Zephyr-Projekt

Echtzeit-OS für das Internet der Dinge

Projektarbeit

Die Linux Foundation hat mit dem Projekt Zephyr mit der Entwicklung eines Echtzeit-Betriebssystems für das Internet der Dinge (IoT) begonnen. Zephyr ist ein Open-Source-Betriebssystem mit dem Ziel ein solides OS für IoT Geräte mit geringen Ressourcen bereitzustellen. Es nutzt eine echtzeitfähige Kombination aus Nano- und Microkernel. Im Gegensatz zu einem Linux Kernel benötigt Zephyr nur zwischen 8 und 512 KByte an Arbeitsspeicher. Aktuell werden folgenden Plattformen unterstützt: x86, ARM und ARC EM4

Studiengang:	Elektro- und Kommunikationstechnik
Autoren:	Aaron Schmocker, David Wyss
Betreuer:	Martin Aebersold
Auftraggeber:	Martin Aebersold
Experten:	Martin Aebersold
Datum:	29.09.2016

Versionen

Version	Datum	Status	Bemerkungen
0.1	29.09.2016	Entwurf	Titelblatt erstellt und Template für Linux angepasst

Management Summary

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus scelerisque, leo sed iaculis ornare, mi leo semper urna, ac elementum libero est at risus. Donec eget aliquam urna. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc fermentum nunc sollicitudin leo porttitor volutpat. Duis ac enim lectus, quis malesuada lectus. Aenean vestibulum suscipit justo, in suscipit augue venenatis a. Donec interdum nibh ligula. Aliquam vitae dui a odio cursus interdum quis vitae mi. Phasellus ornare tortor fringilla velit accumsan quis tincidunt magna eleifend. Praesent nisl nibh, cursus in mattis ac, ultrices ac nulla. Nulla ante urna, aliquet eu tempus ut, feugiat id nisl. Nunc sit amet mauris vitae turpis scelerisque mattis et sed metus. Aliquam interdum congue odio, sed semper elit ullamcorper vitae. Morbi orci elit, feugiat vel hendrerit nec, sollicitudin non massa. Quisque lacus metus, vulputate id ullamcorper id, consequat eget orci

Cite test: [1] [3] [**?**]

Inhaltsverzeichnis

1. Einleitung

1.1. TEST

1.2. TEST2

2. ZephyrOverview

2.1. Übersicht über Zephyr

Zephyr ist laut Beschreibung der Linux-Foundation [2] ein Open-Source-Echtzeitbetriebssystem, speziell optimiert für Anwendungen im Internet der Dinge. Die Architektur basiert auf einer echtzeitfähigen Kombination von Nano- und Mikrokernel. Es wird aktuell von der Linux-Foundation in Zusammenarbeit mit den Firmen Intel, NXP und Synopsys in der Form eines Collaborative-Projects entwickelt. Dadurch soll versucht werden die bei der Linux- und Open-Source-Entwicklung verwendeten Arbeitsweisen und Ideen auch im Bereich der Industrie einzubringen. Ziel ist es ein robustes und sicheres Betriebssystem für das Internet der Dinge zu schaffen. Zephyr ist vollständig Open-Source steht laut Information des Newsportals Heise.de [3] unter der Apache Lizenz Version 2.0. Dieses Lizenzierungsmodell kommt Firmen und Unternehmen entgegen, welche den Einsatz von Open-Source-Software generell scheuen da diese oft unter der GNU General Public License (GPL) stehen. [2] Wird in Produkten Software verwendet welche unter der GPL lizensiert ist, zwingt dieses Lizenzmodell die Firmen dazu ihre Produkte ebenfalls unter GPL zu veröffentlichen. Dies beinhaltet auch sämtliche Änderungen welche vorgenommen wurden. Bei der Apache-Lizenz ist dies gemäss Definition [1] nicht zwingend.

Momentan unterstützt der Zephyr-Kernel gemäss Angaben auf der Projektseite [4] Prozessoren der Architekturen ARC, ARM-v7 aber auch x86. Dadurch ist das System auf populären Plattformen wie dem Arduino 101 Board, dem Arduino Due Board, dem NXP Freedom DK lauffähig und Intel Galileo Gen 2. Zur Kommunikation stehen unter anderem Protokolle Ipv4, Ipv6, Bluetooth 4.0, LoWPAN zur Verfügung.

2.2. Ziele

Zephyr ist für den Einsatz auf Geräten mit geringem Speicherplatz und feststehender Hardwarekonfiguration gedacht. Darunter fallen unter anderem Steuerungen für Heizungs- und Beleuchtungssysteme aber auch Geräte aus allen Bereichen des täglichen Lebens mit Internet-Anbindung.

Das ZephyrProjekt verfolgt laut Beschreibung der Linux-Foundation [2] folgende Ziele:

- Kleiner footprint - lauffähig mit minimal 10kB
- CPU unabhängige Architektur
- Modular und Skalierbar
- Hoche Sicherheitsstandards
- Unterstützt von Grund auf viele unterschiedliche Boards und Kommunikationsprotokolle
- Mächtige Entwicklungswerkzeuge
- OpenSource Kernel mit Apache v2.0 Lizenz

2.3. Aufbau

Das Zephyr OS setzt auf eine Kombination von Nano- und Mikrokernel. Dadurch soll Zephyr bereits mit nur 10 Kbyte an Speicherplatz lauffähig sein. Das macht Zephyr besonders für Anwendungen auf kleinen Mikrokontrollern attraktiv. Bei einem herkömmlichen Linux-Kernel wäre dies nicht denkbar. Gängige Adaptionen für Smartphone SoCs benötigen laut [1] in der kleinsten Konfiguration noch bis zu 200KB RAM und rund 1MB Flash.

Der Nanokernel bietet Echtzeit-Fähigkeiten. Die Zeit die der Nanokernel für die Abarbeitung einer Aufgabe benötigt ist also deterministisch. Dies unabhängig davon wie stark das System gerade ausgelastet ist. Für alle Aufgaben welche keine Anforderungen an Echtzeit-Verarbeitung stellen verfügt das System über einen Microkernel. [1] http://linuxdevices.linuxgizmos.com/my-linux-is-smaller-than-your-linux-a/

2.4. Aufsetzen der SDK unter Ubuntu

2.5. Vergleich mit anderen RTOS

2.6. Sicherheitsaspekte

2.7. Portierungsaufwand

3. nRF52DK

3.1. Installation der GNU ARM Embedded Toolchain

Um die Software auf dem Hostsystem für das nRF52DK kompilieren zu können benötigen wir eine sogenannte crosscompiler-toolchain. Für ARM basierte Controller ist hier die erste Wahl die gnu-arm-none-eabi Toolchain. Sie lässt sich von folgender Seite beziehen:

`https://launchpad.net/gcc-arm-embedded/+download`

Es gilt darauf zu achten die neueste Version zu verwenden und sich zu notieren welche Version heruntergeladen wurde, da diese später im Makefile angegeben werden muss. Die Software sollte im Verzeichnis /opt installiert werden. Die Binaries sollten mit den nötigen Rechten ausgestattet werden. Weiter empfiehlt es sich symbolische Links zu generieren um die Programme systemweit sichtbar zu machen.

Die Befehle dazu lauten:

```
tar -xf gcc-arm-none-eabi-{VERSION}-linux.tar.bz2 /opt/gcc-arm-none-eabi-{VERSION}
ln -s /opt/gcc-arm-none-eabi-{VERSION}/bin/* /usr/local/bin/
```

Unter Arch Linux lässt sich die Software auch aus den offiziellen Paketquellen installieren, die Befehle dazu lauten:

```
pacman -S community/arm-none-eabi-binutils
pacman -S community/arm-none-eabi-newlib
pacman -S community/arm-none-eabi-gcc
pacman -S community/arm-none-eabi-gdb
```

3.2. Installation der nordic SDK

Nordic verfügt über ein Software Repository von welchem alle benötigte Software bezogen werden kann. Das Repo befindet sich unter folgendem Link:

`https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF52-DK#Downloads`

Um mit dem nRF52DK arbeiten zu können benötigen wir folgende Pakete:

1. nRF5 SDK Zip File
2. nRF5x-Command-Line-Tools-Linux64

Im ersten Paket befinden sich alle Files welche zur Entwicklung von Software auf dem nRF52dk notwendig sind. Darunter Makefiles, Linkerskripte und auch Beispielcode. Im zweiten Paket befinden sich die Command Line Tools welche es uns ermöglichen über die Kommandozeile mit dem nRF52DK zu kommunizieren. Das wichtigste Tool ist nrfjprog welches Hex Files auf das Board hochladen kann.

Die SDK sowie die Command Line Tools sollten auf dem System unter dem Ordner /opt installiert werden und mit den nötigen Berechtigungen versehen werden. Ebenfalls sollten die Binaries in einen Ordner gelinkt werden welcher im Systempfad angegeben ist.

Die Befehle dazu lauten:

```
tar -xf nRF5x-Command-Line-Tools-Linux64.tar /opt/
ln -s /opt/nrfjprog/nrfjprog /usr/bin/nrfjprog
ln -s /opt/mergehex/mergehex /usr/bin/mergehex
```

Unter Arch Linux lassen sich die Command Line Tools auch über das Arch User Repository beziehen. Dazu kann man den Pacman Wrapper "yaourt"verwenden:

```
yaourt -S aur/nrf5x-command-line-tools
```

3.3. Installation des SEGGER JLink Debuggers

Die Software von SEGGER welche fürs Debuggen gebraucht wird, findet man unter folgendem Link:

`https://www.segger.com/downloads/jlink`

Das Paket "J-Link Software and Documentation Pack"verfügt über verschiedene Binaries. Die Wichtigsten davon sind der JLinkCommander und der JLinkGDBServer. Über den Commander lässt sich die CPU des nRF52 vollständig via JTAG oder SWD Schnittstelle kontrollieren. Der JLinkGDBServer stellt auf dem Localhost einen Socket unter Port 2331 zur Verfügung, auf welchen man sich mit GDB verbinden kann.

Die JLink Toolchain sollte auf dem System unter dem Ordner /opt installiert werden und mit den nötigen Berechtigungen versehen werden. Die Binaries sollten mittels symbolischen Links in einen Ordner gelinkt werden welche in der Pfadvariable PATH zu finden ist.

```
mkdir /opt/SEGGER/JLink
tar -xzf JLink_Linux_{VERSION}.tar.gz /opt/SEGGER/JLink
ln -s /opt/SEGGER/JLink/JLinkExe /usr/bin/JLinkExe
ln -s /opt/SEGGER/JLink/JLinkGDBServer /usr/bin/JLinkGDBServer
```

SEGGER bietet auch ein Debianpaket an. Dieses kann auf Systemen welche über den Paketmanager DPKG verfügen mittels `dpkg -i JLink_linux.deb` installiert werden. Dadurch entfällt obiger Installationsaufwand.

Unter ArchLinux lässt sich die Software auch aus dem Arch-User-Repository installieren, der Befehl dazu lautet:

```
yaourt -S aur/jlink-software-and-documentation
```

3.4. Verwendung des JLinkGDBServers

```
JLinkGDBServer -Device nRF52832_xxAA -If SWD -Speed 4000 -Autoconnect 1
```

3.5. Verwendung von nrfjprog

```
nrfjprog --eraseall -f nrf52
nrfjprog --program outdir/nrf52_pca10040/zephyr.hex -f nrf52
nrfjprog --reset -f nrf52
```

3.6. Vorteile

3.7. Technische Beschreibung des Boards

3.8. Bluetooth Low Energy

3.9. Analog to Digital Converter

3.10. I2C Bus

4. DemoApp

4.1. Technische Übersicht

4.2. Hardware Dokumentation

4.3. Software Dokumentation

4.4. Softwaretests

5. Fazit

5.1. Vergleich Ist/Soll

5.2. Wünschenswerte Erweiterungen

Selbständigkeitserklärung

Ich/wir bestätige/n, dass ich/wir die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der im Literaturverzeichnis angegebenen Quellen und Hilfsmittel angefertigt habe/n. Sämtliche Textstellen, die nicht von mir/uns stammen, sind als Zitate gekennzeichnet und mit dem genauen Hinweis auf ihre Herkunft versehen.

Ort, Datum:	[Biel/Burgdorf], 29.09.2016	
Namen Vornamen:	[Test Peter]	[Müster Rösä]
Unterschriften:	.....	.....

Glossar

API Application Programming Interface

CPU Central Processing Unit

BLE Bluetooth Low-Energy

SoM System On Module

SoC System On Chip

GPL GNU General Public License

Literaturverzeichnis

[1] A. S. Foundation, "Apache license 2.0." [Online]. Available: http://www.apache.org/licenses/LICENSE-2.0

[2] L. Foundation, "Linux foundation announces rtos for iot devices."

[3] T. Leemhuis, "Linux-foundation startet projekt für iot-betriebssystem."

[4] ZephyrProject, "Zephryproject documentation." [Online]. Available: https://www.zephyrproject.org/doc/board/board.html

Abbildungsverzeichnis

Tabellenverzeichnis

A. Beliebiger Anhang

B. Weiterer Anhang

B.1. Test 1

Phasellus eget velit massa, sed faucibus nisi. Etiam tincidunt libero viverra lorem bibendum ut rutrum nisi volutpat. Donec non quam vitae lacus egestas suscipit at eu nisi. Maecenas non orci risus, at egestas tellus. Vivamus quis est pretium mauris fermentum consectetur. Cras non dolor vitae nulla molestie facilisis. Aliquam euismod nisl eget risus pretium non suscipit nulla feugiat. Nam in tortor sapien.

B.1.1. Umfeld

Nam lectus nibh, laoreet eu ultrices nec, consequat nec sem. Nulla leo turpis, suscipit in vulputate a, dapibus molestie quam. Vestibulum pretium, purus sed suscipit tempus, turpis purus fermentum diam, id cursus enim mi a tortor. Proin imperdiet varius pellentesque. Nam congue, enim sit amet iaculis venenatis, dui neque ornare purus, laoreet porttitor nunc justo vel velit. Suspendisse potenti. Nulla facilisi.