How to smarten up your mechatronic projects

Internet of Things with MQTT, Node-Red

Author: Prof. Dr. Markus Graf
Revision: 1.0
Date: 2022-04-01
Framework & source code by Eduard Junimann, Markus Graf
Platform: RaspberryPi (3B+), ESP, RaspAP, Arduino

Introduction

Internet of things and industry 4.0 – as well as the corresponding smart products, and home automation as a new technology emerged with the rise of digitalization and network communication between smallest devices possible. Technological advances in microprocessor platforms and mini computers provided their share in this field.

In this set of tutorials you are going to develop distributed smart products, controllable via intra- or internet as well by the click on your smartphone. Have no fear, we will start slow, we also provide an initial platform to have a head start into the technology which is now forming the new concept of networked matter.

Okay, let's start.

Prerequisites

Before you can start, please get the following electronic parts:
A RaspberryPi (any v. 3 or v. 4 will do), download the iot-image from our repository or use a computer and install your own mqtt broker service integrated into your network.
Additionally it is recommended to install node-red on any device within the computer network (in our image it is on the same mini computer raspberry pi as the router).

Using the raspberry pi and the preinstalled image:
You can login into the system using **IoT-Gateway** and the initial password **smartproducts**.
For the safety of your network please provide other login credentials (at least change the password to something more safe).

Grab a microprocessor platform like Arduino, ESP 8266 or anything alike you are familiar with. We recommend using an ESP 8266 as our demo-code is prepared to run on those devices. From here on we can start right ahead of getting the following first code and project to work.

1. Hello world

The project consists of the following major steps/parts:

a. Prepare the bi-directional communication via MQTT
b. Implement a simple "hello world" application on the microprocessor (controller) side
c. Create a node-red server side system with the logic
d. Run it

a. Prepare MQTT communication and server

In order to have the microprocessor talk to the service, we send and receive messages to and from the MQTT broker. Node-red is also connected to this broker which allows the system's logic later on to be developed within node-red.

Connect to the RaspberryPi AccessPoint (10.3.141.1):

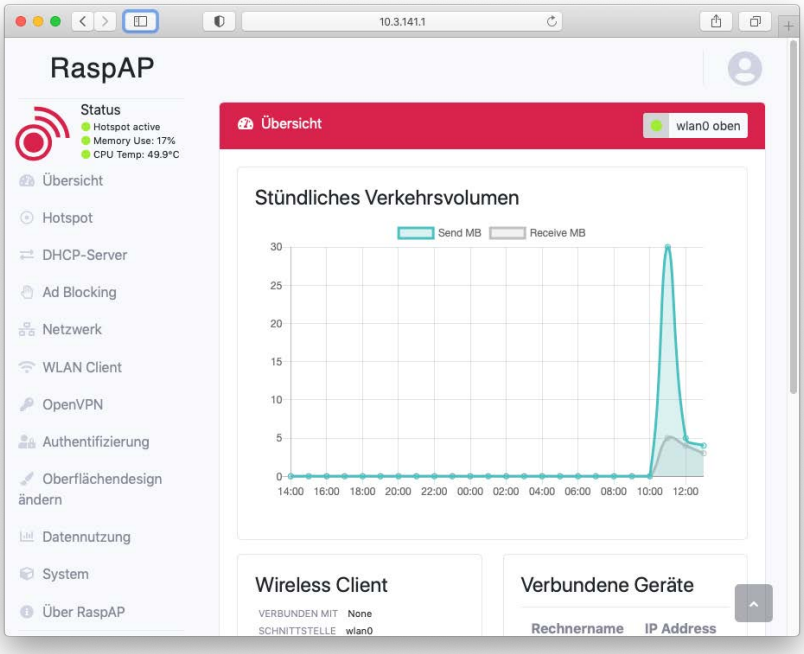

Figure 1 : Raspberry Accesspoint (on http://10.3.141.1)

Here you can control the WiFi-HotSpot, Password-settings, IP addresses in case you need to reconfigure something, but please be aware to also adjust the template code for the microprocessor, the **MQTT-Broker** connection of **mosquitto**. In **mosquitto** there is a configuration program running on the RaspberryPi: *mosquitto_passwd* to create a password file, this has to be put under */etc/mosquitto/* so that *mosquitto.conf* can reference to it. The mosquitto broker is available on 10.3.141.1:1883

The more interesting part is the node-red environment which will later contain the logic and the controls / display part of the system. It can be reached online via http://10.3.141.1:1880/

b. Microcontroller implementation

For the purpose to establish the communication, we have to connect our devices to the broker from the microprocessor using the following code.

First, include necessary libraries for the Arduino development environment, here we use Wifi-connecting module and also the publisher-subsciber client module (for the MQTT communication).

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

Then we need all connection credentials for the MQTT broker system's WIFI and broker.

```
String clientId = "ESP8266-01";
const char* SSID = "IoT-Gateway";
const char* PSK = "smartproducts";
```

```
const char* MQTT_BROKER = "10.3.141.1";
const char* mqtt_user = "IoT-Gateway";
const char* mqtt_password = "smartproducts";
```

Create some MQTT topics for the communication, with these messages will be distinguished so that the server can react to specific application messages and sensor data.

```
// MQTT message topics
#define status_topic "helloworld/status"
#define message_topic "helloworld/message"
#define answer_topic "helloworld/answer"
```

We begin with starting up the wifi-connection and establish publisher-subscribing protocol via the wifi connection.

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

void setup_wifi() {
  delay(10);
  WiFi.begin(SSID, PSK);
  while (WiFi.status() != WL_CONNECTED) {
    delay(200);
    Serial.print(".");
  }
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Establishing MQTT connection...");
    if (client.connect(clientId.c_str(), mqtt_user, mqtt_password)) {
      Serial.println("success: connected");
      client.publish(status_topic, "ESP01 alive");
    } else {
        Serial.print("failed, rc=");
        Serial.print(client.state());
        Serial.println("... trying again in 5 seconds");
        delay(5000);
    }
  }
}
```

Establish the connection via WIFI to the broker.

In the setup routine we connect by calling the setup_wifi function as well initially set the information to the MQTT publisher-subscriber client.

```
void setup() {
  Serial.begin( 115200 );
  setup_wifi();
  client.setServer( MQTT_BROKER, 1883 );
  client.setCallback(mqtt_callback);
}
```

Inside the main loop the connection of to the MQTT broker is constantly checked. If it isn't a reconnect will be executed.
Important notice: If various similar devices should be running the same peace of software, a

distinguishing element should be integrated onto the systems. This could be a random number, however, the mac-address of the device would be the most convenient solution (*Therefore, please see how to append the mac address, that you could put into the setup-routine*):

```
      String clientId = "ESP8266-01";
      uint8_t mac[6];
      WiFi.macAddress(mac);
      clientName += macToStr(mac);
      clientName += "-";
      clientName += String(micros() & 0xff, 16);
```

Please refer here to the the main loop of the initial example.

```
void loop()
{
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  delay(500);     // just for now, as you know I prefer elapsed times

  message = “Hello World!”;
  client.publish("usermessage", message );
}
```

Send and receive data for the hello world example

First, we add the subscription to the answer from the MQTT broker server.

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Establishing MQTT connection...");
    if (client.connect(clientId.c_str(), mqtt_user, mqtt_password)) {
      Serial.println("success: connected");
      client.publish(status_topic, "ESP01 alive");
      Serial.println("Subscribing to ");
      Serial.println( answer_topic );
      client.subscribe(answer_topic);
    } else {
        Serial.print("failed, rc=");
        Serial.print(client.state());
        Serial.println("... trying again in 5 seconds");
        delay(5000);
    }
  }
}
```

c. Node-Red flow

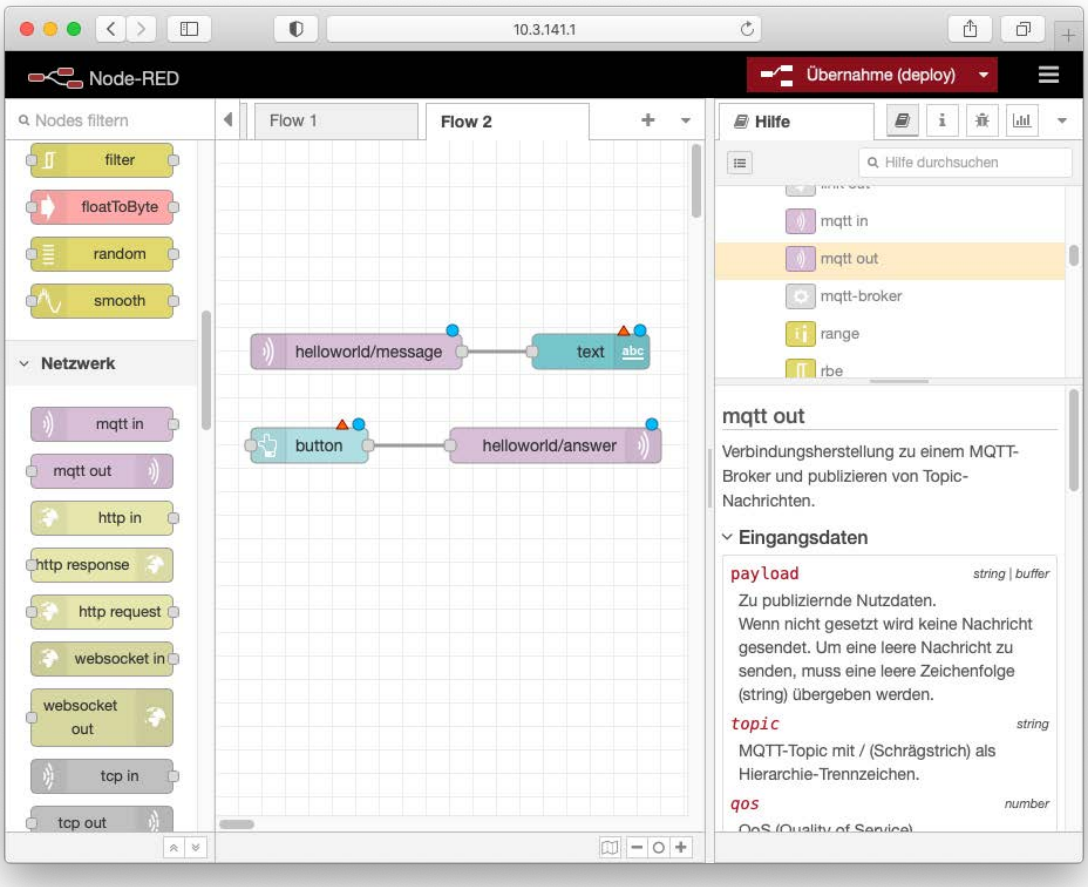

Figure 2: Node-Red flow

Add the appropriate nodes to the flow and create the necessary dashboard.

d. Run it

Afterwards connect to the dashboard in a browser on

http://10.3.141.1:1880/ui

Start the microprocessor and see what happens.

2. Add Sensordata

Let's add the sensor data to the Arduino (.ino) file and also to the flow.

First, create the file *IoT_DHTExample.ino* by copying the content of the previous HelloWorld example.
Next, **add the necessary DHT library**, attach the correct pins and type to the DHT-sensor object while instantiating it.

```
#include <DHT.h>

#define DHT_PIN 14
#define DHT_TYPE DHT22

DHT dhtSensor(DHT_PIN, DHT_TYPE);
```

Modify the messages to fit the new example in order to create a specific flow component on Node-Red like this:

```
#define status_topic "dht/status"
#define temperature_topic "dht/temperature"
#define humidity_topic "dht/humidity"
#define answer_topic "dht/answer"
```

Add the setup of the DHT sensor to the setup-routine:

```
void setup() {
  Serial.begin( 115200 );
  setup_wifi();

  dhtSensor.begin();

  client.setServer( MQTT_BROKER, 1883 );
  client.setCallback(mqtt_callback);
}
```

In the loop modify the part where messages will be published to the MQTT broker into reading the values and then pushing them to the MQTT broker:

```
  if ( current_ms - last_sent_ms > 1000 )
  {
    float temperature = dhtSensor.readTemperature();
    float humidity = dhtSensor.readHumidity();

    client.publish( temperature_topic, String(temperature).c_str() );
    client.publish( humidity_topic, String(humidity).c_str() );
    last_sent_ms = current_ms;
  }
```

That's it. The complete code can be downloaded from the repository or seen in Listing 2.

Assignment #1:

Create a flow within node-red in order to display a dashboard with temperature and humidity controls.
In case you need further help, please check out the video tutorials on YouTube:
https://youtube.com/playlist?list=PLZYUq1aSlh3XkbbCMmhEKwqkyWMVRy6qM

Assignment #2:

Integrate a switch or a button into the flow in order to remotely control the program on the ESP (for example to switch on/off the temperature measurement.

Attachment 1

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

// connection details
String clientId = "ESP8266-01";
const char* SSID = "IoT-Gateway";
const char* PSK = "smartproducts";
const char* MQTT_BROKER = "10.3.141.1";
const char* mqtt_user = "IoT-Gateway";
const char* mqtt_password = "smartproducts";

WiFiClient espClient;
PubSubClient client(espClient);
long last_sent_ms = millis();

// MQTT message topics
#define status_topic "helloworld/status"
#define message_topic "helloworld/message"
#define answer_topic "helloworld/answer"

void setup_wifi() {
  delay(10);
  Serial.print("Connecting to "); Serial.println(SSID);
  WiFi.begin(SSID, PSK);
  while (WiFi.status() != WL_CONNECTED) {
    delay(200);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected with IP address: ");
  Serial.println(WiFi.localIP());
}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Establishing MQTT connection...");
    if (client.connect(clientId.c_str(), mqtt_user, mqtt_password)) {
      Serial.println("success: connected");
      client.publish(status_topic, "ESP01 alive");
      Serial.println("Subscribing to ");
      Serial.println( answer_topic );
      client.subscribe(answer_topic);
    } else {
        Serial.print("failed, rc=");
        Serial.print(client.state());
        Serial.println("... trying again in 5 seconds");
        delay(5000);
    }
  }
}

void mqtt_callback(char* topic, byte* message, unsigned int length) {
  Serial.print("Nachricht eingetroffen [");
  Serial.print(topic);
  Serial.print("]: ");
  String completeMessage;

  for (int i=0;i<length;i++) {
```

```
    completeMessage += (char)message[i];
  }
  Serial.println("Message received: ");
  Serial.println( completeMessage );
  Serial.println();
}


void setup() {
  Serial.begin( 115200 );
  setup_wifi();
  client.setServer( MQTT_BROKER, 1883 );
  client.setCallback(mqtt_callback);
}


void loop() {
  if ( !client.connected() )
  {
    reconnect();
  }
  client.loop();

  long current_ms = millis();

  if ( current_ms - last_sent_ms > 1000 )
  {
    String message = "Hello World! " + String( current_ms );
    client.publish( message_topic, message.c_str() );
    last_sent_ms = current_ms;
  }
}
```

Listing 1 - Hello World IoT for the microcontroller

Attachment 2

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

#define DHT_PIN 14
#define DHT_TYPE DHT22

DHT dhtSensor(DHT_PIN, DHT_TYPE);

// connection details
String clientId = "ESP8266-01";
const char* SSID = "IoT-Gateway";
const char* PSK = "smartproducts";
const char* MQTT_BROKER = "10.3.141.1";
const char* mqtt_user = "IoT-Gateway";
const char* mqtt_password = "smartproducts";

WiFiClient espClient;
PubSubClient client(espClient);
long last_sent_ms = millis();

// MQTT message topics
#define status_topic "dht/status"
#define temperature_topic "dht/temperature"
#define humidity_topic "dht/humidity"
#define answer_topic "dht/answer"

void setup_wifi() {
  delay(10);
  Serial.print("Connecting to "); Serial.println(SSID);
  WiFi.begin(SSID, PSK);
  while (WiFi.status() != WL_CONNECTED) {
    delay(200);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected with IP address: ");
  Serial.println(WiFi.localIP());
}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Establishing MQTT connection...");
    if (client.connect(clientId.c_str(), mqtt_user, mqtt_password)) {
      Serial.println("success: connected");
      client.publish(status_topic, "ESP01 alive");
      Serial.println("Subscribing to ");
      Serial.println( answer_topic );
      client.subscribe(answer_topic);
    } else {
        Serial.print("failed, rc=");
        Serial.print(client.state());
        Serial.println("... trying again in 5 seconds");
        delay(5000);
    }
  }
}
```

```
void mqtt_callback(char* topic, byte* message, unsigned int length) {
  Serial.print("Nachricht eingetroffen [");
  Serial.print(topic);
  Serial.print("]: ");
  String completeMessage;

  for (int i=0;i<length;i++) {
    completeMessage += (char)message[i];
  }
  Serial.println("Message received: ");
  Serial.println( completeMessage );
  Serial.println();
}


void setup() {
  Serial.begin( 115200 );
  setup_wifi();
  dhtSensor.begin();
  client.setServer( MQTT_BROKER, 1883 );
  client.setCallback(mqtt_callback);
}


void loop() {
  if ( !client.connected() )
  {
    reconnect();
  }
  client.loop();

  long current_ms = millis();

  if ( current_ms - last_sent_ms > 1000 )
  {
    float temperature = dhtSensor.readTemperature();
    float humidity = dhtSensor.readHumidity();

    client.publish( temperature_topic, String(temperature).c_str() );
    client.publish( humidity_topic, String(humidity).c_str() );
    last_sent_ms = current_ms;
  }
}
```

Listing 2 - IoT-DHTExample.ino – sensor reading and publishing for the Arduino / ESP