

INTERNET OF THINGS

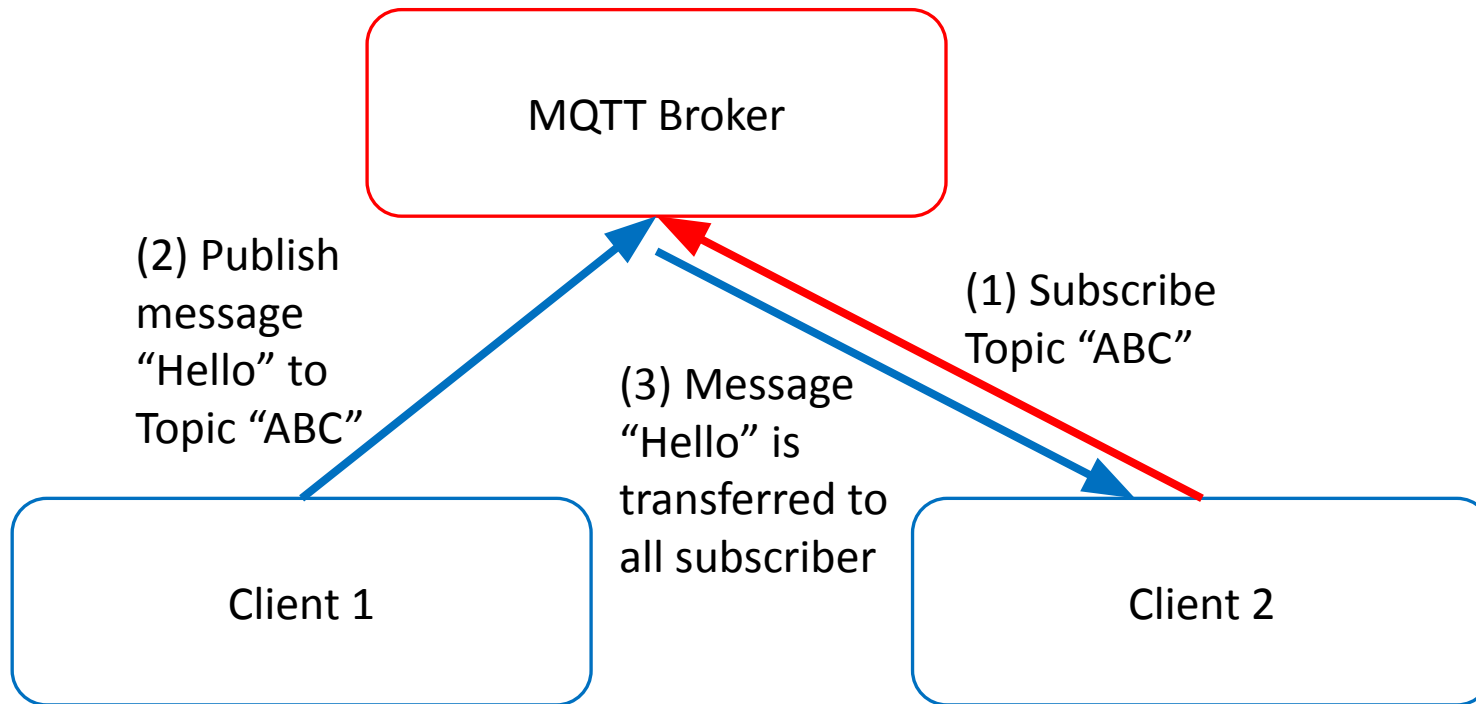
3.6

MQTT



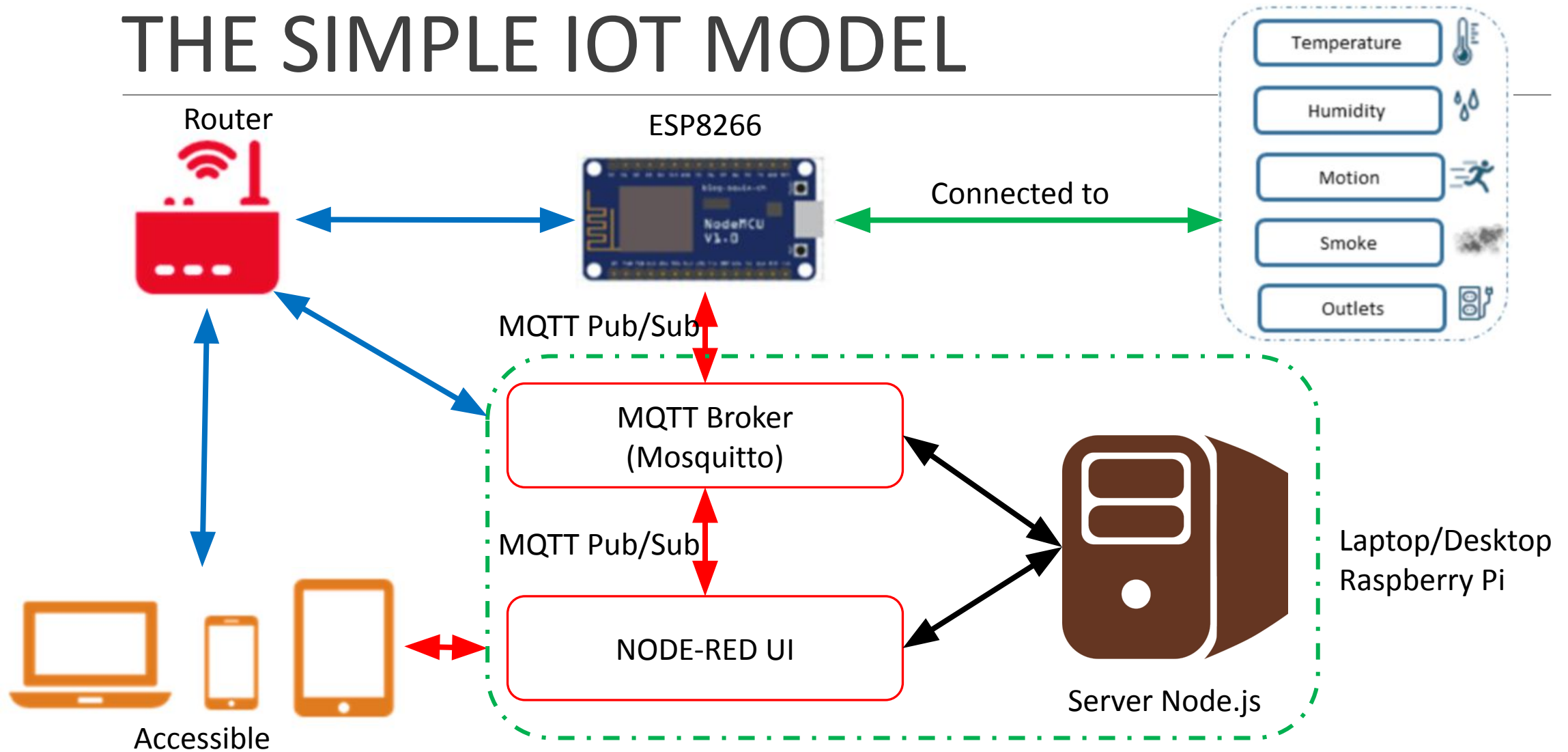
MQTT

WHAT IS MQTT?



- **MQTT (Message Queuing Telemetry Transport)** is a publish-subscribe network protocol that transports messages between devices.

THE SIMPLE IOT MODEL



INSTALL MOSQUITTO MQTT BROKEN

Install Mosquitto MQTT Broken
following by this instruction:

<https://mosquitto.org/download/>



START MOSQUITTO

- On Windows:
 - Open Command Prompt
 - Navigate to the Mosquitto root folder, such as “C:\Program Files (x86)\mosquitto”
 - Running the command: **net start mosquitto**
- On MacOS:
 - Open Terminal
 - Navigate to the Mosquitto root folder, such as “/usr/local/sbin/”
 - Running the command: **mosquitto -c /usr/local/etc/mosquitto/mosquitto.conf**

HELLO WORLD!

- Drag & Drop a “**mqtt in**” node and **Debug** node
- DoubleClick this node to **configure mqtt broker** and **subscribe a Topic**

Server: Add new mqtt broker... [dropdown arrow] [edit icon]

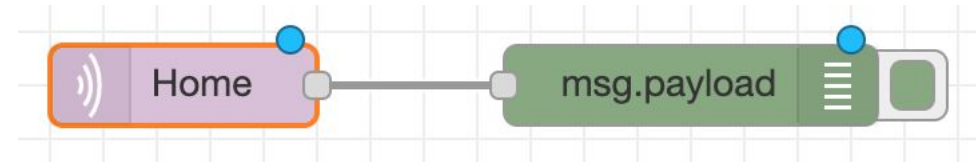
Topic: Home

QoS: 2 [dropdown arrow]

Output: auto-detect (string or buffer) [dropdown arrow]

Name: Name

Sample Topic: **home/kitchen/lamp**



Connection | Security | Messages

Server: localhost Port: 1883

☐ Enable secure (SSL/TLS) connection

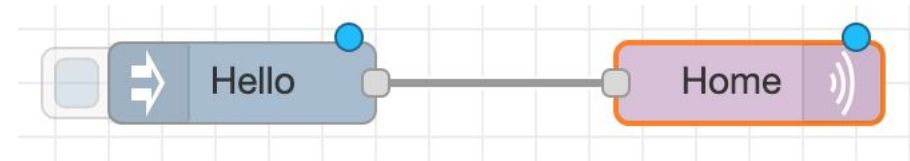
Client ID: Leave blank for auto generated

Keep alive time (s): 60 ☒ Use clean session

☐ Use legacy MQTT 3.1 support

HELLO WORLD! (tt)

- Drag & Drop a “**mqtt out**” node and **Inject** node
- DoubleClick this mqtt node to **configure mqtt broker** and **publish a Topic**

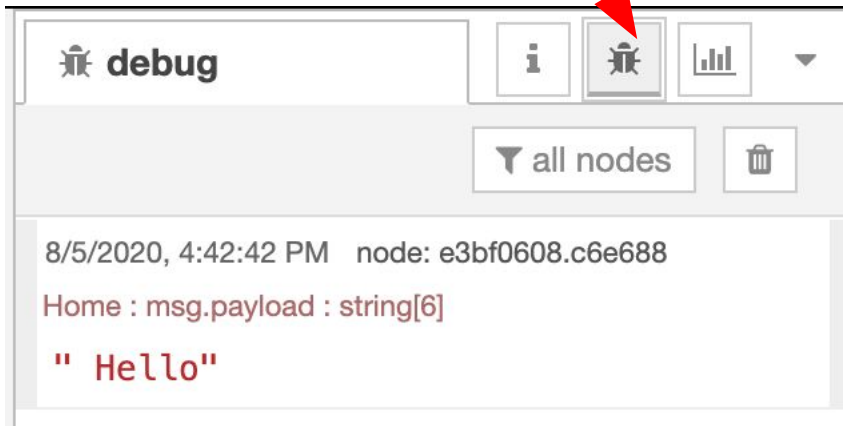
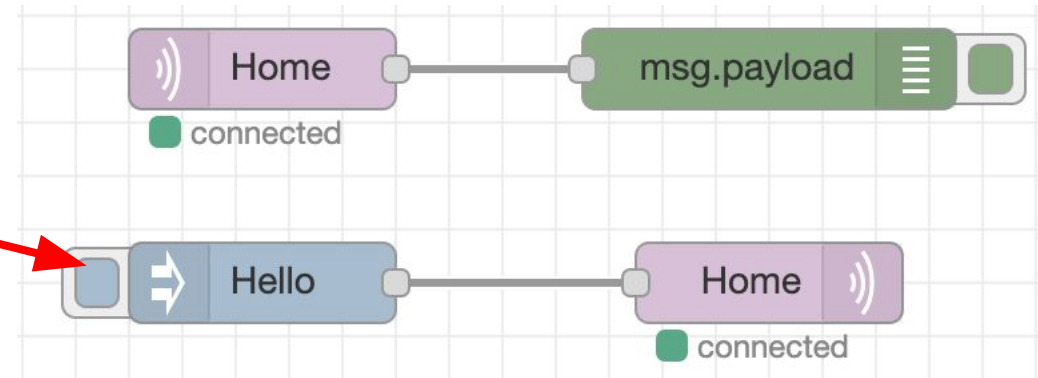


A configuration dialog for an MQTT node. It has four sections: 'Server' with a dropdown showing 'localhost:1883' and a pencil icon; 'Topic' with a text input containing 'Home'; 'QoS' with a dropdown and a 'Retain' checkbox; and 'Name' with a text input containing 'Name'. Two red arrows point to the 'Topic' and 'Server' fields respectively.

Server	localhost:1883	✓	✎
Topic	Home		
QoS		Retain	
Name	Name		

HELLO WORLD! (tt)

- Deploy
- Trigger Inject node
- View the result on Debug window

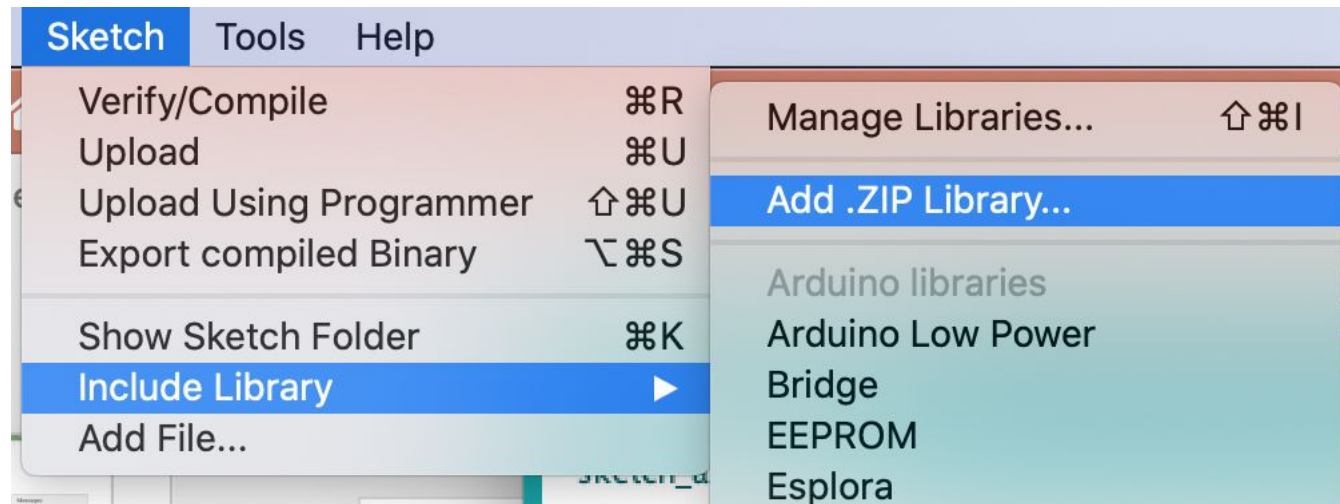


ESP8266 + NODERED



SETUP PUBSUBCLIENT LIBRARY

- **PubSubClient Library** provides a client for doing something publish/subscribe messaging with a server that supports MQTT.
- Download lib from here: <https://github.com/knolleary/pubsubclient/archive/master.zip>
- Import Zip file into the Arduino IDE.
- Restart Arduino IDE



Programming Instructions

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

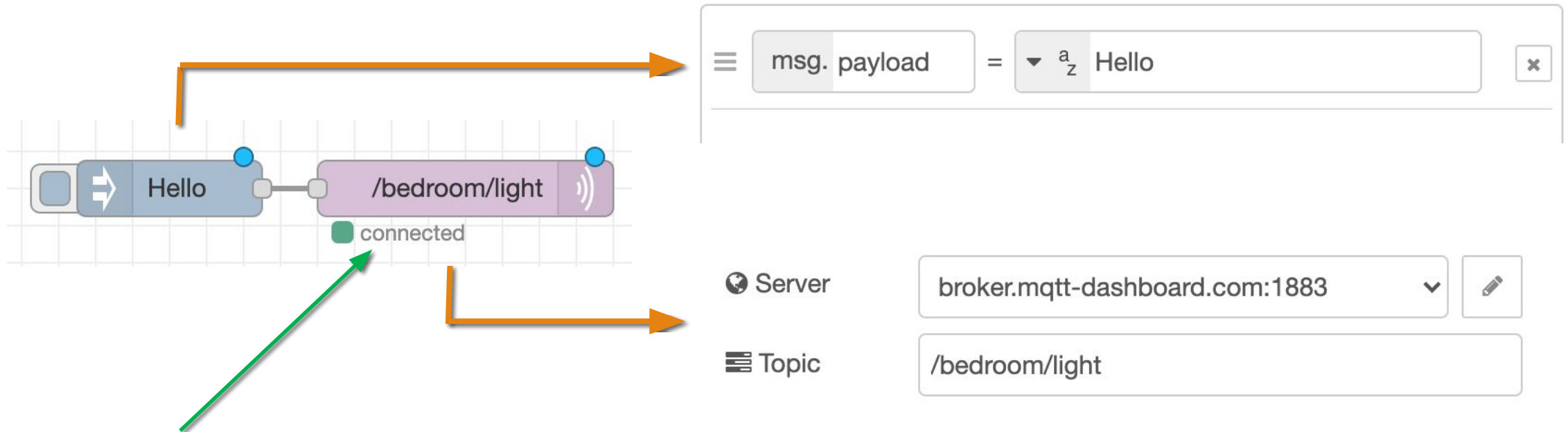
const char* mqtt_server = "broker.mqtt-dashboard.com";

void mqttCallback(char* topic, byte* payload, unsigned int length) {
    Serial.println(topic);
    Serial.println((char*)payload);
}
```

```
void reconnect() {  
    while (!client.connected()) {  
        String clientId = "ESP8266Client-";  
        clientId += String(random(0xffff), HEX);  
        if (client.connect(clientId.c_str())) {  
            Serial.println("connected");  
  
            // Subscribe all topics that your system need  
            client.subscribe("/bedroom/light");  
  
        } else {  
            Serial.println(" try again in 5 seconds");  
            delay(5000);  
        }  
    }  
}
```

```
void setup() {  
    Serial.begin(9600);  
  
    //Do it yourself. Connect to wifi by SSID or SoftAP  
  
    client.setServer(mqtt_server, 1883);  
    client.setCallback(mqttCallback);  
}  
  
void loop() {  
    if (!client.connected()) {  
        reconnect();  
    }  
    client.loop();  
}
```

Node-RED config



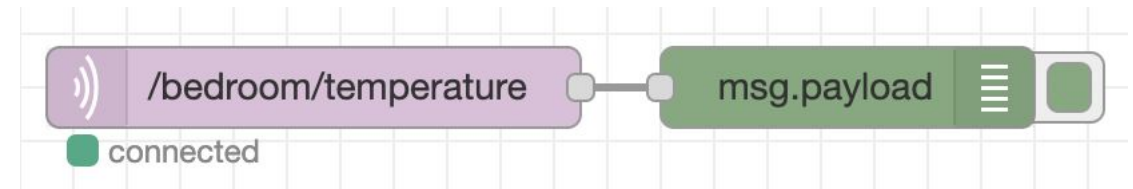
Notice: this topic name is exactly same in the ESP code

Publish to Node-RED

```
char msg[50];  
snprintf (msg, 50, "%ld", temperature);  
client.publish("/bedroom/temperature", msg);
```

ESP8266

Node-RED



Server

broker.mqtt-dashboard.com:1883

Topic

/bedroom/temperature

