MQTT Doku

Tobias Reichel, Martin Neumann, Lukas Miller

08.06.2020

Contents:

Hochschule Augsburg
Fakultät für Informatik
Veranstaltung IOT (Prof. Dr. Volodymyr Brovkov)
Sommersemester 2020

Teammitglieder

Tobias Reichel, #2022398, INF6, <Tobias.Reichel@hs-augsburg.de
Martin Neumann, #2026143, INF6, <Martin.Neumann1@hs-augsburg.de
Lukas Miller, #, INF6, <Lukas.Miller@hs-augsburg.de

KAPITEL 1

Funktionsweise des Projekts

Temperaturen via Python Max Min via Website Wemos verarbeitet Daten und sendet zustand heizung an Website.

Website läuft auf https://www.hs-augsburg.de/homes/schnecke/mqtt/ Git hier https://r-n-d.informatik.hs-augsburg.de:8080/schnecke/mqtt

KAPITEL 2

Python

KAPITEL 3

Publishing der Heizungslimits mittels Javascript

3.1 Grundlagen

Als Publisher für die Heizungslimits wurde eine Website mittels Javascript und HTML5 erstellt.

Als MQTT-Implementierung wurde Eclipse Paho für verwendet. Dieses wird über https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.js bereitgestellt. Man bindet die Implementierung im HTML mittels eines `script`-Tags ein. Als zweites Script wurde im HTML das `ui.js` importiert, welches die Werte published sowie den Werten subscriped, welches das WEMOS Modul sendet.

3.2 HTML

Als Benutzeroberfläche wurde eine sehr einfache Website erzeugt. Diese Website enthält zwei Regler, mit welchen die mindest und maximal Temperatur gesetzt werden kann. Außerdem gibt es einen Button, der das Senden der Werte ermöglicht, und eine Fläche, um die Informationen über den Zustand der Heizung, welche vom WEMOS Modul übertragen werden, darzustellen.

Auf der Oberfläche ist es möglich, den maximal Wert niedriger als den minimal Wert zu setzen, beim Absenden der Werte wird aber eine Fehlermeldung angezeigt und die Werte nicht gesendet.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <script
      src="https://cdnjs.cloudflare.com/ajax/libs/paho-mqtt/1.0.1/mqttws31.
↪js"
      type="text/javascript"
    ></script>
    <script src="ui.js" type="text/javascript" defer></script>

    <meta charset="utf-8" />
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" /
↪>
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    <title>MQTT</title>
  </head>
  <body>
    <p>Temperature low:</p>
    <input id="low" type="range" min="1" max="40" onchange="DragEnd()"/>

    <p>Temperature high:</p>
    <input id="high" type="range" min="1" max="40" onchange="DragEnd()"/>

    <p id='lblLow'>Low:</p>
    <p id='lblHigh'>High:</p>

    <p><button onclick="Send()">Send</button></p>

    <p id='status'></p>
  </body>
</html>
```

Nachfolgenden sieht man ein Screenshot, wie die Website im Betrieb aussieht.

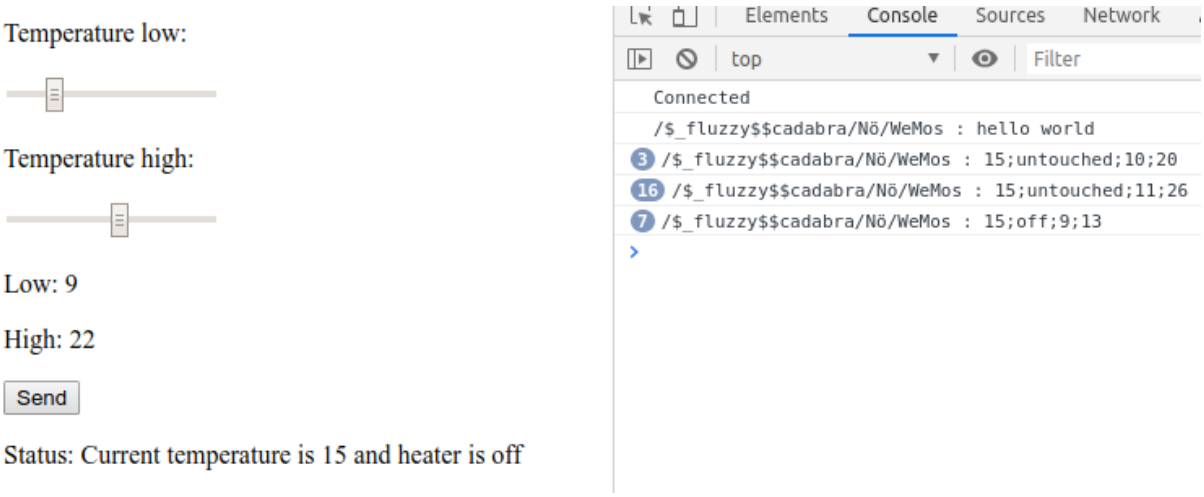

Abb. 1: Website sendet und empfängt Daten vom Broker

3.3 Javascript

Als Broker wurder `test.mosquitto.org` verwendet. Es konnte aber nicht der unverschlüsselte Port 1883 benutzt werden, sondern der für WebSockets vorbereitete verschlüsselte Port 8081.

Es gibt Funktionen, um sich mit dem Broker zu verbinden und im Fall des Verbindungsverlust eine neue Verbindung aufzubauen. Außerdem gibt es Funktionen, um die Werte bevor sie an den Broker gesendet werden aufzubereitet, sowie eine Funktion, um auf eingehende Nachrichten vom WEMOS zu reagieren und im HTML anzuzeigen.

In der Funktion `send()` wird zudem überprüft, ob der maximal Wert niedriger als der minimal Wert ist und dem entsprechend eine Fehlermeldung ausgegeben. Die Werte werden an den Broker im Format `low;high` übertragen. Einstellige Temperaturen werden mit führenden Nullen aufgefüllt.

Eingehende Nachrichten vom WEMOS werden in Temperatur und Zustand der Heizung gesplitted und im HTML angezeigt. Da das WEMOS Modul den Zustand der Heizung mit `on`, `off` und `untouched`

überträgt, können die Werte direkt in den String, welcher im HTML angezeigt wird eingebaut werden.

```
/*
 * MQTT-WebClient example for Web-IO 4.0
 */
var hostname = "test.mosquitto.org";
var clientId = "mqttJs";
var port = 8081;
clientId += new Date().getUTCMilliseconds();
var subscription = "/$_fluzzy$$cadabra/Nö/Heizung";
var subscription1 = "/$_fluzzy$$cadabra/Nö/WeMos";

mqttClient = new Paho.MQTT.Client(hostname, port, clientId);
mqttClient.onMessageArrived = MessageArrived;
mqttClient.onConnectionLost = ConnectionLost;
Connect();

function Connect() {
  mqttClient.connect({
    onSuccess: Connected,
    onFailure: ConnectionFailed,
    useSSL: true,
  });
}

function Connected() {
  console.log("Connected");
  mqttClient.subscribe(subscription1);
}

function ConnectionFailed(res) {
  console.log("Connect failed:" + res.errorMessage);
}

function ConnectionLost(res) {
  if (res.errorCode !== 0) {
    console.log("Connection lost:" + res.errorMessage);
    Connect();
  }
}

/*Callback for incoming message processing */
function MessageArrived(message) {
  console.log(message.destinationName + " : " + message.payloadString);
  let payloadArray = message.payloadString.split(';');
  let temp = payloadArray[0];
  let heater = payloadArray[1];
  document.getElementById('status').innerHTML =
  `Status: Current temperature is ${temp} and heater is ${heater}`;
}

function Send() {
  let low = parseInt(document.getElementById("low").value);
  let high = parseInt(document.getElementById("high").value);

  if (low > high) {
    alert("Low higher then High!");
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
  } else {
    var message = new Paho.MQTT.Message(
      PadZero(low, 2) + ";" + PadZero(high, 2)
    );
    message.destinationName = "/$_fluzzy$$cadabra/Nö/Heizung";
    mqttClient.send(message);
  }
}

function DragEnd() {
  let low = document.getElementById("low").value;
  let high = document.getElementById("high").value;

  document.getElementById("lblLow").innerText = "Low: " + low;
  document.getElementById("lblHigh").innerText = "High: " + high;
}

function PadZero(n, size) {
  let s = n + "";
  while (s.length < size) s = "0" + s;
  return s;
}
```

KAPITEL 4

Wemos

4.1 Voraussetzungen

Die Hardware besteht aus einem **IZOKEE ES8266 WIFI Modul**. Per MicroUSB ist das Modul mit einem Notebook verbunden worden. Zusätzlich wurde ein WLAN in Form eines Hotspots erstellt, welches im Rahmen des Projekts als Kommunikationsweg mit einem MQTT-Broker dient.

Programmiert wurde mittels einer Arduino-Oberfläche in der Sprache C. Die Testumgebung bestand aus einer MQTTBox.

4.2 Vorlage

Als Vorlage der umgesetzen Projektarbeit dient das bereits in der Vorlesung besprochene „WeMosMQTTSubPub_Projekt5“. Einzelne Codeabschnitte konnten hierfür verwendet werden und mussten nicht mehr selbst implementiert werden.

4.3 Implementierung

4.3.1 Konfiguration Connection

WLAN Konfiguration

```
const char* ssid = "SSID";
const char* password = "PASSWORD";
```

MQTT Konfiguration

```
const char* mqtt_server = "test.mosquitto.org";
const char* topic0 = "/$_fluzzy$$cadabra/Nö/Heizung";
const char* topic1 = "/$_fluzzy$$cadabra/Nö/Temperatur";
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
const char* topic2 = "/$_fluzzy$$cadabra/Nö/WeMos";
String clientId = "WeMos_";
char* clientID_c_str = "12345678901234567890123456789012345678";
```

4.3.2 Startwerte definieren

```
//actual temperature
int temp = 15;

//temperature limits
int bottomLimit = 10;
int upperLimit = 20;
```

4.3.3 Callback Function

Konvertierung der Grenzen der Heizung (Input der Webanwendung)

```
if(strcmp(topic, "/$_fluzzy$$cadabra/Nö/Heizung") == 0) {
    bottomLimit = 10*(payload[0]-'0') + (payload[1]-'0');
    upperLimit = 10*(payload[3]-'0') + (payload[4]-'0');
}
```

Konvertierung der aktuellen Temperatur (Input der Pythonanwendung)

```
if(strcmp(topic, "/$_fluzzy$$cadabra/Nö/Temperatur") == 0) {
    for (int i = 0; i < length; i++) {

        if(payload[0] != '-'){                  //negativ values
            if(i==0 && length > 1){             //double digit
                temp = 10*(payload[i]-'0');
            }
            else if (i==0){                     //single digit
                temp = (payload[i]-'0');
            }
            else{
                temp = temp + payload[i]-'0';
            }
        }
        else{                                   //positive values
            if(i==1 && length > 2){             //double digit
                temp = 10*(payload[i]-'0');
            }
            else if (i==1){                     //single digit
                temp = (payload[i]-'0');
            }
            else{
                temp = temp + payload[i]-'0';
            }
        }
    }
    if(payload[0] == '-'){                      //add minus in front of␣
↪negativ value
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
        temp = -temp;
    }
}
```

4.3.4 Verbindungsaufbau

Anbindung an die richtigen Subscriber und Publisher

```
client.publish(topic2, "hello world");
client.subscribe(topic0);
client.subscribe(topic1);
```

4.3.5 Abfrageroutine

Auswertung der aktuellen Temperatur alle 5 Sekunden

```
if (temp < bottomLimit) {
    snprintf (msg, 20, "%ld;on;%ld;%ld",temp,bottomLimit,upperLimit);
  }
  else if (temp > upperLimit){
    snprintf (msg, 20, "%ld;off;%ld;%ld",temp,bottomLimit,upperLimit);
  }
  else {
    snprintf (msg, 20, "%ld;untouched;%ld;%ld",temp,bottomLimit,
↪upperLimit);
  }

  client.publish(topic2, msg);
```

4.4 Testing

Nachfolgend ist ein Versuchsaufbau in MQTTBOX dargestellt, der die Kommunikation der einzelnen Komponenten (Website, Pythonanwendung und Wemos-Modul) aufzeigt.

Abb. 1: MQTT Connection

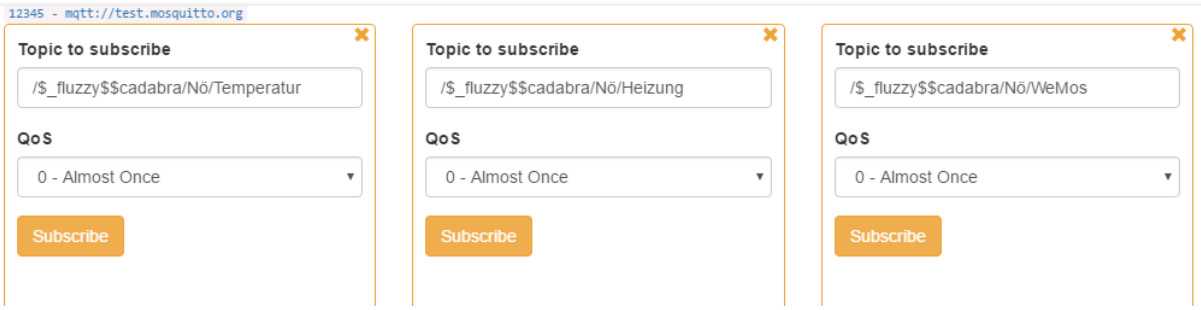

Abb. 2: MQTT Subscriber Übersicht

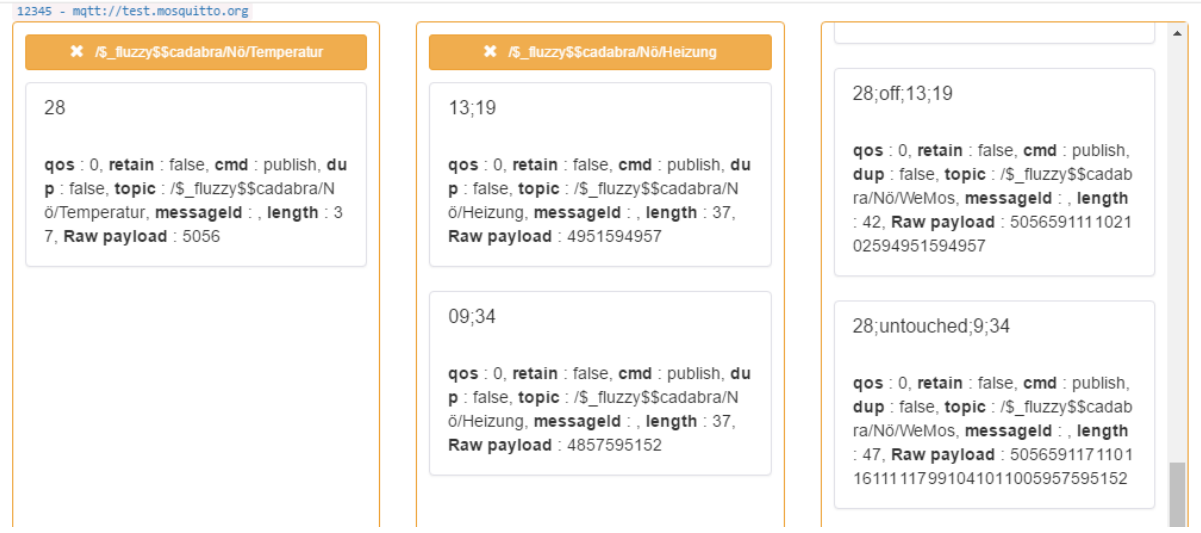

Abb. 3: MQTT Abfrage Beispiel