Embedded Systems Hands-On 1: Entwurf und Realisierung von Hardware/Software-Systemen

TECHNISCHE UNIVERSITÄT DARMSTADT

Jaco Hofmann M.Sc., Dipl.-Inform. Andreas Engel

Sommersemester 2016 (Stand 21. April 2016)

Aufgabe 1: Embedded Linux

Diese Aufgabe beschäftigt sich mit dem Raspberry Pi und Embedded Linux. Die vorgesehene Bearbeitungsdauer beträgt zwei Wochen.

Zusammenfassung

- Raspberry Pi in Betrieb nehmen
- Vorgefertigtes Embedded Linux verwenden
- OpenOCD Kommunikation mit dem Mikrocontroller
- Linux Kernel compilieren mit TFT Treiber
- Devicetree und Kernel Module

Vorgehensweise

Diese Aufgaben verweisen stark auf Internetressourcen, um das üblichen Vorgehen beim Entdecken neuer Plattformen zu verdeutlichen. Fassen Sie die Ergebnisse und Probleme der jeweiligen Teilaufgaben in einem Dokument zusammen und stellen Sie dieses ins GitLab. Committen Sie außerdem ihren erstellten Code.

1.1 Raspberry Pi

Diese Teilaufgabe beschäftigt sich mit der Inbetriebnahme des Raspberry Pi. Dafür benötigen Sie zuerst ein Betriebsystem auf einer Micro SD Karte, das vom Bootloader des Raspberry Pi geladen wird.

Zusammenfassung

- Laden Sie sich eine aktuelle Raspbian Version herunter und starten Sie diese auf dem Raspberry Pi

Auf der Seite des Raspbian Projekts finden Sie eine Linux Distribution auf Basis von Debian für das Raspberry Pi. Obwohl das Raspberry Pi auch eine grafische Ausgabe über HDMI erlaubt, werden wir diese im Rahmen des Praktikums nicht nutzen. Stattdessen wird mit dem Raspberry

Pi über SSH kommuniziert. Daher bietet sich der Net-Installer, den Sie auf Gitlab finden, für diese Aufgabe an.

Installieren Sie Raspbian auf Ihrem Raspberry Pi mit Hilfe des Net-Installers. Verbinden Sie den Raspberry Pi über ein Netzwerkkabel entweder mit Ihrem Router oder mit der zweiten Netzwerkkarte eines Poolrechners. Testen Sie die SSH und Internetverbindung des Raspberry Pi. Auf den Poolrechnern können Sie mit Hilfe von NetworkManager die Netzwerkverbindung mit dem Raspberry Pi teilen.

1.2 OpenOCD

Diese Teilaufgabe beschäftigt sich mit OpenOCD, das zur Kommunikation mit den ARM Cortex über SWD verwendet wird.

Zusammenfassung

- Kompilieren Sie OpenOCD und testen Sie die Verbindung mit dem STM32F3 Discovery Board.

OpenOCD erlaubt die Kommunikation mit einer Vielzahl an Prozessoren über JTAG oder SWD. Dafür erlaubt OpenOCD die Verwendung von verschiedenen Interfaces für die Kommunikation. Im Rahmen des Praktikums verwenden wir den Raspberry Pi als Interface. Hierfür werden die vorhanden GPIO Pins verwendet um das SWD Protokoll umzusetzen. Eine Einführung in SWD wird im Rahmen von Aufgabe 2 stattfinden. Diese Aufgabe beschränkt sich darauf, OpenOCD zu installieren und die Kommunikation mit dem STM32F3 Discovery Board zu testen.

Abseits von den Paketquellen lässt sich OpenOCD auch aus den Quellen kompilieren. Da in diesem Praktikum die aktuellste Version verwendet werden soll wird der zweite Weg gewählt. Das offizielle Repository verwendet Git und befindet sich unter `https://sourceforge.net/p/openocd/code/ci/master/tree/`. Installieren Sie OpenOCD auf dem Raspberry Pi wie in der Datei README des Repositories angegeben.

Zur Konfiguration des Interfaces (Kommunikationsgerät) und des Targets (Prozessors) verwendet OpenOCD Konfigurationsdateien. Diese finden Sie nach erfolgreicher Installation unter `/usr/share/openocd/scripts`. Zur Vereinfachung der Konfiguration erlaubt es OpenOCD eigene Konfigurationsdateien zu erstellen, die das verwendete Interface, Adaptereinstellung (SWD statt JTAG) und den Zielprozessor kombinieren. Wir verwenden als Interface die GPIO Pins des Raspberry Pi, als Prozessor einen ARM Cortex M4 und verwenden SWD. Eine solche Konfigurationsdatei namens `stm32f3discovery.cfg` könnte aussehen wie in Fig. 1 angegeben.

Eine solche Konfigurationsdatei verwenden Sie mit `openocd -f stm32f3discovery.cfg`.

Abseits der Softwareseite muss auch die Hardware verbunden werden. Die dafür nötigen Pins finden Sie in dem Datenblatt des Prozessors sowie in dem Interface File von OpenOCD. Das relevante Interface ist im oben genannten `script` Verzeichnis von OpenOCD. Das Datenblatt des verwendeten STM32F303 Prozessors beinhaltet die Pinbelegung des Prozessors ab Seite 31. Suchen Sie die passenden SWDIO und SWCLK Pins im Datenblatt und suchen Sie die dazu passenden Pins auf dem STM32F3 Discovery Board. Verbinden Sie das Board mit den passenden Pins des Raspberry Pi. Vergessen Sie nicht, die Ground Signale der beiden Boards zu verbinden.

```
interface sysfsgpio

sysfsgpio_swdio_num      6
sysfsgpio_swclk_num      5

transport select swd

source [find target/stm32f3x.cfg]

init
targets
```

Abbildung 1: OpenOCD Konfiguration für das STM32F3 Discovery mit Raspberry Pi als Interface

Führen Sie nun OpenOCD aus. Wenn alles richtig Konfiguriert und verbunden ist, sehen Sie die Prozessor ID und OpenOCD führt einen GDB Server aus. Alles weitere zur OpenOCD Nutzung wird in den folgenden Aufgaben besprochen.

Abgabe

- Dokumentieren Sie die verwendeten Pins auf dem Raspberry Pi und STM32F3 Discovery
- Dokumentieren Sie die Ausgaben von OpenOCD

1.3 Linux Kernel kompilieren

In dieser Aufgabe wird der Linux Kernel für den Raspberry Pi kompiliert und der FBTFT Treiber hinzugefügt.

Zusammenfassung

- Kompilieren Sie einen aktuellen Linux Kernel mit FBTFT Treiber als Modul

Am einfachsten lässt sich ein passender Linux Kernel direkt auf dem Raspberry Pi kompilieren, was durch die beschränkte Rechenleistung des Raspberry Pi aber sehr lange dauert. Für diese Aufgabe werden wir den Kernel stattdessen Cross-Kompiliereni. Wir benutzen also einen Compiler, der auf einem X86-64 Prozessor ausgeführt wird, aber Maschinencode zur Ausführung auf dem Raspberry Pi erzeugt.

Benutzen Sie Git, um die aktuelle Version des Kernels von https://github.com/raspberrypi/linux.git zu beziehen. Dieser Vorgang kann durch die Größe des zum Kernel gehörenden Git Repositories lange dauern.

In der Zwischenzeit kann die Toolchain für die Cross-Kompilierung des Kernels installiert werden. Diese erhalten Sie von https://github.com/raspberrypi/tools. Fügen Sie `tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-x64/bin` ihrer `PATH` Umgebungsvariable hinzu.

Im Linux repository führen Sie nun folgende Befehle aus, um die Standardkonfiguration für das Raspberry Pi zu laden:

```
KERNEL=kernel7
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bcm2709_defconfig
```

Die Parameter `ARCH` definiert dabei die Architektur des Zielsystems und `CROSS_COMPILE` den zu verwendenden Compiler mit.

Mit

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
```

können Sie nun die Konfiguration des Linux Kernels einsehen und verändern. Stellen Sie sicher, dass

```
Device Drivers -> Staging drivers -> Support for small TFT LCD display modules
```

als Modul installiert wird (m). Wenn es nicht ausgewählt ist oder es in den Kernel gebaut wird (‘*’), ändern Sie die Einstellung mit Hilfe der Leertaste auf (m). Stellen Sie außerdem sicher, dass `FB driver for the ILI9340 LCD Controller` und `Module to for adding FBTFT devices` als Modul installiert werden. Beenden Sie `menuconfig` und speichern Sie die Konfiguration. Bauen Sie den Kernel mit

```
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j4 zImage modules
```

wobei Sie `-j4` an die Anzahl an CPU Kernen ihres PCs anpassen können.

Installieren Sie nun den zuvor gebauten Kernel und die zugehörigen Module wie unter https://www.raspberrypi.org/documentation/linux/kernel/building.md beschrieben. Alternativ können Sie die selben Schritte auch über SSH ausführen und sparen sich somit das Umstecken der SD Karte.

Abgabe

- Die verwendete Kernel `.config`

1.4 Linux Modul laden

In dieser Aufgabe wird der in der vorherigen Aufgabe gebaute Kernel verwendet um das ILI9340 Display anzusteuern.

Zusammenfassung

- Laden Sie den passenden FBTFT Treiber mit dem Modul `fbtft_devices`.

Bevor der Treiber geladen wirdi, muss zuerst die physikalische Verbindung der beiden Geräte sichergestellt werden. Verwenden Sie das FBTFT Wiki (Adafruit 2.2) und GPIO Referenzi, um das Raspberry Pi mit dem Display zu verbinden.

Der verwendete Treiber bietet das Modul `fbtft_device` zur Konfiguration an. Dieses können Sie mit `modprobe fbtft_device` laden. Sehen Sie sich die Ausgabe des Moduls in `dmesg` an. Benutzen Sie das FBTFT Wiki, um ein Custom Module mit ILI9340 als Chipsatz zu laden. Bei erfolgreicher Konfiguration und Verbindung erhalten Sie ein Framebuffer Device `/dev/fb1`i, das

zur Ansteuerung des Displays verwendet werden kann. Testen Sie das Display zum Beispiel mit `con2fbmap`.

Abgabe

- Dokumentieren Sie die verwendete Pinbelegung
- Dokumentieren Sie die Konfiguration des Treibers

1.5 FBTFT in den Kernel kompilieren

In dieser Aufgabe wird der FBTFT Treiber in den Linux Kernel kompiliert und über Devicetrees gesteuert.

Zusammenfassung

- Kompilieren Sie einen aktuellen Linux Kernel mit FBTFT Treiber als Bestandteil
- Verwenden Sie einen Devicetree, um den Treiber zu laden

Kompilieren und installieren Sie wie oben gelernt den Linux Kernel erneut. Kompilieren Sie den Display Treiber als Bestandteil (*) anstatt als Modul (m).

Device Trees stellen eine Möglichkeit dar, Treiber direkt beim Starten des Kernels zu kompilieren. Diese Listen hierarchisch die Parameter der verwendeten Peripheriekomponenten (in diesem Fall SPI Hardware) und laden die passenden Treiber mit den angegebenen Einstellungen. Ein Beispiel finden Sie unter https://github.com/notro/fbtft/blob/master/dts/overlays/rpi/hy28a-overlay.dts. Erstellen Sie einen Device Tree mit den in 1.4 herausgefunden Parametern. Kompilieren und pflegen Sie ihren erstellten Tree wie unter https://github.com/notro/fbtft/wiki/FBTFT-RPI-overlays beschrieben ein.

Wenn alles richtig konfiguriert ist sollten Sie beim Booten des Systems bereits eine Konsolenausgabe auf dem Display sehen.

Abgabe

- Device Tree für das ILI9340 Display