Fachhochschule Wedel

- University of Applied Sciences -

Praktikum Echtzeitsysteme
Dokumentation

Informatik

Aufgabe:

Grillfest

Betreuer:
Dipl.-Ing. Timm
Bostelmann
Fachhochschule Wedel
Feldstraße 143
22880 Wedel
(04103) 80 48 – 31
bos@fh-wedel.de

Verfasser:
Oliver Westphal
inf100849@fh-wedel.de

Verfasser:
Lennart Kindermann
inf1073@fh-wedel.de

Eingereicht am: 22.9.2016

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

In diesem ersten Kapitel dieser Arbeit soll zunächst die Motivation für ihre Erstellung dargestellt werden. Nachdem dann ihr grober Aufbau erklärt wird, folgt noch eine allgemeine Beschreibung der Aufgabe, die einen Überblick über ihre Problemstellung geben soll.

1.1 Motivation

Die Motivation dieser Arbeit war die Erfüllung der Veranstaltung *Praktikum Echtzeitsysteme* für zwei ECTS Punkte. In diesem Semester sollten im Rahmen des Praktikums zum einen mithilfe verschiedener Kommunikationsdienste als echtzeitfähiges Programm zu einer gegebenen Problemstellung realisiert werden. Die zweite Teilaufgabe war diese ergänzende Dokumentation zum Ablauf der Bearbeitung der Aufgabe.

1.2 Aufbau der Dokumentation

In dieser Dokumentation werden die Prozesse der verschiedenen Entwicklungsphasen für die Aufgabenstellung beschrieben. Nach der Einführung, folgen angefangen mit der Darstellung und Analyse der Aufgabenstellung die Planung der einzelnen Tasks und Datenstrukturen. Anschließend wird noch beschrieben, wie

die Implementierung des entstandenen Konzepts realisiert wurde, ergänzt durch eine Übersicht der Tests die hierfür erstellt und genutzt wurden. Zuletzt enden wir mit einer persönliche Bewertung der Aufgabe und der Abläufe in unserer Gruppe, sowie einem kurzen Fazit zum gesamten Praktikum.

1.3 Allgemeine Problemstellung

Dieses Semester wurde die Aufgabe *Grillfest* gestellt. Es war eine echtzeitfähige Simulation des Grillprozesses zu realisieren. Der Prozess sollten vom Erstellen der Würste, über das Legen auf den Grill und dem Bräunen, bis zum Verzehr dargestellt werden.
Der Ablauf sollte über verschiedene Akteure funktionieren, die entweder physikalischer oder persönlicher Natur sein konnten und vom Benutzer mehr oder weniger zu beeinflussen waren.
Eine genauere Erklärung der Problemstellung folgt im nächsten Kapitel.

2
Planung

2.1 Problemstellung

Ziel des Echtzeitsysteme Praktikums war es eine Grill Simulation zu entwickeln. Das Hauptaugenmerk der Simulation lag auf den Grillwürsten, die von verschiedenen Prozessen behandelt werden. Diese Prozesse sind teilweise physikalischer Natur (z.B. das Bräunen), teilweise handelt es sich auch um „Akteure“ (Fleischer, Grillmeister und Feuerwehr). Jeder dieser natürlichen Prozesse wird in der Simulation auf einen Task abgebildet. Dort, wo die natürlichen Prozesse Informationen und Gegenstände austauschen, werden die Tasks entsprechende Daten austauschen.
Diese Kommunikation zwischen den Tasks ist Kern dieser Aufgabe!
Jede Wurst durchläuft während der Simulation mehrere Stationen an denen sie von den unterschiedlichen Prozessen behandelt wird. Nach ihrer Erzeugung durch den Fleischer liegt die Wurst in einer Kühlbox. Dort verbleibt sie so lange, bis sie durch Eingreifen des Benutzers entnommen und dem Grillmeister übergeben wird. Der Grillmeister platziert die Wurst auf dem Grill. Die Wurst wird nun von einer Seite gebräunt (sie hat insgesamt vier zu bräunende Seiten), bis sie vom Grillmeister gedreht oder entfernt wird. Wenn eine Wurst von einer Seite zu lange gebräunt wird, platzt sie und entzündet den Grill. Der Grill kann nur von der Feuerwehr gelöscht werden.

2.2 Analyse

2.2.1 Tasks

Aus der Problemstellung lassen sich folgende, als Tasks zu modellierende, Programmteile identifizieren.

- Fleischer (Produziert Würste)
- Grillmeister (Verwaltet die auf dem Grill befindlichen Würste)
- Feuerwehr (Prüft und löscht den Grill)
- Physik (Bräunt Würste und entzündet den Grill)
- Eingabe (Abarbeiten von Benutzereingaben)
- Ausgabe (Gibt den Status der Grill-Simulation aus)

Die Prioritäten wurden nach Analyse der Aufgabestellung so angepasst, dass sich die Tasks nicht unnötig blockieren und jeder Task die Chance hat vom Scheduler erfasst zu werden.
Zudem wurden Ein- und Ausgabe so priorisiert, dass Eingaben schnell erkannt werden und die Ausgabe den Rest der Simulation nicht blockiert.

2.2.2 Prozesskommunikation

Damit Tasks untereinander Kommunizieren können müssen Kommunikationsdienste eingerichtet werden, die andere Tasks nicht blockieren und den reibungslosen Ablauf der Simulation gewährleisten.

2.2.2.1 Kühlbox

Die Kühlbox soll von dem Fleischer-Task befüllt und von dem Grillmeister-Task geleert werden. Hierfür eignet sich eine Nachrichten-Warteschlange, da hier nur Objekte (Würste) rein- und rausgenommen werden und keine größeren kritische Abschnitte entstehen.

Im Gegensatz zur Semaphor können in der Warteschlange direkt Objekte hinterlegt werden und eine zusätzliche Datenstruktur kann vermieden werden.

2.2.2.2 Grill

Der Grill wird permanent vom Grillmeister-, Physik und Feuerwehr-Task auf verschiedene Parameter und einzelne Würste abgefragt und verändert.
Hier wurde eine durch ein Mutex-Semaphor abgesicherte Datenstruktur geplant, da hier größere kritische Abschnitte, wie das sequentielle Bräunen aller Würste, entstehen.

2.2.2.3 Fleischer und Grillmeister

Um die Benutzereingabe zu realisieren müssen Fleischer und Grillmeister vom Eingabe-Task aus gesteuert werden können.

Die erste Überlegung, die Kommunikation über Semaphore zu realisieren, wurde nach kurzer Evaluation verworfen. Hierfür hätten zusätzlich Semaphore zum Absichern der Zustände von Fleischer und Grillmeister eingeführt werden müssen, die die Komplexität syntaktisch als auch semantisch drastisch erhöht und die Erweiterbarkeit beschränkt hätten.

Da der Nachrichtenaustausch nur jeweils in Richtung Fleischer- und Grillmeister-Task geschehen muss, wurde jeweils eine Nachrichten-Box pro Task angestrebt um so die Nutzereingabe durch versenden von Nachrichten abzubilden.

2.2.3 Zusammenfassung

Nach Analyse der Problemstellung wurde ein Diagramm zur Visualisierung der Prozesskommunikation, inklusive der zu verwendenden Kommunikationsdienste, erstellt.

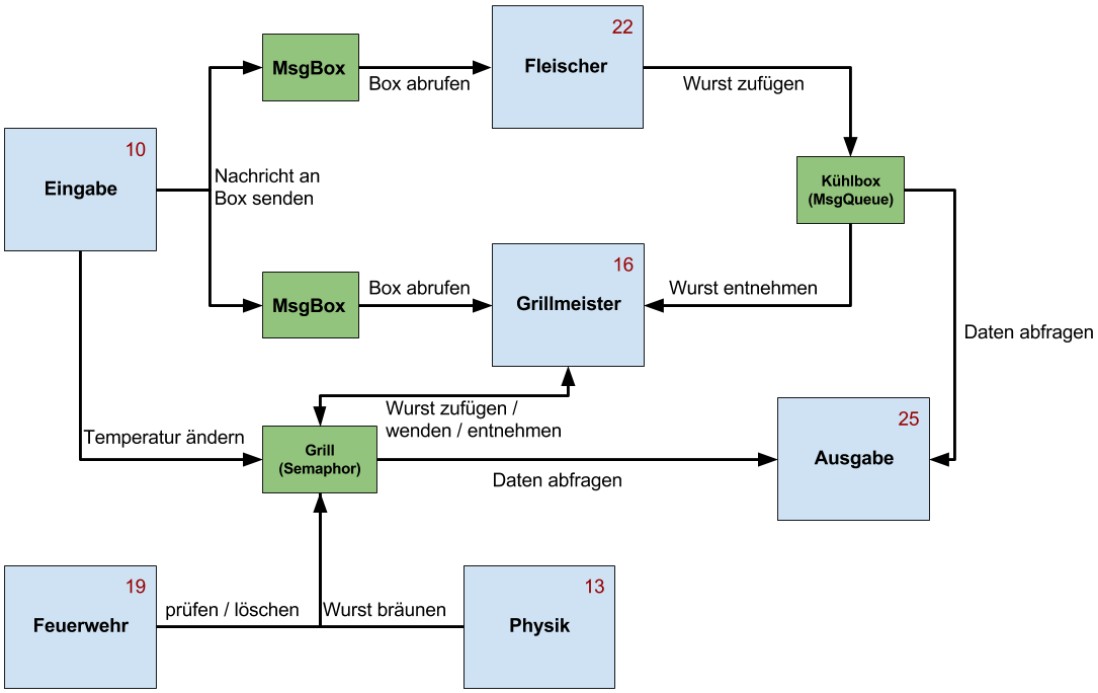

Abbildung 2.1: IPC

3 Implementierung

Das Kapitel Implementierung beinhaltet neben der Darstellung der Abläufe unserer Gruppe während der Realisierung des Konzepts, das aus der Analyse und Planung entstanden ist, noch eine Auflistung der unterschiedlichen Programmtests die bei eben dieser Realisierung hilfreich waren und eine Bedienungsanleitung, um die Steuerungsmöglichkeiten bei der Eingabe des Benutzers zu erläutern.

3.1 Datenstrukturen

3.1.1 Wurst

Eine Wurst besteht aus 4 Seiten die jeweils einen (float) Bränungsgrad haben und einer Seite auf der sie sich gerade befindet.

Die Würste wurden als struct implementiert, dessen Zeiger zur Verarbeitung in die Kühlbox-Nachrichte-Warteschlange bzw. in die Grill-Datenstruktur gegeben wird.

```
/* Wurst Datenstruktur */
typedef struct {
  float seiten[4];
  INT8U seite;
} Wurst;
```

```
typedef Wurst *wurst_ptr;
```

Der Speicher für die Würste wird über µC/OS-II Partitionen dynamisch alloziiert.

```
wurst = OSMemGet(WurstBuff, &err);

/* Wurst Seiten initialisieren. */
wurst->seiten[0] = 0;
wurst->seiten[1] = 0;
wurst->seiten[2] = 0;
wurst->seiten[3] = 0;
wurst->seite     = 0;
```

3.1.2 Nachrichten

Nachrichten an den Fleischer und Grillmeister werden durch ENUMs realisiert, derren Pointer in die jeweilige Nachrichtenbox gegeben werden.

```
/* Nachrichtentypen */
typedef enum {
  START_TRINKEN = 1,
  STOP_TRINKEN  = 2,
  PLACE_WURST   = 3,
  CREATE_WURST  = 4
} MsgTypes;

typedef MsgTypes * msg_ptr;
```

Der Speicher für die Nachrichten wird über µC/OS-II Partitionen dynamisch alloziiert.

```
msg_ptr create_msg (MsgTypes msg_type) {
  INT8U  err;
  msg_ptr msg;

  OSSemPend(MsgSema, 0, &err);
  msg  = OSMemGet(MsgBuff, &err);
  *msg = msg_type;
  OSSemPost(MsgSema);

  return msg;
```

```
}
```

3.1.3 Kühlbox

Die Kühlbox dient dem Zwischenspeichern von Würsten bzw. derren Pointern.

Sie ist indirekt über eine Nachrichtenbox (MsgQ) realisiert.

3.1.4 Grill

Ein Grill besteht aus einer Datenstruktur für Würste, der Anzahl der Würste, der aktuellen Temperatur und dem aktuellen Status (grillt/brennt).

Auch er Grill wurde als struct implementiert, auf ihm wird jedoch nur indirekt über das Grill-Modul und dessen Funktionen gearbeitet.
Der Grill muss über die Modul-Funktionen per Semaphor gesperrt/entsperrt werden.

```
typedef struct {
  wurst_ptr wuerste[MAX_GRILL];
  INT8U     anzahl;
  BOOLEAN   brennt;
  INT16U    temperatur;
} Grill;
```

3.2 Programmiertagebuch

Im folgenden Abschnitt sollen die Termine, an denen wir uns als Gruppe im Labor für die Implementierung des Programmcodes für die Hauptaufgabe getroffen haben, einzeln beleuchtet werden und dabei sowohl die Fortschritte als auch unerwartet auftretende Probleme dargestellt werden.
Die Lösung der vorbereitenden Übungsaufgabe wird hier nicht näher beschrieben.

3.2.1 Vorarbeiten

Nach der Veranstaltung zur Aufgabenvorführung in der 22. Kalenderwoche wurde sofort die Planung der Abläufe geplant. Da wir beide noch durch verschiedenen andere Übungen beschäftigt waren, einigten wir uns die Bearbeitung nach Beendigung dieser zu beginnen.
Die Übungsaufgabe bearbeiteten wir bereits beim ersten Orientierungstreffen vor der Klausurenphase, bei dem wir uns mit dem DE-Board und den User Manuals vertraut machten und auch die Termine der weiteren Bearbeitung vereinbarten, sowie individuelle theoretische Vorarbeiten für die Zeitraum bis zum ersten Treffen besprachen.

3.2.2 Erstes Treffen - 1.9.

Wir beide hatten uns zuvor bereits unabhängig voneinander Gedanken zur Aufteilung der Tasks und der besten Lösung der Funktionalität durch einen der verschiedenen Kommunikationsdienste gemacht. Beim ersten Treffen brachten wir diese nun zusammen. Zum besseren Verständnis fertigen wir eine Skizze an, welche die Aufgaben der Tasks und deren Kommunikation untereinander darstellen sollte, siehe Kapitel Planung; Abbildung 1.

Nachdem wir die letzte theoretische Vorplanung abgeschlossen hatten, begannen wir mit dem praktischen Teil der Implementierung. Zuerst schufen wir die Funktionalität des Fleischers, da es uns sinnvoll erschien in der Reihenfolge des Verlaufs eines Wurstzyklus die Tasks abzuarbeiten. Beim Fleischer traten keine größeren Probleme auf, sodass wir unser Konzept in diesem Bereich nicht anpassen mussten.
Nachdem der Fleischer implementiert war, testeten wir die Produktion der Würste. Bei diesem Test traten wie vermutet noch keine Fehler auf, weil der Tasks bis jetzt noch alleine arbeiten konnte, und somit über die gesamten Zugriffszeiten auf die Speicher verfügen konnte.

Als nächsten begannen wir mit der Realisierung des Grillmeisters, der durch sei-

ne vielen verschiedenen Funktionen und Zugriffe sicherlich den umfangreichsten Tasks darstellte, sodass wir diesen am ersten Termin nicht mehr fertig implementierten. Wir schufen lediglich die Funktionalitäten zum entnehmen der Würste aus der Kühlbox und dem Platzieren auf dem Grill, und gingen somit weiterhin chronologisch vor.
Die Eingabe-Benutzerschnittstelle realisierten wir entsprechend der für den aktuellen Task benötigten Eingabemöglichkeiten.

3.2.3 Zweites Treffen - 8.9.

Zu Beginn des zweiten Treffens, testen wir das Zusammenspiel der bisherigen zwei Tasks Fleischer und Grillmeister. Da der Grillmeister eine höhere Priorität als der Fleischer hatte, und somit den Zugriff auf die Kühlbox bei Bedarf übernahm, konnte eine korrekte Funktionalität festgestellt werden.
Danach erweiterten wir den Grillmeister um die Möglichkeit die sich auf dem Grill befindlichen Würste auf ihren Bräunungsgrad hin zu kontrollieren. Da der Grillmeister hierdurch immer noch als einziger Task auf den Grill als Speicher zugriff, funktionierte auch dies ohne größere Komplikationen.

Dieser alleinige Zugriff sollte durch das folgende Hinzufügen des Physik-Tasks nun gestört werden. Nachdem die Funktionalität zum Bräunen der Würste implementiert war, testeten wir erneut das Zusammenspiel der verschiedenen Tasks. Wir stellten fest, dass durch die Kombination der von uns gewählten Prioritäten und Delay-Timings keine sinnvolle Abwechslung des Zugriffs auf den Grill durch die Tasks des Grillmeisters und der Physik gegeben war. Entweder kontrollierte der Grillmeister durchgehend, sodass die Physik niemals aktiv werden konnte, oder der Grillmeister bekam nicht genügend Zugriffszeiten, wodurch die Würste auch bei niedriger Temperatur regelmäßig verbrannten. Es waren einige Versuche nötig um den gewünschten Wechsel des Zugriffs der Tasks auf den Grill-Speicher zu realisieren. Wir konnten jedoch bei den von uns im Vorhinein festgelegten Prioritäten bleiben.

Nachdem wir mit diesen Speicherzugriffen durch die zwei beschriebenen Tasks bereits kleinere Schwierigkeiten hatten, erwarteten wir, dass diese sich nun noch ver-

größerten, denn die beiden verbleibenden Tasks (Feuerwehr und Ausgabe) mussten ebenfalls noch auf den Grill-Speicher zugreifen.
Zum Abschluss des zweiten Treffens fertigten wir noch eine rudimentäre textuelle Ausgabe des aktuellen Programmzustands über den VGA-Bildschirm an, um das Testen über diese zu vereinfachen. Die Zugriffe dieses zusätzlichen Tasks auf den Grill verursachten entgegen unserer Erwartungen keine größeren Probleme im Scheduling. Für den Zeitraum bis zum nächsten Treffen verteilten wir die theoretischen Ausarbeitung der grafischen Ausarbeitung und die Überarbeitung des bisher produzierten Codes.

3.2.4 Drittes Treffen - 15.9.

Durch den guten Fortschritt in den ersten beiden Treffen, war für das letzte nur noch die Implementierung des Feuerwehr-Tasks und der grafische Ausgabe geplant, zwei verhältnismäßig kurze Aufgabenteile. Beginnen taten wir jedoch mit etwas ungeplantem.
Durch die Überarbeitung des bisherigen Codes fiel ein Speicherzugriffsfehler auf, der durch eine ungenügende Absicherung des Grill-Speichers theoretisch eintreten könnte. Wir mussten uns nun entscheiden zwischen einer Absicherung durch noch eine zusätzliche Semaphore oder einer Nachrichtenbox, mit der wir uns bisher nur rudimentär auseinander setzten. Wir entschieden uns für die Nachrichtenbox und versuchten uns die korrekte Verwendung aus den gegebenen User-Manuals anzueignen. Hierbei stießen wir nach vielfachem Ausprobieren auf Fehler in den User-Manuals, die uns an diesem letzten Tag viel Zeit kosteten.

Nach der erfolgreichen Umstellung des Kommunikationsdienstes auf die Nachrichtenbox konnten wir dann endlich mit der Implementierung der verbleibenden Tasks fortfahren.
Die Erweiterung der grafischen Ausgabe konnte durch vorgeschriebenen Pseudocode schnell fertiggestellt werden. Durch den geringen Umfang der Feuerwehr-Funktionalität verhielt es sich hiermit sehr ähnlich, sodass wir an diesem dritten Termin noch die Vorführung absolvieren konnte. Durch Versäumnisse bei der Bereichsüberlaufprüfung der Eingabe trat unser Programm bei der ersten Abnahme in einen unerwarteten Fehlerzustand. Da die vergessene Überprüfung eigentlich obligatorisch war, konnte die dieser Fehlers entsprechende unkompliziert behoben

werden, und wir konnten bei der zweiten Vorführung das Programm erfolgreich abnehmen lassen.

Für die Nacharbeit verblieb lediglich noch etwas Arbeit in Sachen Modularisierung des Quellcodes. Zudem hatten wir mit der Dokumentation bis zu diesem Termin noch gewartet hatten, was jedoch zu unserem Ablaufkonzept gehörte.

3.2.5 Nacharbeiten

Durch Nachlässigkeiten bei der Modularisierung und Kommentierung des Quellcodes, mussten wir das fertige Programm in feingranularere Module aufteilen. Dies erforderte einen erneuten Test an einem zusätzlichen Termin im Labor. Durch erneut unerwartet auftretende Komplikationen bei der Modularisierung des Programms verschob sich der geplante Termin für diesen Finalen Test um zwei Tage.

3.2.6 Finaler Test - 21.9.

Nach der zusätzlichen Modularisierung die nicht im Labor umgesetzt wurde, war ein finaler Test auf die vorherige korrekte Funktionalität des Programms notwendig geworden.
Das Programm lief dabei nach einbinden einer fehlenden Bibliothek ohne Probleme.

3.3 Programmtests

Im Folgenden werden grundsätzliche Tests dargestellt, die während des gesamten Entwicklungsverlauf durchgeführt wurden. Diese Tests dienten zusätzlich als Grundlage für Grenzwerttests bzw. Doppelaktivierungstests.

Durch...	getestet nach...	Erwartetes Ergebnis:	tatsächliches Ergebnis:
Eingabe(H)	Grill befüllen	Wurst auf Grill	Wurst auf Grill
Eingabe(D)	Kühlbox befüllen	Würste in Kühlbox	Würste in Kühlbox
Eingabe(F)	F. Wasser reichen	Wurstproduktion stoppt	Wurstproduktion stoppt
Eingabe(G)	G. Wasser reichen	keine Wurstkontrolle	keine Wurstkontrolle
Eingabe(V)	F. Wasser entziehen	keine Wurstproduktion	keine Wurstproduktion
Eingabe(B)	G. Wasser entziehen	Wurstkontrolle startet	Wurstkontrolle startet
Eingabe(+)	Temperatur +	Temperatur steigt	Temperatur steigt
Eingabe(-)	Temperatur -	Temperatur sinkt	Temperatur sinkt
Timer(60s)	Kühlbox befüllen	Würste in Kühlbox	Würste in Kühlbox
Timer(1s)	Wurstkontrolle	Drehen bei +80	Drehen bei +80
Timer(60s)	Feuerwehr	Grill kontrolliert	Grill kontrolliert
Timer(1s)	Physik	Wurst bräunen	Wurst bräunen
Grenzwert(30)	Wurstproduktion	Produktion stoppt	Produktion stoppt

3.4 Aufwandsverteilung

Die gesamte Aufwandsverteilung verhielt sich in etwa wie folgt:

Entwicklungsphase	Zeitaufwand in Stunden
Analyse	3
Entwurf	4
Implementierung	18
Testen	5
Nacharbeiten	5
Dokumentation	11

3.5 Bedienungsanleitung

Die Steuerung der Simulation findet komplett über eine Tastatur statt, die an das DE-Board angeschlossen ist. Die Belegung der Tasten mit den verschiedenen Steuerungsmöglichkeiten sind im folgenden dargestellt:

Tastenbelegung	**Funktion**
d/D	Fleischer zur Produktion veranlassen
h/H	Grillmeister Wurst auflegen lassen
f/F	Fleischer Mineralwasser geben
g/G	Grillmeister Mineralwasser geben
v/V	Fleischer Mineralwasser entziehen
b/B	Grillmeister Mineralwasser entziehen
+	Grilltemperatur erhoehen
-	Grilltemperatur reduzieren

4
Bewertung und Fazit

Wir möchten zum Schluss der Dokumentation noch eine Bewertung der Aufgabe und des Ablaufes ziehen ehe wir mit einem kurzen persönlichem Fazit enden.

4.1 Bewertung

Der Ablauf des Praktikums war im Gegensatz zu den anderen Praktika (Computergrafik, Rechnernetze) terminlich ungewöhnlich gelegt. Wir glauben aber, dass das kein negativer Kritikpunkt ist, denn so kann man sich als Student während des Semesters besser auf andere Übungen konzentrieren, was die Überschneidung von Fristen und den damit oft verbundenen Stress verringert. Natürlich kann es auch so Konflikten mit möglichen Urlaubsplanungen nach den Klausuren kommen, was bei uns in aktuellen Semester aber nicht der Fall war. In diesem Fall hatte man ja auch die Möglichkeit, das Praktikum vor dieser Phase abzuschließen.

Aus mehreren Gründen fanden wir die Aufgabenstellung des Praktikums gut gewählt:
Zum Ersten war die Aufgabe nicht ganz so umfangreich wie von uns zuvor, durch das Rechnernetze Praktikum geprägt, befürchtet wurde. Allerdings hatten wir weitestgehend auch keine Schwierigkeiten, da wir beide zuvor das Praktikum Computergrafiken absolvierten und somit noch gut mit der Sprachen vertraut waren, was die benötigte Zeit wahrscheinlich in Grenzen gehalten hat.
Zudem konnte man sich unter der Aufgabe etwas vorstellen, und alleine schon

durch die Benennung der Funktionen mit Namen wie *placeWurst* war die Grundstimmung während des gesamten Praktikums eine positivere als sie bei einer langweiligeren Aufgabenstellung gewesen wäre.
Letztendlich war die Aufgabe auch deutlich gestellt, und ließ trotzdem an vielen Stellen gewollt Raum für Eigeninterpretationen. Auch waren die User-Manuals, obwohl teilweise fehlerhaft und doch sehr umfangreich, nach einer recht kurzen Eingewöhnung ein sehr gutes Hilfsmittel. Durch diese Kombination kam die Notwendigkeit von Rückfragen meist erst gar nicht auf.

4.2 Fazit

Wir haben gemerkt, dass wir durch die praktische Anwendung der theoretischen Inhalte aus der Vorlesung *Echtzeitsysteme* die Zusammenhänge und tatsächlichen Anwendungsmöglichkeiten der verschiedenen Kommunikationsdienste gut vertiefen konnten. Wir lernten außerdem wie aufwendig und schwierig selbst das Testen von vergleichsweise einfachen Echtzeitsystemen ist. Dadurch schätzt man die Arbeit an harten Echtzeitsystemen in der Wirtschaft umso mehr.
Insgesamt hat das Praktikum uns beiden Spaß gemacht; naja - so viel Spaß so ein Praktikum halt machen kann.