



## 4.1

The background features a light gray circuit board pattern with various IoT-related icons in circular nodes. These include a radio tower, a water drop, a thermometer, a laptop, a server rack, a lightbulb, and a microchip. A large central circle contains a Wi-Fi symbol and the word 'MQTT' in a faint, light gray font.

# MQTT in ESP32 SIMULATOR

# WHAT IS MQTT?

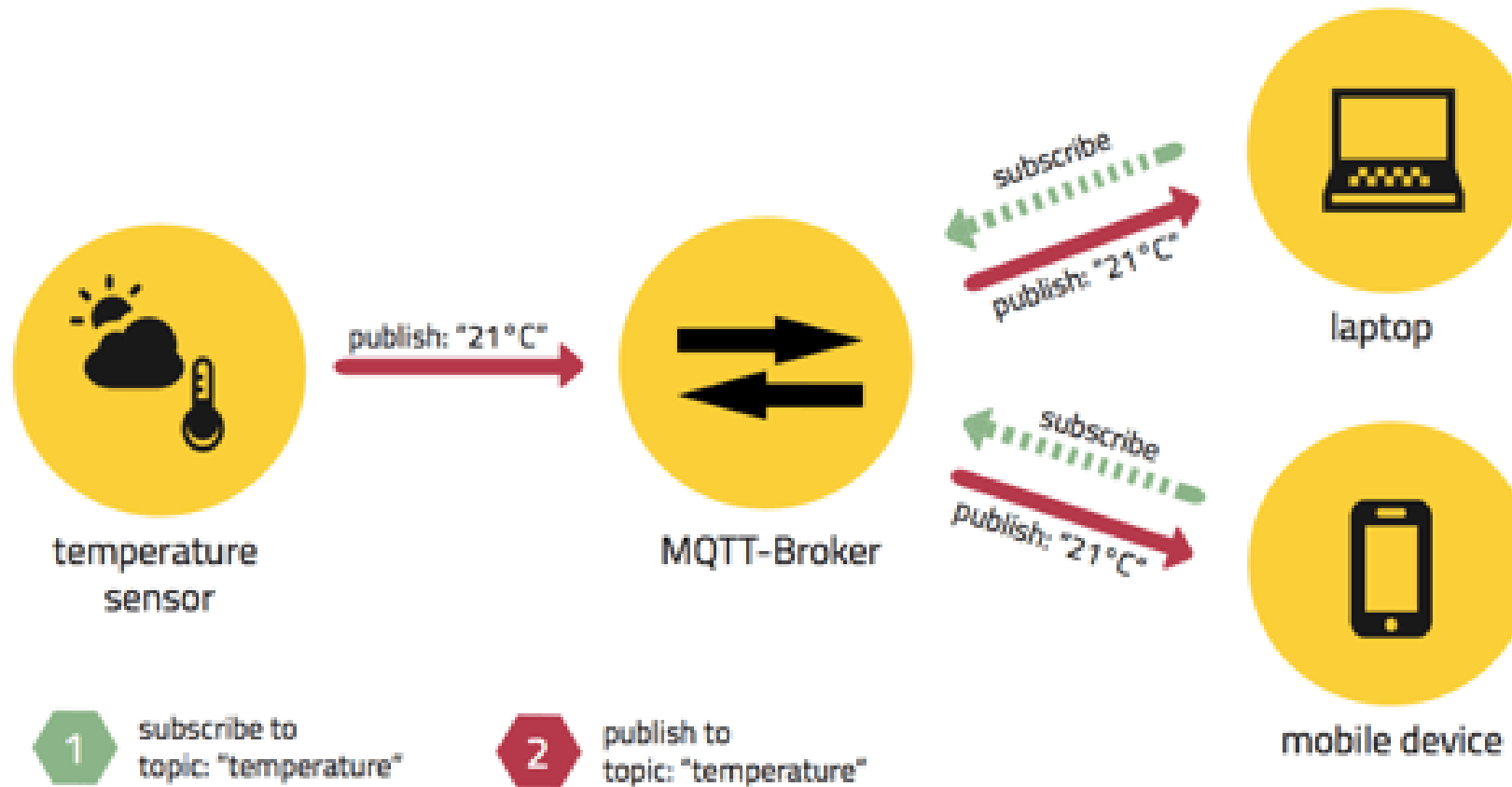
---



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

# HOW DOES MQTT WORK?

---



broker.mqtt-  
dashboard.com



test.mosquitto.org



broker.hivemq.com



# MQTT BROKER

---

The background features a light gray circuit board pattern with various IoT-related icons in circular frames: a radio tower, a water drop, a thermometer, a lightbulb, a microchip, a laptop, and server racks. A large central circle contains a Wi-Fi symbol and the text 'MQTT'.

# Send message to MQTT

1. Add *PubSubClient* Library
2. Include lib in program

```
#include <WiFi.h>
#include "PubSubClient.h" ←

const char* ssid = "Wokwi-GUEST";
const char* password = "";
const char* mqttServer = "test.mosquitto.org";
int port = 1883;

WiFiClient espClient;
PubSubClient client(espClient);
```

```
void wifiConnect() {
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println(" Connected!");
}

void setup() {
    Serial.begin(9600);
    Serial.print("Connecting to WiFi");

    wifiConnect();

    client.setServer(mqttServer, port);
}
```

Set unique-id  
your device

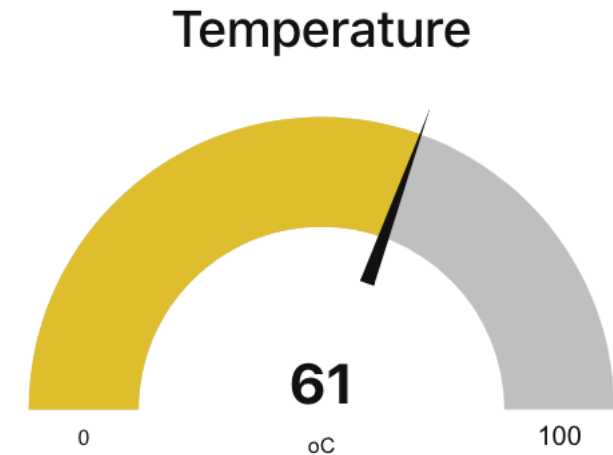
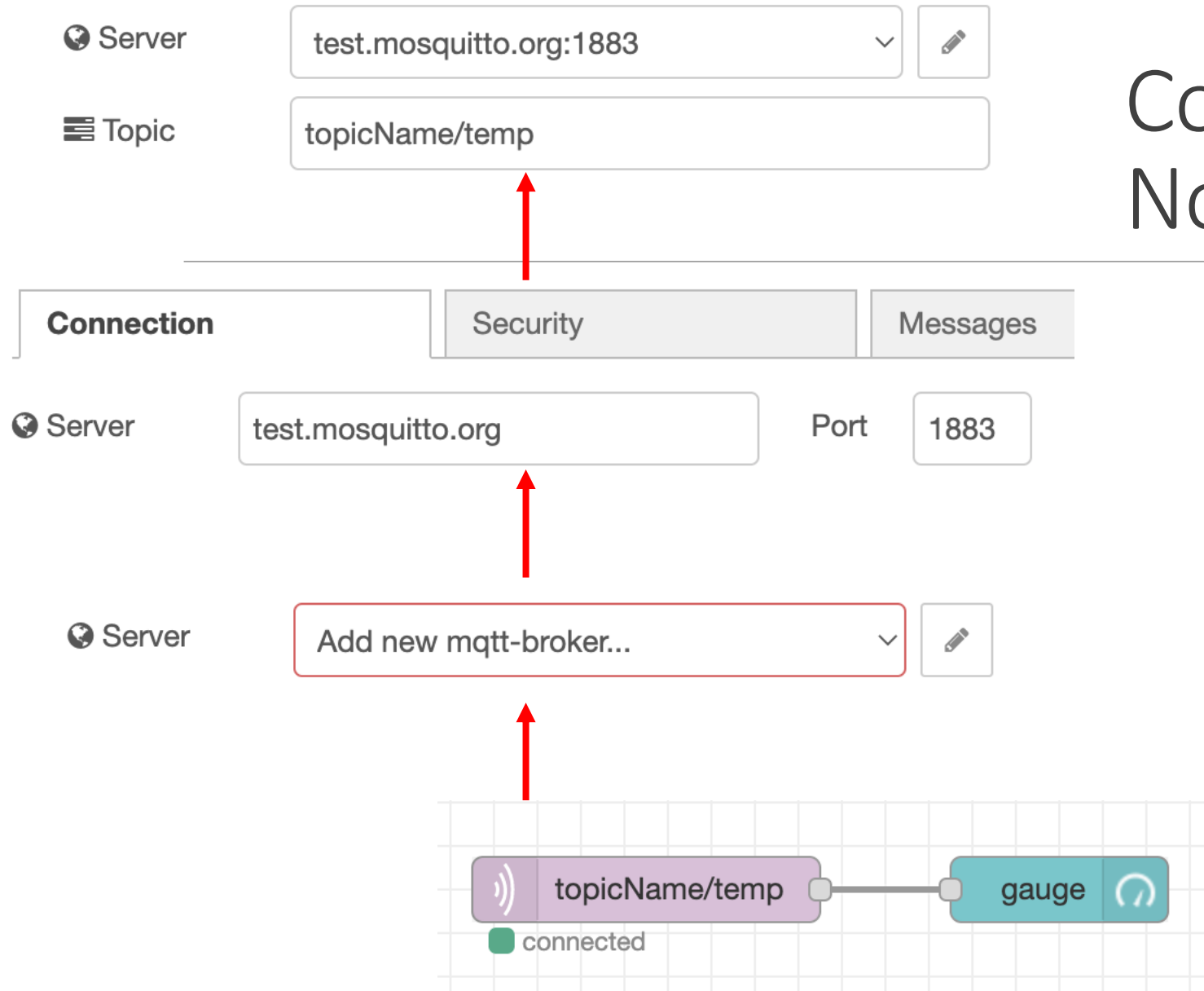


```
void mqttReconnect() {  
  while (!client.connected()) {  
    Serial.print("Attempting MQTT connection...");  
    if (client.connect("12345678")) {  
      Serial.println(" connected");  
    } else {  
      Serial.println(" try again in 5 seconds");  
      delay(5000);  
    }  
  }  
}
```

```
void loop() {  
  if (!client.connected()) {  
    mqttReconnect();  
  }  
  client.loop();  
  
  int temp = random(0, 100);  
  char buffer[50];  
  sprintf(buffer, "%d", temp);  
  client.publish("topicName/temp", buffer);  
  delay(5000);  
}
```



# Config “MQTT in” in Node-RED





Send real  
temperature  
from DHT22  
to Node-RED

---

# JSON

```
{  
  "temperature": 40,  
  "humidity": 10  
}
```

Label

Temperature

Value format

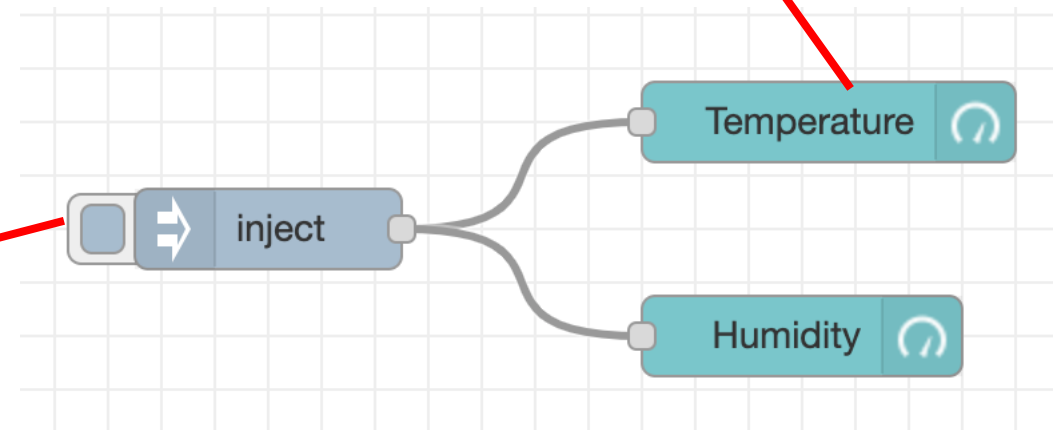
{{msg.payload.temperature}}

Units

oC

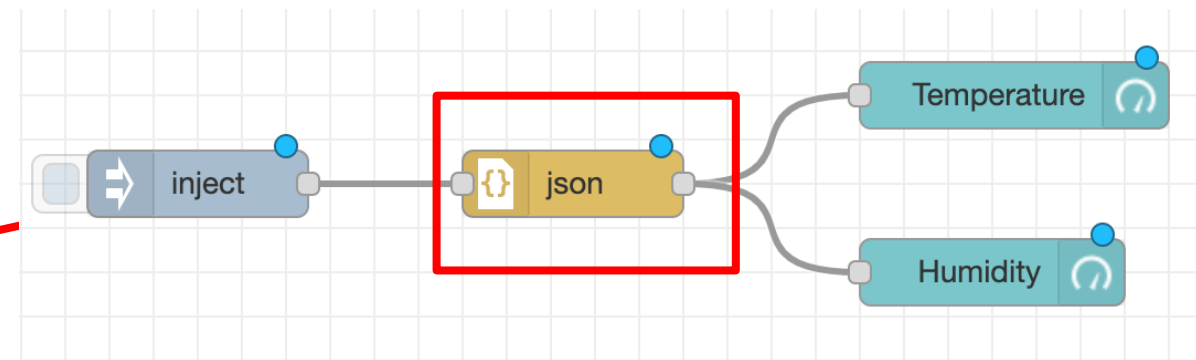
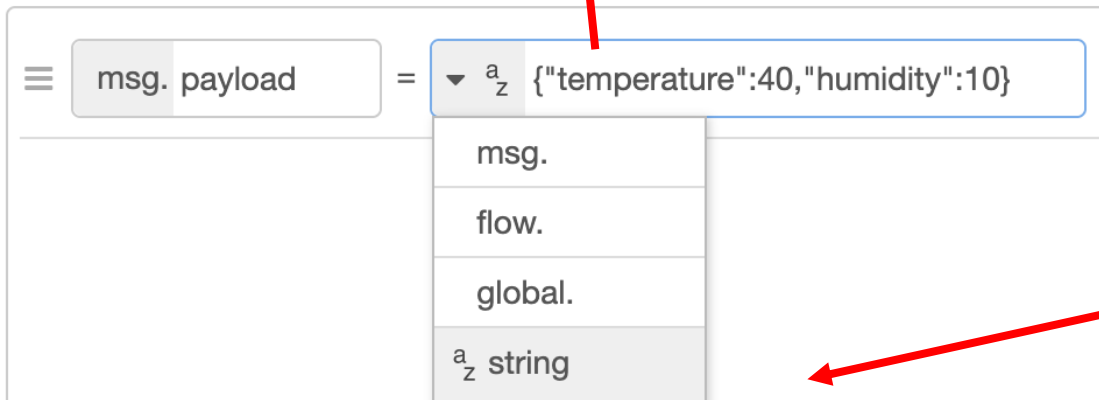
msg. payload = `{ "temperature":40,"humidity":10 }`

- msg.
- flow.
- global.
- string
- number
- boolean
- JSON



# Parse JSON string

`{"temperature":40,"humidity":10}`





Send both  
temperature  
and humidity  
from DHT22  
to Node-RED

---

```
char buffer[50];  
sprintf(buffer, "{\"temperature\":%d,\"humidity\":%d}", temp, humidity);  
client.publish("topicName/temp", buffer);
```


The background features a light blue gradient with a network of thin grey lines. Various circular icons are scattered around: a radio tower, a water drop, a thermometer, a lightbulb, a microchip, a laptop, and server racks. In the center, a large grey circle contains a white Wi-Fi symbol and the text 'MQTT' in a light, sans-serif font.

# Receive message from MQTT

Subscribe topic

```
void mqttReconnect() {  
  while (!client.connected()) {  
    Serial.print("Attempting MQTT connection...");  
    if (client.connect("12345678")) {  
      Serial.println(" connected");  
      client.subscribe("topicName/led");  
    } else {  
      Serial.println(" try again in 5 seconds");  
      delay(5000);  
    }  
  }  
}
```

```
void setup() {  
    Serial.begin(9600);  
    Serial.print("Connecting to WiFi");  
  
    wifiConnect();  
  
    client.setServer(mqttServer, port);  
    client.setCallback(callback);  
}
```



```
void callback(char* topic, byte* message, unsigned int length) {  
    Serial.println(topic);  
    String stMessage;  
    for (int i = 0; i < length; i++) {  
        stMessage += (char)message[i];  
    }  
    Serial.println(stMessage);  
}
```

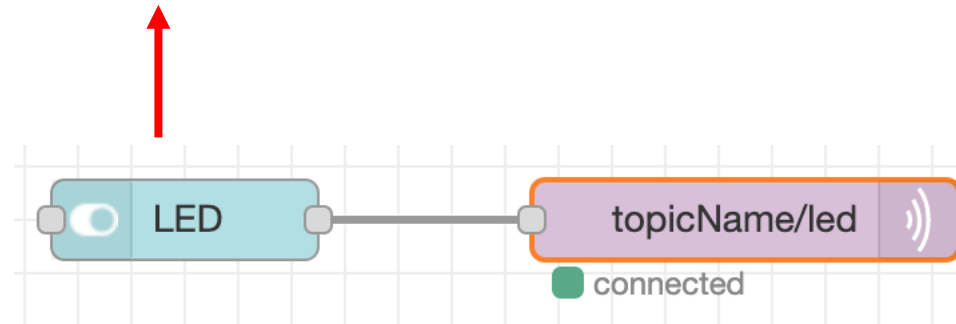


# Config “MQTT out” in Node-RED

☑ When clicked, send:

On Payload ▼ a<sub>z</sub> on

Off Payload ▼ a<sub>z</sub> off



```
void callback(char* topic, byte* message, unsigned int length) {  
  Serial.println(topic);  
  String stMessage;  
  for (int i = 0; i < length; i++) {  
    stMessage += (char)message[i];  
  }  
  Serial.println(stMessage);  
}
```

Diagram illustrating the mapping of the callback function to the MQTT configuration:

- The `topic` parameter is mapped to `topicName/led`.
- The `stMessage` variable is mapped to the `on` and `off` payload options.

🌐 Server

test.mosquitto.org:1883

📄 Topic

topicName/led



Turn LED  
on/off from  
Node-RED

---

## Send

Name \*  
abc

Content \*  
How are you?

SUBMIT

## Receive

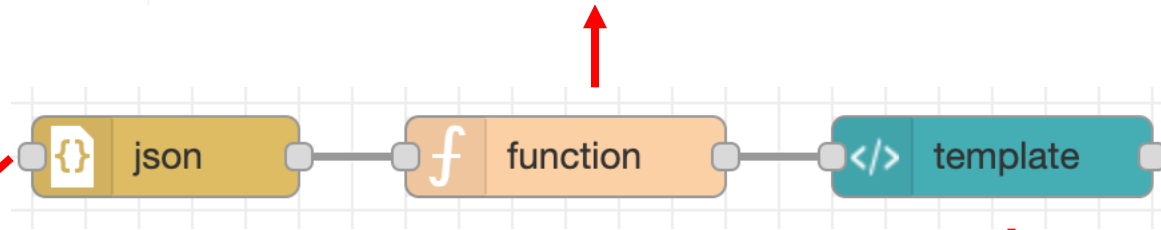
abc: Hello !!!  
def: Hello, I'm David  
abc: How are you?



# Chat App

# Hint

```
history = flow.get("history") || [];  
history.push(  
  {  
    "name":msg.payload.name,  
    "content":msg.payload.content  
  }  
);  
flow.set("history", history);  
msg.payload = history;  
return msg;
```



Assume we receive a JSON string of the following format:

```
{"name":"abc","content":"hello"}
```

Template

```
1 <div ng-repeat="x in msg.payload">  
2   <font color="red">{{x.name}}: </font>  
3   <font color="black">{{x.content}}</font>  
4 </div>  
5
```