Steuerung eines Arduino Uno Mikrokontrollers via WebSocket

Samuel Hess

11. Mai 2020

Inhaltsverzeichnis

1 Abstract

In der vorliegenden Arbeit wurde untersucht, wie ein Arduino Uno über eine WebSocket-Verbindung gesteuert werden kann, während dieser im Sekundentakt Statusmeldungen versendet.

Der verwendete Funduino Uno R3[13] verfügt über keine WLAN-Schnittstelle. Daher erfolgt die WebSocket-Kommunikation extern auf einem Gateway. Als Gateway wird ein Windows-PC verwendet. Arduino und Gateway sind per USB verbunden und kommunizieren über eine virtuelle serielle Schnittstelle.

Das Ergebnis zeigt, dass eine Steuerung eines Arduinos per WebSocket problemlos möglich ist. Die eingehenden Nachrichten müssen auf dem Arduino geparst werden. Dieses Parding ist einfacher, wenn das verwendete Austauschformt schlank gehalten wird. Deshalb wurde anstelle von JSON das URL Format verwendet.

1.1 Einleitung

In diesem Kapitel wird die Motivation erläutert und genaue Fragesellung definiert. Dann folgt eine kleine Übersichtsarbeit mit dazugehöriger Literaturrecherche.

2 Fragestellung

Welche Möglichkeiten gibt es, einen Arduino Uno via Websocket zu steuern?

Während einer explorativen Online-Suche wurden einzelne Lösungen gefunden. Eine systematische Zusammenstellung der Möglichkeiten fehlt jedoch.

2.1 Motivation

Die Motivation für die vorliegende Arbeit ist die Beantwortung der nachfolgenden Fragestellung. Weiter soll der Artikel interessierten Lesern als Einstiegslektüre diesen.

2.2 Literatur-Review

Zum Thema existiert diverse Fachliteratur unter anderem von Erik Bartmann[8][7][6].

2.2.1 Arduino mit integriertem WLAN

Der Arduino Uno hat keine eingebaute WLAN Schnittstelle. Es gibt jedoch andere Arduino Modelle mit integriertem WLAN, wie z.B. der Arduino MKR1000.

2.2.2 WLAN Erweiterung

Mehrere Autoren berichten[1][22], wie der Arduino mit dem dem WLAN Modul ESP8266 erweitert werden kann.

2.2.3 Serial Gateway

Eine weitere Möglichkeit, ist behelfsweise einen PC als Serial Gateway einzusetzen. Mangels kurzfristig verfügbarer Hardware wollen wir diese Option verfolgen.

3 Experimenteller Teil

3.1 Informationsquellen

Als Informationsquellen sind die Datenblätter zur jweiligen Hardware sowie die Manuals zu den einsesetzten Softwarekomponenten zu nennen.

3.2 Prinzipskizze

3.3 Hardware

Verwendet wurde das Lernset Nr. 8 von Funduino[13]. Darin enthalten ist ein Funduino Uno. Weiter benötigen wir den Temparatursensor TMP36 und den Fotowiderstand.

3.3.1 Anschluss der Sensoren

3.4 Software

3.4.1 Entwicklungsumgebung

Zur Entwicklung wurde folgende Software eingesetzt.

- Visual Studio Code[20] mit der Erweiterung C/C++ IntelliSense[9]
- Arduino CLI[3]
- Git for Windows[12] und TortoiseGit[19]

Nicht verwendet wurde die Arduino IDE. Windows verwendet den Standardtreiber *usbser.sys* für den virtuellen COM Port.

3.4.2 Node Libraries

Weiter wurde folgende NPM Packages eingesetzt:

- WebSockets[15]
- Express[11]
- Chart.js[16]
- SerialPort[14]

3.4.3 Arduino Libraries

Weiter wurde folgende Arduino Libraries eingesetzt:

- Arduino Library (Arduino.h)[17][2][4]
- AVR Libc[5]

3.4.4 Arduino Sketch

Zunächst müssen wir klären, in welcher Programmiersprache die Arduino Sketches geschrieben werden. Nachdem man sich die Build-Umgebung genauer unter die Lupe genommen hat, wird klar, dass keine eigene Arduino-Sprache existert[10]. Im Hintergrund wird aus dem Sketch eine C++ Datei erstellt und mit *avr-g++* kompiliert.

Die Problematik der Heap-Fragmentierung wird von mehreren Autoren aufgeworfen und diskutiert[23][21]. Matt ist der Meinung, dass man deshalb auf die String Klasse in der Arduino Library gänzlich verzichten soll[18]. In der Konsequenz müsste man die Stringfunktion aus der Standard C Library[5] verwerden und in C programmieren. Ich

sehe dies nicht ganz so eng und setze die Arduino String Klasse trotzdem, jedoch mit Zurückhaltung ein. Ich befolge Matt's Rat, die Variablen by Reference zu übergeben[18].

Der Quellcode befindet sich im Anhang.

3.4.5 Serial Gateway

Der Quellcode befindet sich im Anhang.

3.4.6 WebSocket Server

Der Quellcode befindet sich im Anhang.

3.4.7 Web GUI

Der Quellcode befindet sich im Anhang.

4 Resultate

Es hat sich gezeigt, dass ein Seriell-zu-Websocket-Gatway unter Node.js einfach zu implementieren ist. Über diesen Umweg kann der Arduino Uno ans Internet angebunden werden.

5 Diskussion

6 Zusammenfassung

Statt des Arduino Uno könnte ein Arduino MKR1000 verwendet werden. Dieser könnte kann auch an die Arduino Clound angebunden werden. Ein weitere Option ist die Beschaffung einer WLAN Erweiterung wie das Modul ESP8266.

7 Danksagung

Ich danke den Lernenden der Klasse BINF2017A für die Zusammenarbeit.

8 Interessenskonflikte

Das Projekt wurde im Rahmen des Beruffachschulunterrichts durchgeführt und erhielt keine externde Finanzierung. Demnach bestehen keien Interessenkonflikte.

Literatur

[1] Igor Khanenko Andrew Shvayka Igor Kulikov. *Temperature Dashboard Using Arduino UNO, ESP8266 And MQTT*. URL: https://www.hackster.io/thingsboard/temperature-dashboard-using-arduino-uno-esp8266-and-mqtt-5e26eb.

[2] *Arduino Befehlsübersicht*. URL: https://www.arduinoforum.de/code-referenz.

[3] *Arduino CLI*. Arduino. URL: https://github.com/arduino/arduino-cli.

[4] *Arduino Programming Cheat Sheet*. URL: https://github.com/liffiton/Arduino-Cheat-Sheet.

[5] *AVR Libc*. URL: https://www.nongnu.org/avr-libc/.

[6] Erik Bartmann. *Das ESP32-Praxisbuch: Programmieren mit der Arduino-IDE*. Elektor Verlag, 2018. ISBN: 978-3-89576-333-5.

[7] Erik Bartmann. *Das ESP8266-Praxisbuch: Mit NodeMCU und ESPlorer*. Elektor Verlag, 2016. ISBN: 978-3-89576-321-2.

[8] Erik Bartmann. *Mit Arduino die elektronische Welt entdecken*. 3. Aufl. Bombini-Verlag, 2017. ISBN: 978-3-946496-00-7.

[9] *C/C++ IntelliSense, debugging, and code browsing*. Microsoft. URL: https://marketplace.visualstudio.com/items?itemName=ms-vscode.cpptools.

[10] *C++ vs. The Arduino Language?* URL: https://arduino.stackexchange.com/questions/816/c-vs-the-arduino-language.

[11] *Express Schnelles, offenes, unkompliziertes Web-Framework für Node.js*. URL: https://expressjs.com.

[12] *Git for Windows*. URL: https://gitforwindows.org.

[13] *Lernset Nr.8 mit UNO Controller - Kit für Arduino*. Funduino. URL: https://www.funduinoshop.com/epages/78096195.sf/de_DE/?ObjectPath=/Shops/78096195/Products/01-U8.

[14] *Node SerialPort*. URL: https://serialport.io.

[15] *Simple to use, blazing fast and thoroughly tested WebSocket client and server for Node.js*. URL: https://github.com/websockets/ws.

[16] *Simple yet flexible JavaScript charting for designers & developers*. URL: https://www.chartjs.org.

[17] *Sprach-Referenz*. URL: https://www.arduino.cc/reference/de/.

[18] *The Evils of Arduino Strings*. 4. Feb. 2016. URL: https://majenko.co.uk/blog/evils-arduino-strings.

[19] *TortoiseGit*. URL: https://tortoisegit.org.

[20] *Visual Studio Code*. Microsoft. URL: https://code.visualstudio.com/.

[21] Colin Walls. *Dynamic Memory Allocation and Fragmentation in C and C++*. 6. Nov. 2018. URL: https://www.design-reuse.com/articles/25090/dynamic-memory-allocation-fragmentation-c.html.

[22] *WebSocket communication with an ESP8266 or Arduino in Python. Test with the ws4py library on Raspberry Pi*. URL: https://diyprojects.io/websocket-communication-esp8266-arduino-python-test-ws4py-library-raspberry-pi/#.Xq6bKagzaUk.

[23] *What is Heap Fragmentation?* 6. Nov. 2018. URL: https://cpp4arduino.com/2018/11/06/what-is-heap-fragmentation.html.