IOTS Web Client / Server

Jonas Arnold, Simon Frei, Fabian Guggenbühl, Dario Troxler

November 8, 2021

Abstract

Die Temperatur- / Lichtsensorwerte werden vom MKR board ausgelesen und auf Dweet gepostet. Der Webserver bietet die Möglichkeit, eine LED auf dem Arduino ein-/auszuschalten, die Häufigkeit der Sensor Updates einzustellen und zeigt einen den Wert eines Analog Pins des Arduinos an. Mit einem HTTP 301 "Moved Permanently" redirect wird die URI immer gelöscht nach einem Request.

1 Implementierung Web Client / Web Server Arduino

1.1 Impelemtierung / Gelerntes

Die Implementierung des Programms verlief gut. Alle Anforderungen konnten erfüllt werden und erfolgreich getestet werden.

- Periodisch Sensordaten an Dweet.io übermitteln
- Gleichzeitig laufender Webserver um die Periodizität zu ändern

Mit einfachen Präprozessoranweisungen wurde eine Unterscheidung zwischen Schul- und Heimnetzwerk eingebaut.

Die Eingabe im Web UI für das Setzen der Periodizität wurde mit einer HTML-form umgesetzt (input type=number):

```
<form action="/get">Sensor update cycle periodicity (in ms): <input type=
    "number" name="periodMs" min="2000" max="60000" value="10000"><input
    type="submit" value="Set">
```

Zwar kann damit Min/Max begrenzt werden, jedoch sollte im Arduino code trotzdem erneut der Wert geprüft werden. Im Webbrowser kann der Quellcode angepasst werden und somit kann auch das Min/Max angepasst werden. So kann also trotzdem ein falscher Wert an den Arduino übermittelt werden.

Für die Erkennung und das Auslesen des Get requests (HTML form) wurden einige Strinmanipulationen verwendet:

```
String getRequestKey = "/get?periodMs=";
int indexOfGetRequest = currentLine.indexOf(getRequestKey);
// get request has been found in the uri
if(indexOfGetRequest > 0) {
  // extract value from URI
  long requestedPeriodicity = currentLine.substring(indexOfGetRequest +
      getRequestKey.length()).toInt();
  Serial.print("New periodicity requested: ");
  Serial.println(requestedPeriodicity);
  // check range of allowed values
  if(requestedPeriodicity >= 2000 && requestedPeriodicity < 60000)
  {
    postingInterval = requestedPeriodicity;
    Serial.print("New periodicity set.");
```

```
  }
  getRequestRecognized = true;
}
```

Arduino bietet mit dem Objekt-Typ *String* eine gute Lösung um Stringmanipulationen vorzunehmen. Mit den wenigen verfügbaren Funktionen können einige Manipulationen einfacher als mit ANSI C vorgenommen werden.

Mit dem implementierten HTML-form wird der Get request einfach hinten an die URI angefügt (z.B. */get?PeriodMs=5000*). Das ist einfach auszulesen im Arduino code, jedoch ist die URI im Browser dann immer mit solchen zusätzlichen Argumenten und Parametern "verschmutzt". Deshalb wurde folgendes implementiert, um den Client nach einen Request automatisch auf die Hauptseite zu "forwarden" (mittels HTTP 301 Moved Permanently redirect):

```
// reload client page when a get request was recognized => clear URI
if(getRequestRecognized)
{
    client.println("HTTP/1.1 301 Moved Permanently");
    client.println("Location: /");
    client.println("Content-type:text/html");
    client.println();
}
```

1.2 Effizienz steigern

Die Energieeffizienz des Arduinos könnte noch gesteigert werden, indem der Sleep Modus oder Deepsleep Modus des Arduinos konfiguiert würde. Dann würde der Arduino regelmässig aufwachen, sich mit dem WiFi verbinden und die Daten übermitteln. Höchstwahrscheinlich würde jedoch das periodische Verbinden mit dem WiFi mehr Energie verbrauchen, als einfach den Arduino dauerhaft verbunden zu lassen.

Ansonsten könnte man eine effizientere Übertragungstechnik (z.B. Zigbee) verwendet werden. Dann müsste die Verbindung nicht dauerhaft bestehen.

2 Anbindung MQTT

2.1 Arduino

Bei der Implementierung traten Probleme, als wir das erste Mal die MqttClient Library benutzen wollten. Beim Verbinden zum Broker kam die Fehlermeldung "Unrecognized Protocol Version" (Fehlercode 1).