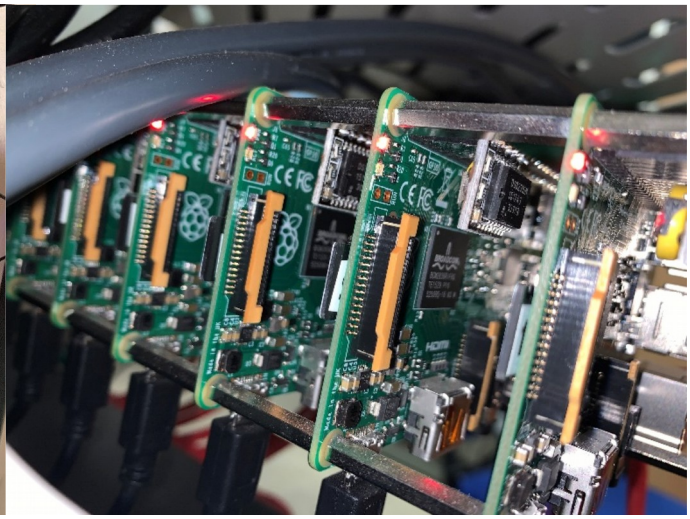
Universität Kassel
Fachgebiet Rechnerarchitektur & Systemprogrammierung
Wilhelmshöher Allee 71,
34121 Kassel

Labor Embedded Systems
im
Fachgebiet Rechnerarchitektur und Systemprogrammierung
SS 2020

Beschreibung der Aufgabenstellung und Spezifikation



v1.0 – 10.06.2020

1. Einleitung

Im Rahmen des Labor-Praktikums Embedded Systems / Systemprogrammierung im SS 2020 soll in eigenständiger Gruppenarbeit von 2 Personen ein Linux Treiber für die Real-Time-Clock (RTC) der Praktikumshardware, ein Raspberry Pi mit einer DS3231 RTC, entwickelt werden.

Der Treiber soll das Auslesen und Beschreiben der DS3231 Hardware-Uhr ermöglichen. Die Entwicklung des Treibers soll innerhalb eines strukturierten und ingenieurmäßigen Entwicklungsprozesses stattfinden. Aus diesem Grund ist es notwendig, dass im Studium vermittelte Wissen im Bereich der systematischen Systemsoftware-Entwicklung anzuwenden. Der Entwicklungsprozess muss sorgfältig geplant werden und alle vorgenommenen Schritte und Entscheidungen müssen ausführlich dokumentiert werden. Dabei sollen die Programmierung, Strukturierung und die Dokumentation nach Coding-Guidelines erfolgen, die dem MISRA-C Standard angelehnt / entnommen sind. Dadurch werden wichtige und hilfreiche Leitlinien bekannt gemacht, die gute und professionelle Programmierung ausmachen.

Die Software des Projekt-Systems basiert auf dem Betriebssystem Linux und der Kernelversion 4.9.x. Die Hardware-Echtzeituhr (RTC) des Systems speichert die aktuelle Uhrzeit auch bei ausgeschaltetem Zustand. Bei Systemstart holt sich die Softwareuhr die aktuelle Uhrzeit vom RTC-Chip. Alle zeitgesteuerten Aufgaben des Systems werden dann üblicherweise von der Softwareuhr behandelt.

Die Hardwareuhr ist mittels des I²C Bus an das System (Raspberry Pi) angeschlossen. Dieser Bus arbeitet seriell mit zwei Leitungen. Eine Leitung ist für den seriellen Daten/Adressen Ein- und Ausgang verantwortlich. Die zweite Leitung liefert das Taktsignal. Der Zugriff auf den I²C Bus erfolgt mit Hilfe der „i2c-core“ Schnittstelle des Linux Kernels. Der Datenaustausch zwischen dem Master und den Clients kann entweder direkt erfolgen oder mit Hilfe des SMBus (System Management Bus) Protokolls (siehe die Kernel Dokumentation `Documentation/i2c/writing-clients`). Der Systemsoftware-Entwickler ist dafür zuständig den Client korrekt beim System zu registrieren, so dass die Hardware, die am Bus angeschlossen ist, verwendet werden, und somit als Teil der Systemsoftware betrachtet werden kann.

2. Vorgehensweise

Der Treiber soll in selbstständiger Gruppenarbeit entwickelt werden. Es werden Gruppen bestehend aus 2 Studenten gebildet. Diese Gruppen entwickeln jeweils einen eigenen Treiber. Die Gruppen arbeiten dabei selbstständig. Bei nicht selbstständiger Arbeit, d.h. z.B. wenn zwei (oder mehr) Gruppen dieselbe Lösung abgeben bzw. eine Lösung die in einem der vergangenen Semester eingereicht wurde, so wird diese Lösung für alle beteiligten mit n.b. bewertet.

Innerhalb einer Gruppe arbeiten die Studenten zu gleichen Teilen an der Lösung der aktuellen Aufgabenstellung. Der Treiber und die Dokumentation werden demnach zusammen entwickelt.

Innerhalb der Übungen findet eine Einführung in das System statt. Dazu gehört eine kurze Einführung in das zugrunde liegende Linux Betriebssystem und die wichtigsten Befehle für die Kommandozeile. Außerdem wird auf die Besonderheiten des Systems eingegangen. Die wichtigsten Informationen zu den benötigten Entwicklungstools, dazu gehören u.a. Editor, Compiler und Übersetzungsautomatisierung, werden ebenfalls vermittelt. Die benötigten Kenntnisse der systematischen Systemsoftware-Entwicklung werden eigenständig erarbeitet und bei Bedarf in den entsprechenden Übungen behandelt.

Die Anforderungsspezifikation wird in diesem Dokument mitgegeben. Diese Spezifikation beschreibt die Funktionalität, die der Treiber bereitstellen soll und wie der Treiber reagieren soll, wenn z.B. falsche Parameter übergeben werden sollen.

Zusätzlich gibt es eine Liste mit Coding-Rules. Diese sind Regeln, die an den Programmierstandard MISRA-C angelehnt sind. MISRA-C ist ein nicht mehr wegzudenkender Programmierstandard der nicht mehr nur in der Automobilindustrie zur Softwareentwicklung Verwendung findet, sondern überall dort, wo zuverlässige Software benötigt wird. Diese Regeln stellen lediglich einen kleinen Auszug dar, sollten nicht als vollständig oder nicht erweiterbar betrachtet werden. Sie sollen ein erstes Kennenlernen mit einem solchen Programmierstandard erleichtern. Gleichzeitig soll dadurch das professionelle Programmieren und Dokumentieren von Anfang an nach einem Industriestandard gelernt und verinnerlicht werden. Das Einhalten der Coding-Rules in diesem Kurs ist zwingend erforderlich, und keine Option.

Jede Gruppe erhält einen einzigartigen Benutzerzugang inklusive Passwort, mit der sie sich per SSH auf ein Board einloggen kann. Jeder Benutzer enthält eine Vorlage für den Treiber (Grundgerüst), welches wichtige Datenstrukturen und die Deklarationen der benötigten Funktionen enthält. Davon ausgehend muss ein Design erarbeitet werden. Hierbei wird festgelegt welches Element welche Aufgabe wie erledigt. Zusätzliche Datenstrukturen und Funktionen müssen ggf. in das Modell eingearbeitet werden. Außerdem muss die Vorlage an die Coding-Rules angepasst werden, sodass am Ende der gesamte Code inklusive Kommentaren coding-rule-konform ist.

3. Anforderungsspezifikation

Dieser Abschnitt beschreibt die Anforderungen an den RTC-Treiber.

1. Laden des Moduls

Beschreibung

Das Modul muss sich korrekt beim Kernel anmelden und registrieren. Das Modul wird als Device `dev/<device>` in das System eingebunden. Die benötigten Schnittstellen und Datenstrukturen werden korrekt initialisiert. Der Status der Registrierung wird über die Kernel-Messages ausgegeben. Jede Kernel-Message beginnt mit dem Namenskürzel des Treibers. Jede Nachricht bekommt eine eigene Zeile, sodass die Kernel-Messages einfach zu lesen sind.

Abnahme

Die Abnahme erfolgt direkt über die entsprechenden Kernel-Messages und indirekt über die korrekte Funktion des Moduls.

2. Entfernen des Moduls

Beschreibung

Das Modul muss korrekt aus dem Kernel entfernt werden. Alle allozierten Ressourcen, wie Speicher, Zugriff auf Bussysteme, usw., müssen freigegeben werden. Der Status des entfernens wird über die Kernel-Messages ausgegeben.

Abnahme

Die Abnahme erfolgt direkt über die entsprechenden Kernel-Messages und über Reviews des Source-Codes.

3. Auslesen von Datum und Uhrzeit aus dem RTC-Chip

Beschreibung

Die aktuellen Werte für Datum und Uhrzeit werden über die entsprechende `read()`-Funktion aus dem Chip ausgelesen. Der Treiber liefert die Daten im folgenden (24 Stunden-) alphanumerischen Format:

```
DD. M hh:mm:ss YYYY
```

mit DD – Tag (2-stellig)
M – Monat (vollständig ausgeschrieben als Januar, Februar, Maerz, usw.)
YYYY – Jahr (4-stellig)
hh – Stunde (2-stellig)
mm – Minute (2-stellig)
ss – Sekunde (2-stellig)

Beispiel: `15. April 19:34:56 2019`

Abnahme

Die Abnahme erfolgt über den Test mit Linux Systembefehlen (z.B.: cat) und über den Test mit Linux Bibliotheksfunktionen (z.B.: `open(/dev/<device>)`, `read()`).

4. Schreiben von Datum und Uhrzeit in den RTC-Chip

Beschreibung

Eigene Werte für Datum und Uhrzeit werden über die entsprechende `write()`-Funktion in den Chip geschrieben. Der Treiber verarbeitet die Daten im folgenden (24 Stunden-) numerischen Format:

`YYYY-MM-DD hh:mm:ss`

mit DD – Tag (2-stellig)
MM – Monat (2-stellig)
YYYY – Jahr (4-stellig)
hh – Stunde (2-stellig)
mm – Minute (2-stellig)
ss – Sekunde (2-stellig)

Beispiel: `2019-04-15 19:34:56`

Ungültige Formate werden erkannt. Es erfolgt kein schreibender Zugriff auf den RTC-Chip. Bei einem ungültigen Format bzw. einer ungültigen Uhrzeit muss der Treiber den Fehler „`-EINVAL`“ (Invalid Argument) ausgeben.

Abnahme

Die Abnahme erfolgt über den Test mit Linux Systembefehlen (z.B.: `cat`) und über den Test mit Linux Bibliotheksfunktionen (z.B.: `open(/dev/<device>)`, `write()`), sowie Anhand der Kernel-Messages.

5. Gleichzeitiger Zugriff auf den RTC-Chip

Beschreibung

Mehrere Prozesse können „gleichzeitig“ lesend bzw. schreibend auf den Chip zugreifen ohne die Daten zu verfälschen. Es treten keine race conditions auf. Bei einem bereits verwendeten Bus / Device muss als Fehler „`-EBUSY`“ (Device or resource busy) zurückgegeben werden.

Abnahme

Die Abnahme erfolgt wenn „gleichzeitiger“ Schreib- Lesezugriff korrekte Daten zurück liefert, sowie Anhand der Kernel-Messages.

6. Einschränkung des Wertebereiches

Beschreibung

Der Wertebereich des RTC-Chips ist für die Anforderungen 3 und 4 beschränkt auf die Zeit vom 01. Januar 2000 um 00:00:00 Uhr bis zum 31. Dezember 2199 um 23:59:59 Uhr. Werte außerhalb dieses Wertebereiches werden abgelehnt und mit einer Fehlermeldung „`-EOVERFLOW`“ beantwortet.

Abnahme

Die Abnahme erfolgt mit entsprechenden Tests, sowie Anhand der Kernel-Messages.

7. Behandlung von falschen bzw. unlogischen Daten

Beschreibung

Falsche Werte in der Eingabe, z.B.: Minute 75, Minute -3, Stunde 26, oder Tag 32 werden vom Treiber erkannt und abgelehnt. Eine Fehlermeldung „`-ENOEXEC`“ wird zurückgegeben. Bitte achten Sie unbedingt auch auf **Schaltjahre** in dem gültigen Zeitraum!

Abnahme

Die Abnahme erfolgt mit entsprechenden Tests, sowie Anhand der Kernel-Messages.

8. Statusinformationen

Beschreibung

Bei jedem regulären Zugriff auf den Treiber werden folgende Statusinformationen aus dem RTC-Chip gelesen und in einer entsprechenden C-Datenstruktur gespeichert:

- OSF-Flag
- BSY-Flag
- Temperatur des RTC-Chip in °Celsius als vorzeichenbehaftete Ganzzahl

Ebenfalls muss bei jedem Zugriff das OSF-Flag getestet werden. Wenn dieses nicht aktiviert ist, muss es aktiviert, sowie eine entsprechende Kernel-Message erzeugt werden. Der derzeitige Zugriff, unabhängig ob lesend oder schreibend muss abgebrochen werden. Die Fehlermeldung „`-EAGAIN`“ muss zurückgegeben werden.
Sollte die Temperatur des RTC-Chip größer als 85°C, oder kleiner als -40°C sein, wird eine entsprechende Kernel-Message ausgegeben, der derzeitge Zugriff jedoch wird ausgeführt. Für einen einfachen Test, **muss** eine Möglichkeit implementiert werden, um den Temperaturwert der C-Datenstruktur per „`write()`“ manipulieren zu können. Diese funktioniert so, dass als erstes ASCII-Zeichen „$“ übergeben wird, gefolgt vom neuen dreistelligen vorzeichenbehafteten zu schreibenden Temperaturwert (z.B. „`$-46`“, oder „`$101`“).

Abnahme

Die Abnahme erfolgt mit entsprechenden Tests, sowie Anhand der Kernel-Messages.

4. Coding-Rules

Folgende Coding-Rules sollen zur Programmierung des Treibers in diesem Labor-Praktikum angewendet bzw. eingehalten werden. Diese Regeln sind an den MISRA-C (2004) Standard angelehnt und sollen lediglich einen kleinen Teil der dort enthaltenen Regeln (über 120) und Richtlinien widerspiegeln.

Regel 1.1: Im Source Code darf nur `/* … */` als Kommentarstil verwendet werden.

Andere Kommentarstile wie „ `//` „ sind nicht erlaubt.

Regel 1.2: Innerhalb von Kommentaren darf die Zeichenfolge `/*` nicht verwendet werden.

Regel 1.3: Auskommentierter Quellcode ist nicht erlaubt

Wenn es notwendig erscheint, darf das Kompilieren von Quellcode durch `#if / #ifdef` Konstrukte bestimmt werden.

Regel 2.1: Bezeichner (Variablennamen bzw. Funktionsnamen) dürfen nicht länger als 31 Zeichen lang sein.

Außerdem müssen die Namen leicht voneinander unterscheidbar sein, um eine Verwechslung zu verhindern.

Regel 2.2: Alle Bezeichner müssen einmalig sein und dürfen nicht wiederverwendet werden.

```
struct my_info
{
      uint16_t status; */ Status Information über … */
} * x;

uint16_t status = 1; */ ← nicht erlaubt da Bezeichner doppelt verwendet! */
```

Regel 3.1: Der Datentyp `char` darf nur für die Speicherung von Alphanumerischen Zeichen verwendet werden.

Die einzig erlaubten Operationen mit dem Datentyp `char` sind folgende: `=, ==, !=`

Regel 3.2: Der Datentyp `unsigned` und `signed char` darf nur für die Speicherung von numerischen Werten verwendet werden.

Regel 4.1: Wenn eine Funktion oder ein Objekt deklariert oder definiert wird, muss sein Datentyp explizit angegeben werden.

Regel 4.2: Für jeden Übergabeparameter einer Funktion muss der Datentyp in der Deklarationen und der Definition übereinstimmen. Die Rückgabewerte müssen ebenfalls identisch sein

Regel 4.3 Es dürfen keine Definitionen von Objekten in Header-files angelegt werden. Lediglich Deklarationen sind in Header-files erlaubt.

Regel 4.4: Objekte müssen innerhalb einer Funktion definiert werden, wenn sie nur innerhalb dieser Funktion verwendet werden.

Die Verwendung von globalen Objekten muss klar begründet werden und sollte vermieden werden.

5. Einführung in die Arbeitsumgebung

Linux Kommandos

Folgende Tabelle zeigt eine sehr kurze Übersicht über die notwendigen Kommandozeilenbefehle, die Sie mindestens kennen müssen. Mit diesen wenigen Befehlen, sollten Sie in der Lage sein, sich auf dem Board zurechtzufinden und mit der Programmierung zu beginnen. Die Tabelle ist nur als Hilfestellung anzusehen, weitere Befehle könnten notwendig sein.

Für die Verwendung des Editors VIM, wird das VIM-cheat-sheet empfohlen. Dieser Editor ist nach einer Eingewöhnungsphase sehr praktisch und ermöglicht schnelles und effizientes arbeiten.

Kommando	**Beschreibung**
`cd`	change directory / Verzeichnis wechseln
`ls (-la)`	list / Verzeichnisinhalt anzeigen lassen
`unzip Datei.zip`	eine Zip-Datei entpacken
`make`	Programm „make“ aufrufen (um Treiber zu übersetzen)
`lsmod`	Anzeigen der derzeit geladenen Kernel-module
`dmesg`	Anzeigen des Kernel-logs
`sudo „Kommando“`	Ein Kommando als „Super-user“ ausführen (Administrator)
`sudo insmod ds3231.ko`	Laden des Kernelmoduls
`sudo rmmod ds3231`	Entfernen des Kernelmoduls
`„STRG“ + „C“`	Abbruch eines Prozesses im Userspace
`cat /dev/ds3231`	Lesen aus der Character Device Datei
`echo „daten“ > /dev/ds3231`	Schreiben der Zeichenkette „daten“ in die Character Device Datei
`vim file.c`	Editor „VIM“ starten und die Datei mit dem Namen „file.c“ öffnen
`nano file.c`	Editor „nano“ starten und die Datei mit dem Namen „file.c“ öffnen

1. Bewertung

Die Note für das Praktikum Systemprogrammierung im SS 2020 setzt sich wie folgt zusammen:

- Kommentierter Quellcode des Treibers, Umfang und Qualität
- Funktionalität des Treibers (Testfälle und Testergebnisse)
- Einhalten der Anforderungsspezifikation
- Einhalten und Umsetzen der Coding-Guidelines