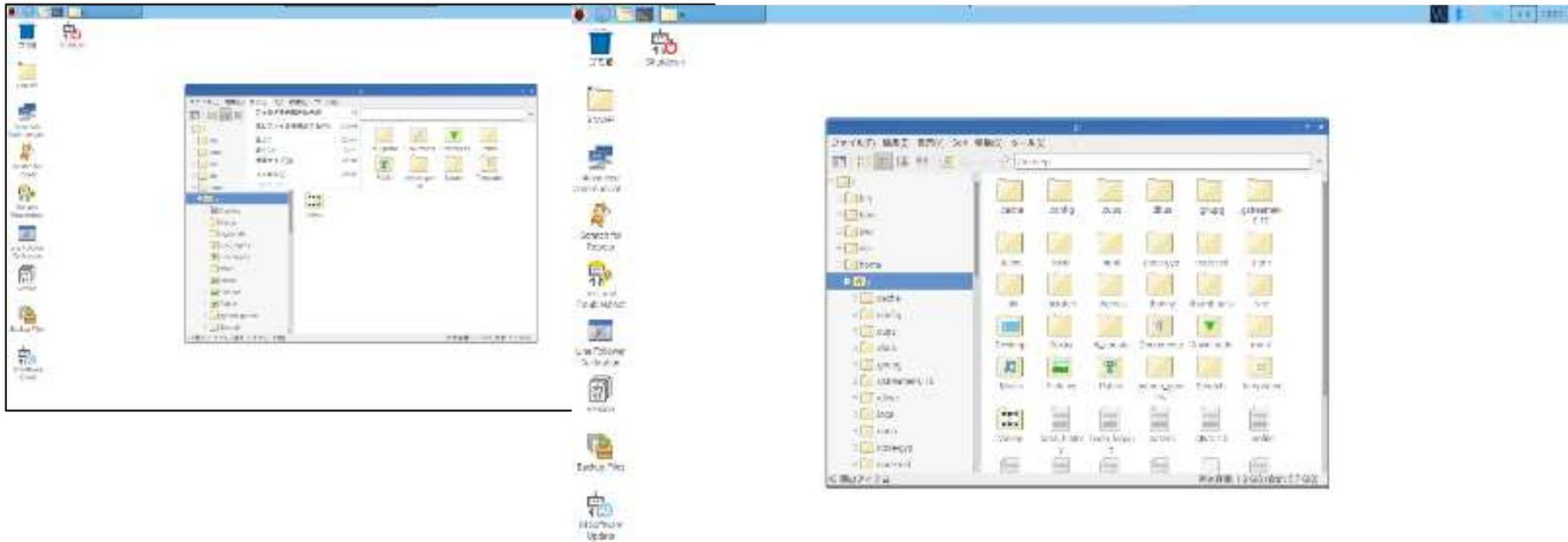
```
pi@mmit190xx: /usr/lib/node_modules/node-red
ファイル(F) 編集(E) タブ(T) ヘルプ(H)
pi@mmit190xx:~ $ cd /usr/lib/node_modules/node-red
pi@mmit190xx: /usr/lib/node_modules/node-red $ node -e "console.log(require('bcryptjs').hashSync(process.argv[1], 8))" mmit2019
$2a$08$vx/8Mw6u2YXV914/bAiw.FNNqzM.Gt8T3/igFR0/3AWxL8G64fTG
pi@mmit190xx: /usr/lib/node_modules/node-red $
```

出力された設定パスワードのハッシュ値を全てコピー

# Node-RED設定ファイルを修正する

ファイルマネージャを開き  
表示>隠しファイルを表示する  
をクリック

隠しファイルが表示される。  
.node-redディレクトリを開く



settings.jsを右クリックして  
ファイル名の変更>settings.js.orgとする。

# Node-RED設定ファイルを修正する

反転部分を見つけて  
行の先頭の“//”を削除する

```
settings.js.org
ファイル(F) 編集(E) 検索(S) オプション(O) ヘルプ(H)
//ui: { path: "ui" },

// Securing Node-RED
// -----
// To password protect the Node-RED editor and admin API, the following
// property can be used. See http://nodered.org/docs/security.html for details.
//adminAuth: {
//  type: "credentials",
//  users: [{
//    username: "admin",
//    password: "$2a$08$zZwTXtja0fB1p2D4sHcMyOCMyz2Z6dNbM6t18sJogENOMcxwV9DN",
//    permissions: ""
//  }]
//},

// To password protect the node-defined HTTP endpoints (httpNodeRoot), or
// the static content (httpStatic), the following properties can be used.
// The pass field is a bcrypt hash of the password.
// See http://nodered.org/docs/security.html#generating-the-password-hash
//httpNodeAuth: {user:"user",pass:"$2a$08$zZwTXtja0fB1p2D4sHcMyOCMyz2Z6dNbM6t18sJogENOMcxwV9DN"},
//httpStaticAuth: {user:"user",pass:"$2a$08$zZwTXtja0fB1p2D4sHcMyOCMyz2Z6dNbM6t18sJogENOMcxwV9DN"}

// The following property can be used to enable HTTPS
// See http://nodejs.org/api/https.html#https_https_createserver_options_requestlistener
// for details on its contents.
// See the comment at the top of this file on how to load the 'fs' module used by
// this setting.
//
```

password:の後ろの“ ”内に  
コピーしたハッシュ値を貼付ける

```
*settings.js.org
ファイル(F) 編集(E) 検索(S) オプション(O) ヘルプ(H)
//ui: { path: "ui" },

// Securing Node-RED
// -----
// To password protect the Node-RED editor and admin API, the following
// property can be used. See http://nodered.org/docs/security.html for details.
adminAuth: {
  type: "credentials",
  users: [{
    username: "admin",
    password: "$2a$08$zZwTXtja0fB1p2D4sHcMyOCMyz2Z6dNbM6t18sJogENOMcxwV9DN",
    permissions: ""
  }]
},

// To password protect the node-defined HTTP endpoints (httpNodeRoot), or
// the static content (httpStatic), the following properties can be used.
// The pass field is a bcrypt hash of the password.
// See http://nodered.org/docs/security.html#generating-the-password-hash
//httpNodeAuth: {user:"user",pass:"$2a$08$zZwTXtja0fB1p2D4sHcMyOCMyz2Z6dNbM6t18sJogENOMcxwV9DN"},
//httpStaticAuth: {user:"user",pass:"$2a$08$zZwTXtja0fB1p2D4sHcMyOCMyz2Z6dNbM6t18sJogENOMcxwV9DN"}

// The following property can be used to enable HTTPS
// See http://nodejs.org/api/https.html#https_https_createserver_options_requestlistener
// for details on its contents.
// See the comment at the top of this file on how to load the 'fs' module used by
// this setting.
//
```

ファイル名を“settings.js”として保存する。

# ■ PCのブラウザでNode-REDを開く



ユーザID

パスワード

でログインできるようになる。

## 本教材利用上の注意事項

本教材の著作権は、厚生労働省に帰属します。  
詳細については、下記の利用規約をご確認ください。  
<https://www.mhlw.go.jp/chosakuken/index.html>