Abgabe Iteration 2

Projektplanung

Die Projektplanung wurde mit dem Online-Tool “Comindwork” gemacht. Die relevanten Zugangsdaten sind untenstehend zu finden:

Link**:** http://controlledclock.comindwork.com/
User: *zhaw_view*
Passwort: *dcfzhaw2014*

Da das Online-Tool nicht ganz unsere Erwartungen erfüllt und nicht ganz einfach zum Bedienen ist, pflegen wir in einem separaten Google-Drive-Dokument noch eine Liste mit den geleisteten Aufwänden.
Tracking-Sheet

Projektverwaltung

Das Projekt inklusive Code, Dokumentation, etc. wird in folgendem Git-Repository gepflegt:
https://github.com/MrJack91/controlledClock

Projektdokumentation

Die Projektdokumentation für die zweite Abgabe ist unter folgenden Link abrufbar:
https://www.dropbox.com/s/l53rpsmxdo41qco/ControlledClock_Milestone2_DanielBrun_MichaelHadorn.pdf

Über ein Feedback zu unserer aktuellen Dokumentation und dem Projektverlauf würden wir uns wieder freuen. So können wir die kommenden Iterationen und die Dokumentation weiter verbessern.

Rückblick & Fazit

Entnommen aus unserer Projektdokumentation (Kapitel 2).

Nach Abschluss der ersten Iteration hatten wir mit Syrus Mozafar und George Brügger je ein separates Treffen bezüglich dem aktuellen Projektstand. Aus beiden Treffen konnten wir wertvolle Inputs für den weiteren Projektverlauf mitnehmen.

Beim Treffen mit George Brügger sind wir zum Schluss gekommen, dass wir ein Replan- ning und weitere Verfeinerung der Aufgabenstellung machen müssen. Die Dokumentation und der Zeitplan wurden entsprechend angepasst. Auch kamen wir zum Schluss, dass eine Implementation des Backend in C unumgänglich ist. Da wir beide noch nie grössere, bzw. komplexere Programme in C geschrieben haben, stellt dies für uns eine weitere Herausforde- rung dar. Als Konsequenz davon rechen wir damit, dass wir nicht alle (Zusatz-)Funktionen wie

geplant umsetzen können.

Aktuell stellt vor allem das Auslesen des System-Tics eine grosse Hürde für uns dar. Nach sehr intensiven und langwierigen Recherchen und einigen Dutzend ausprobierten Beispielcodes sind wir zum Schluss gekommen, dass der System-Tic nur über eine Interrupt Service Routine ausgelesen werden kann. Die Interrupt Service Routine wiederum muss als Treiber implementiert werden, da auf die entsprechenden Bereiche des Betriebsystems aus dem User-Mode heraus nicht zugegriffen werden kann. Die Ausnahme bildet hier eine allfällige Implementation in Assembler.

Da wir beide über unterschiedliche Notebooks und Betriebssysteme (Win 7 und Mac OS) verfügen, haben wir einen Versuch mit einem Raspberry Pi gestartet. Mit dem Raspberry verfügen wir über die gleichen Voraussetzungen und haben im Detail die Kontrolle, bzw. das Wissen, über die verwendete Hardware. So können wir uns bei der Implementation auch auf ein OS und eine Hardware beschränken. Niemand von uns hat bisher jemals auf diesen Ebenen programmiert. Dies ist eine spannende Herausforderung, jedoch ist Ungewiss, ob wir unser Ziel auch wirklich erreichen können.

Auch beim „Read-“ und „Decode-Teil“ haben wir uns ein wenig überschätzt. Zur Zeit der ersten Planung wussten wir noch gar nicht, ob wir das Backend in C oder Java umsetzen würden. Das Read und Decode war einiges komplizierter als wir uns das vorgestellt haben. Da war zum Beispiel die fortlaufende Suche nach korrekten Zeitstempeln. Momentan haben wir zur Definition eines korrekten Zeitstempels „nur“ die Kontrolle der Parität eingebaut, dewegen erkennen wir fast viermal mehr Zeitstempel, als es gültige Zeitstempeln im Signal hat. Dies muss verbessert werden.

Aktueller Planungsstand

Auch der Termin für die zweite Abgabe des Meilensteins ist schneller da gewesen, als wir uns das gedacht hatten. Vor allem, da wir etwas mit der Programmiersprache zu kämpfen hatten. Vom String-Handling, über Speicherallozierung bis hin zu Problemen mit Libraries, die es nur für bestimmte OS-Versionen, OS-Typen oder Compiler gibt. Die Probleme waren grundsätzlich sehr spannend und herausfordernd. Jedoch mussten wir in die Problemfindung einige Zeit investieren.