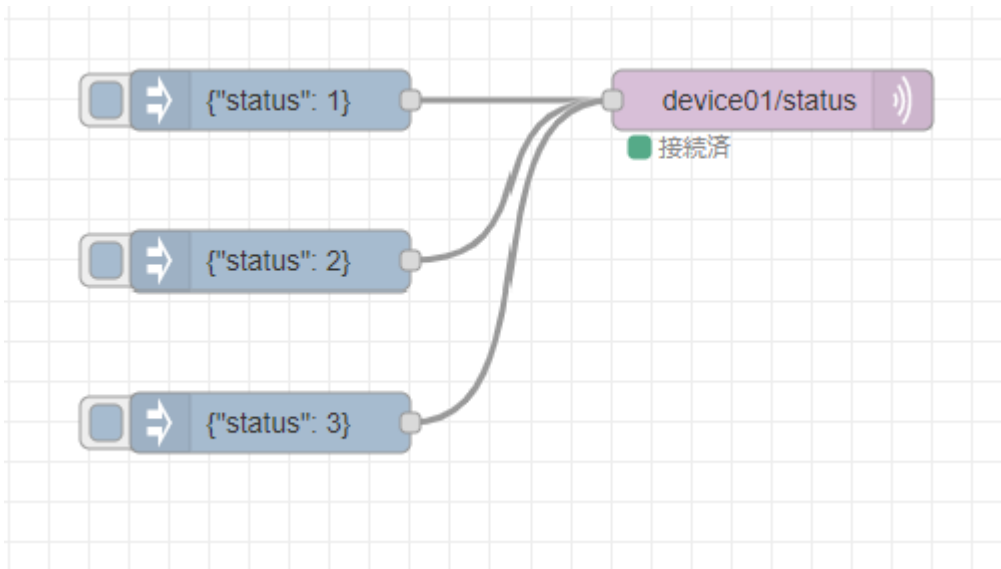


# IoT演習

## MQTT Subscribe

MQTTサーバからの情報を処理する。Subscriberの機能を実装することでサーバーからの要求に応じたデバイスの処理を行うことができます。

### Subscriberへの通知



`mqtt out` ノードを配置して、トピックに `device01/status` と入力します。このノードに `inject` ノードを接続します。 `inject` ノードのpayload は `json` として、それぞれに、status: 1 to 3 を入力します。

```
{"status": 1}
```

## Subscriberの機能追加

トピック `device01/status` へ 0から3のstatusコードに応じてデバイス(Atom lite)のLED表示色を変更します。 ※ M5 stickの場合はdisplayの表示を変更してみてください。

Arduino codeを次のコードを入力します。

```
#include <M5Atom.h>
#include <WiFi.h>
#include <time.h>
```

```
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <SparkFunBME280.h>

// wifi config
#define WIFI_SSID "robot2"
#define WIFI_PASSWORD "robot2090"

// MQTT config
#define MQTT_SERVER "GW IP Address"
#define MQTT_PORT 1883
#define MQTT_BUFFER_SIZE 128
#define TOPIC "device01/bme"
#define TOPIC_STATUS "device01/status"

// デバイスID デバイスIDは機器ごとにユニークにします
#define DEVICE_ID "atom-xxx"

// BME280
BME280 bme;
BME280_SensorMeasurements measurements;

// Ticker
#include <Ticker.h>
Ticker tickerMeasure;

// MQTT Subscribe
const int request_capacity = JSON_OBJECT_SIZE(4);
StaticJsonDocument<request_capacity> json_request;

// MQTT Publish
const int message_capacity = JSON_OBJECT_SIZE(3);
StaticJsonDocument<message_capacity> json_message;
char message_buffer[MQTT_BUFFER_SIZE];

// MQTT用インスタンス作成
WiFiClient espClient;
PubSubClient client(espClient);

// LEDステータス
unsigned long led_status = 0;

// MQTT Subscribeのコールバック
void mqttCallback(char* topic, byte* payload, unsigned int length) {

    DeserializationError err = deserializeJson(json_request, payload, length);
    if( err ){
        Serial.println("Deserialize error");
        Serial.println(err.c_str());
        return;
    }
}
```

```
    serializeJson(json_request, Serial);
    Serial.println("");

    led_status = json_request["status"];

}

// WiFiへの接続
void setupWiFi(){
    // connect wifi
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    while (WiFi.status() != WL_CONNECTED) {
        Serial.println(".");
        delay(100);
    }

    Serial.println("");
    Serial.print("Connected : ");
    Serial.println(WiFi.localIP());
    // sync Time
    configTime( 3600L * 9, 0, "ntp.nict.jp", "ntp.jst.mfeed.ad.jp");

    // MQTTブローカに接続
    client.setServer(MQTT_SERVER, MQTT_PORT);

    // 3sごとにセンサデータを送信する
    tickerMeasure.attach_ms(3000, sendSensorData);

    // MQTT subscribeの設定
    client.setCallback(mqttCallback);

}

void sendSensorData(void){
    //センサからデータの取得
    bme.readAllMeasurements(&measurements);
    Serial.println("Humidity,Pressure,BME-Temp");
    Serial.print(measurements.humidity, 0);
    Serial.print(",");
    Serial.print(measurements.pressure / 100, 2);
    Serial.print(",");
    Serial.println(measurements.temperature, 2);

    // ペイロードを作成して送信を行う。
    json_message.clear();
    json_message["humid"] = measurements.humidity;
    json_message["press"] = measurements.pressure / 100;
    json_message["temp"] = measurements.temperature;
    serializeJson(json_message, message_buffer, sizeof(message_buffer));
    client.publish(TOPIC, message_buffer);
}
```

```
// FastLED(CRGB type)
CRGB dispColor(uint8_t r, uint8_t g, uint8_t b) {
    return (CRGB)((r << 16) | (g << 8) | b);
}

void setup() {

    M5.begin(true, false, true); //UART, I2C, LED
    Serial.begin(115200);
    Wire.begin(26,32);

    bme.setI2CAddress(0x76); //address (0x76 or 0x77)
    if (bme.beginI2C() == false) //Begin communication over I2C
    {
        Serial.println("The sensor did not respond. Please check wiring.");
        while (1); //Freeze
    }

    // WiFi接続
    setupWiFi();
}

void loop() {
    client.loop();
    // MQTT未接続の場合は、再接続
    while(!client.connected() ){
        Serial.println("Mqtt Reconnecting");
        if( client.connect(DEVICE_ID) ){
            client.subscribe(TOPIC_STATUS);
            Serial.println("Mqtt Connected");
            break;
        }
        delay(1000);
    }

    switch (led_status)
    {
    case 0:
        M5.dis.drawpix(0, dispColor(255, 0, 0));
        break;
    case 1:
        M5.dis.drawpix(0, dispColor(0, 255, 0));
        break;
    case 2:
        M5.dis.drawpix(0, dispColor(0, 0, 255));
        break;
    case 3:
        M5.dis.drawpix(0, dispColor(255, 255, 255));
        break;
    default:

```

```
        break;  
    }  
  
    M5.update();  
}
```

Node-RED のFlowをデプロイします. **inject** ノードをクリックしてAtomのLEDの色が変化することを確認します.