Szene-Trends nachgefragt – in dieser Ausgabe bei Niko Köbler

Blumengießen mit JavaScript

JavaSPEKTRUM enthüllt, warum aus einem szenebekannten Softwareentwickler ein Smart-Home-Bauer wird und was es mit Espruino auf sich hat.

Niko Köbler *ist freiberuflicher Softwarearchitekt, Developer & Trainer für (Enterprise-)Lösungen mit Java & JavaScript, Integrationen und Webdevelopment. Er ist Co-Lead der JUG Darmstadt und regelmäßig als Sprecher auf internationalen Fachkonferenzen anzutreffen, so auch auf der OOP 2015. Sein neuestes Steckenpferd: JavaScript für Dinge, realisiert mit Espruino. Gordon Williams (Twitter: @Espruino) startete 2013 zum Crowdfunding eine Kickstarter-Kampagne, in der er ein spezielles Espruino-Board und die Freigabe des Quelltextes anbot. Mit überwältigendem Erfolg! Auch das neueste, minimierte Espruino Pico ist deutlich günstiger als vergleichbare Boards für JavaScript-Programmierung, z. B. Tessel.*

Bildnachweis: Foto Annegret Handel-Kempf

Niko Köbler im Interview zu seinen ganz persönlichen Erfahrungen mit dem JavaScript für Dinge:

„Alles nur JavaScript und keine Raketentechnologie"

▼ *JavaSPEKTRUM: Niko, Du bist Softwarearchitekt, Developer und Java-Enthusiast. JavaScript verhält sich total anders als Java, trotzdem empfiehlst Du es Java-(Web-)Entwicklern. Warum?*

Niko Köbler: JavaScript ist besser und flexibler einsetzbar, seit die Sprache mit Node.js und der weitergehenden ECMA-Script-Standardisierung auf der Serverseite implementiert wurde. Durch die dynamische Typisierung sind mit JavaScript andere Einsatzszenarien denk- und machbar, als es zum Beispiel mit Java möglich ist.

▼ *Die Wirtschaft wird digitalisiert, kein Unternehmen kommt mehr ohne Internet of Things (IoT), kein Neubau mehr ohne Smart-Home-Anwendungen aus. Liegt hier die Zukunft von Java und JavaScript?*

JavaScript gilt als „Sprache des Web" und hat eine echte „Write once, run everywhere"-Philosophie. Und JavaScript läuft heute bereits auf vielen unterschiedlichen Plattformen: Das fängt bei den Internet-Browsern an, geht über Dinge des alltäglichen Gebrauchs wie Smartphones, Smart-TVs, Tablets usw. bis hin zum Server. Warum sollten nicht auch Mikrocontroller und USB-Devices mit JavaScript betrieben werden?

▼ *Auf dem Espruino-Mikrocontroller-Board läuft standardmäßig und nativ ein interaktiver JavaScript-Interpreter. Auch auf der Entwicklungsplattform Tessel 2, die in multiplizierbare Produkte eingebettet wird, ist man mit JavaScript dabei. Was hat Dich speziell zu Espruino geführt?*

Als ich mit IoT noch nichts am Hut hatte, saß ich bei einer Konferenz in einem IoT-Einführungs- und Überblicks-Talk. Dort wurden verschiedene Boards und Mikrocontroller vorgestellt, sowie Möglichkeiten, diese zu programmieren. Für Java war irgendwie nicht viel dabei. Lediglich der altbekannte Raspberry Pi, den ich aber nicht wirklich zur IoT-Landschaft zähle. Der Pi ist „nur" ein kleiner, aber vollständiger Linux-Rechner. Für JavaScript wurde lediglich der Tessel vorgestellt, der aber deutlich zu teuer war, um nebenbei damit rumzuspielen. Also startete ich eine Google-Suche, ob es denn keine günstigen Mikrocontroller-Boards gibt, die mit JavaScript programmiert werden können. Dadurch wurde ich auf Espruino aufmerksam, habe das Board bestellt und begonnen, damit rumzuexperimentieren.

▼ *Was kennzeichnet Espruino JavaScript?*

Hier muss man zwischen Hardware und Software unterscheiden. Das klassische Espruino-Board ist nur ungefähr halb so groß wie eine Visitenkarte, enthält neben einem 32-Bit-Chip aus der STM32F1-Familie mit 72 MHz Taktung noch 256 KB Flash Memory für die Firmware und 48 KB RAM für die Anwendungen. Das klingt nicht nach viel, reicht in den meisten Fällen jedoch vollkommen aus. Ressourcen-hungrige UI-Anwendungen werden eher wenig auf dem Espruino entwickelt – obwohl auch das möglich ist. Dazu kommen noch 44 verschiedene GPIOs, ein Anschluss für eine externe Stromversorgung, ein USB-Anschluss, ein SD-Kartenslot, Pads für einen Bluetooth-Chip und eine Prototyping-Area. Viele Möglichkeiten für ein kleines Board. Ende letzten Jahres wurde per erneuter Kickstarter-Kampagne ein neues, nur USB-Stick großes Espruino-Board ins Leben gerufen: der Espruino PICO, mit einem ähnlichen Chip, etwas mehr RAM, aber weniger Anschlüssen, was aber lediglich der sehr kleinen Bauform geschuldet ist. Die erste Produktion der Picos ist gerade ausgeliefert worden, die zweite Charge wird aufgrund der großen Nachfrage etwas auf sich warten lassen.

Für die Firmware wurde keine bestehende JavaScript-Engine verwendet und angepasst, sondern es wurde ein komplett neuer Interpreter geschrieben, der nativ auf dem Controller läuft, ohne eine weitere Betriebssystemschicht dazwischen. Das erhöht die Ausführungsgeschwindigkeit enorm. Alle wichtigen JavaScript-Funktionen wurden implementiert und einige auf die Board-spezifischen Anwendungsfälle hin optimiert. Der Interpreter stellt ähnlich wie Node.js oder Vert.x im Java-Umfeld eine Event-Loop zur Verfügung, mit der asynchron und nicht-blockierend programmiert werden kann. Es ist sogar möglich, Module aus dem Node-Package-Manager (NPM) zu verwenden, sofern diese keine betriebssystem- oder andere sprachspezifischen Abhängigkeiten haben und in den Speicher passen. Die Nutzung von JavaScript als Programmiersprache auf einem Board hat weiterhin den Vorteil, dass ich Programmcode zur Laufzeit ändern kann und der Deployment-Zyklus sehr kurz wird, da ich nichts mehr kompilieren und installieren muss. Die komplette Doku zu Espruino ist auf der Website [`www.espruino.com` – Anm. der Redaktion] verfügbar und sehr umfangreich.

▼ *Wie geht ein JavaScript-Anfänger, der sich dem Internet of Things mit Espruino annähern will, konkret vor? Was muss er hard- und softwaretechnisch tun?*

Anfängern empfehle ich, sich das „normale" Espruino-Board, also nicht den Pico, zu kaufen, denn hier sind für die ersten Anfänge bereits drei LEDs und zwei Buttons enthalten, die direkt ansprechbar sind. Das reicht für komplexere Anwendungsfälle natürlich nicht aus, ist für den schnellen Erfolg jedoch geradezu ideal. Mit einem USB-Kabel wird das Board mit dem Rechner verbunden. Über eine Terminalsession kann ich nun eine serielle Verbindung zum Espruino aufbauen und direkt mit dem Programmieren beginnen. Wer es komfortabler haben will, der findet im Google Chrome-Store eine Web-IDE, mit der sehr einfach auch umfangreicherer Code geschrieben werden kann – sogar mit einem grafischen Editor. Auch aktuelle Firmware-Updates können mit Hilfe der Web-IDE einfach aufgespielt werden. Mehr ist für den Anfang nicht notwendig. Espruino läuft quasi „out-of-the-box" und ist in weniger als fünf Minuten einsatzbereit.

„Es kommen fast täglich neue Demos und Anwendungsszenarien hinzu. Die Szene ist sehr aktiv, man muss einfach nur Augen und Ohren offen halten."

▼ *Kannst Du ein Beispiel fürs Schreiben des Codes geben, etwa für ein Licht im Eingangsbereich, das mithilfe von Espruino im Dunkeln angehen soll?*

Ok, gerne. Der Einfachheit halber ersetzen wir das Licht im Eingangsbereich durch eine LED auf dem Board, die Logik bleibt aber die gleiche. Zusätzlich brauchen wir noch einen Fotowiderstand, um die Helligkeit zu messen. Diesen Fotowiderstand verbinden wir mit dem Espruino, der dort über den Pin „A5" ansprechbar ist. Mittels der Funktion

```
analogRead(A5)
```

bekommen wir einen Wert zwischen 0 (0 Volt, 100 Prozent hell) und 1 (3,3 V, 100 Prozent dunkel) zurückgeliefert. Wir können entweder mit 0 und 1 arbeiten, oder ermitteln mittels der Funktion die jeweiligen Werte für die gewünschten Hell- und Dunkel-Szenarien. Diese Werte speichern wir dann in den Variablen **`light`** und **`dark`** für den weiteren Gebrauch, zusätzlich ermitteln wir noch den Mittelwert, der als Schaltgrenze dienen soll

```
var mid = 0.5*light + 0.5*dark;
```

Jetzt können wir die entsprechende Funktion zum Schalten der LED schreiben

```
function step() {
  var l = analogRead(A5);
  digitalWrite(LED1, l > mid);
}
```

Diese Funktion müssen wir periodisch aufrufen, damit ständig der Helligkeitswert überwacht und somit die LED geschaltet wird

```
setInterval(step, 10);
```

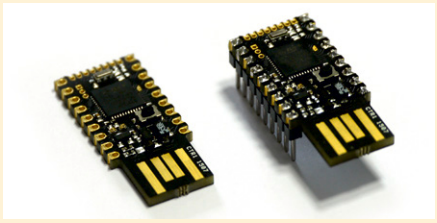
So wird alle 10 Millisekunden, also 100 Mal in der Minute, überwacht, wie hell oder dunkel es am Fotowiderstand ist, und die LED entsprechend ein- oder ausgeschaltet.

Das war eigentlich schon alles. Man muss noch etwas mit den Schwellenwerten herumexperimentieren, bis man den gewünschten Wert findet, wann der Espruino die LED schalten soll. Auch sollte man zusätzlich berücksichtigen, dass es heller wird, wenn die LED oder letztendlich die Lampe angeht, und sich dann der Wert am Fotowiderstand wieder ändert, was ein ständiges An- und Ausschalten der Lampe zur Folge hätte. Hierfür ist noch etwas zusätzliche Logik erforderlich, die aber Espruino-unabhängig auch mit jedem anderen Controller notwendig wäre. Also eher eine fachliche als eine technische Anforderung.

Der gezeigte Code ist eigentlich die Basis für alle Überwachungs- und Schaltprozesse, egal ob es sich um Fotowiderstände oder Sensoren anderer Art, beispielsweise zur Feuchtigkeit, Akustik usw., dreht.

▼ *Wie weiß ich bei einzelnen Programmierschritten, ob sie erfolgreich sind?*

Da in JavaScript jede Funktion einen Rückgabewert hat, auch die Funktionen, in denen kein expliziter Rückgabewert angegeben ist – sie wird dann „undefined" als Wert zurückgeben –, bekomme ich bei der Programmierung jederzeit eine Rückmeldung über die erfolgreiche Ausführung. Tritt ein Fehler auf, sehe ich auf der Konsole natürlich auch die Fehlermeldung.



Pins des Espruino Pico. Fotonachweis: Espruino

Ich kann mich aber auch live mit einem Espruino verbinden, der gerade Code ausführt. So sehe ich, welche Werte die Variablen gerade haben. Ist halt alles nur JavaScript, und somit keine Raketentechnologie.

▼ *Espruino-„Vater" Gordon Williams ist ein begeisterter Energiesparer. Wie reduziert er den Stromverbrauch?*

Gordon Williams hat es geschafft, die Interpreter-Software auf dem Espruino so „intelligent" zu gestalten, dass diese in Abhängigkeit des verwendeten Programmcodes selbstständig entscheidet, wann, wie lange und wie tief sich der Controller schlafen legen kann und damit keinen Strom verbraucht. Das geht sogar so weit, dass ein angeschlossener WiFi-Controller keinen Strom verbraucht, wenn er nur auf Signale wartet. Erst wenn ein neuer Netzwerk-Impuls kommt, auf den der Espruino reagieren soll, wird wieder Strom verbraucht.

Wie stromsparend der Espruino ist, zeigt ein Versuchsaufbau, bei dem eine LED mit einer 700mAh-Batterie möglichst lange blinken soll. Ein Raspberry Pi schaffte gerade mal ein bis zwei Stunden, der Arduino hielt immerhin zwei Tage durch. Der Espruino aber läuft ungefähr sechs Monate in diesem Szenario.

▼ *Wie groß ist die Bandbreite der Szenarien, wo via Espruino mit JavaScript programmiert werden könnte?*

Die Szenarien reichen von allem, was irgendwie mit Sensor-Daten, also Licht, Temperatur, Wetter, Akustik usw., und dem MQTT-Protokoll zu tun hat, bis hin zu neuen Optionen: Beispielsweise ist es möglich, den Espruino über den Kopfhörerausgang eines Smartphones oder Tablets zu „programmieren", der dann in Abhängigkeit der akustischen Signale angeschlossene Geräte steuert. Neu ist jetzt auch, dass der Espruino als USB-Device agieren kann. Damit wird es möglich, eigene Controller, wie Joystick- oder Maus-ähnliche Geräte, zu bauen.

Es kommen fast täglich neue Demos und Anwendungsszenarien hinzu. Die Szene ist sehr aktiv, man muss einfach nur Augen und Ohren offen halten.

▼ *Du baust gerade ein Haus. Bei welchen Automatisierungen soll der Espruino zum Einsatz kommen?*

Ich fange sicherlich bei den weit verbreiteten Szenarien, wie der Rollladen- und Lichtsteuerung, an und schaue dann mal, was mir noch so einfällt. Vielleicht kann ich meine Frau auch dazu überreden, unsere Pflanzen von Espruino bewässern zu lassen. Sie traut der ganzen Technik aber noch nicht so und befürchtet eine Überschwemmung im Wohnzimmer.

▼ *Werden mithilfe von JavaScript-Interpretern klassische Heimwerker künftig zu Smart-Home-Bauern?*

Ich möchte mir nicht vorstellen, wie ein Sanitär-Installateur plötzlich die WC-Spülung per JavaScript programmiert, ohne dass er Programmiererfahrung hat.

▼ *Was denkst Du - Haben im IoT hardwarenahe oder eher hardwareferne Programmiersprachen die größten Chancen, häufig eingesetzt zu werden?*

Das kommt ganz auf die am Markt verfügbaren Boards an. Das IoT wird populärer, wenn es von möglichst vielen Menschen mitgestaltet werden kann. Sind nur Boards und Mikrocontroller verfügbar, die mit hardwarenahen Sprachen programmiert werden können, so würde der Siegeszug des IoT länger dauern. Spielt es letztendlich keine Rolle, mit welcher Sprache ich das Licht in meinem Haus an- und ausgehen lassen kann, wird eine schnellere und weiter verbreitete Adaption stattfinden.

Interview: Annegret Handel-Kempf