Peketeinräumerding

Michael Thomas Andreas Glatz Simon Weitzel Felix Baral-Weber
Stand: 9. Juli 2016 13:09

Inhaltsverzeichnis

1 Einführung **1**
1.1 Zweck des Dokuments 1

2 Allgemeine Beschreibung **1**
2.1 Aufgabenstellung und Umsetzungsansatz 1
2.2 Entwicklungsumgebung und Programmierung 3

3 Spezifische Anforderungen **4**
3.1 Funktionale Anforderungen 4
3.1.1 DFD1 Simulation 4

1 Einführung

1.1 Zweck des Dokuments

Dieses Dokument beschreibt die Anforderungen und Implementierungsdetails an eine Echtzeitsystemanwendung die das Abschlussprojekt des Moduls Echtzeitbetriebssysteme der EAH Jena. Dieses Dokument richtet sich dabei an den Vorgaben für ein Software Requirements Specification (SRS) nach dem IEEE Standard 830-1998. Der Modulverantwortliche ist Oliver Jack aus dem Fachbereich Elektrotechnik und Informationstechnik.

2 Allgemeine Beschreibung

2.1 Aufgabenstellung und Umsetzungsansatz

Die Aufgabestellung des Semesterprojektes ist es eine Hochregallagersimulation unter zuhilfenahme eines Echtzeitbetriebssystems zu erstellen. Diese soll das gesamte System mit einer statisch gewählten Regaldimension sowohl steuern als auch simulieren. Das Simulationmodell umfasst ein Regallagersystem mit $5 * 10$ Regalfächern, einen Ein- und einen Ausgabeslot und einen Turm zum be- und entladen der Fächer. Dieser Turm ist auf einer Achse montiert auf welcher er sich in X-Richtung bewegen kann, desweiteren kann der Ausleger am Turm in Y-Richtung herauf und herab bewegen. An dem Ausleger ist widerum ein Schlitten befestigt welcher in Z-Richtung rein, also zum Hochregal hin, und raus, zu den Ein-/Ausgabe-Slots gefahren werden kann. Die Position dazwischen wird als neutrale Position bezeichnet. Sämtliche Bewegungen des Turms werden von Tastern auf den Schienen ermittelt um daraus die Position des Turmes bestimmen zu

können. Auf der X-Achse und auf der Y-Achse befinden sich jeweils 10 Taster und auf der Z-Achse 3.
Um ein Klötzchen vom Eingabe-Slot oder aus einem Regalfach aufzunehmen wird der Schlitten am Ausleger unterhalb des Klötzchens aus der neutralen Position der Z-Achse in den Slot bzw. das Fach gefahren und dann erst angehoben. Umgekehrt wird beim Ablegen eines Klötzchens in den Ausgabe-Slot oder in ein Regalfach das Paket von oben auf den Slot bzw. das Regalfach abgesenkt bevor der Ausleger am Schlitten in die neutrale Position der Z-Achse zurückgefahren wird.

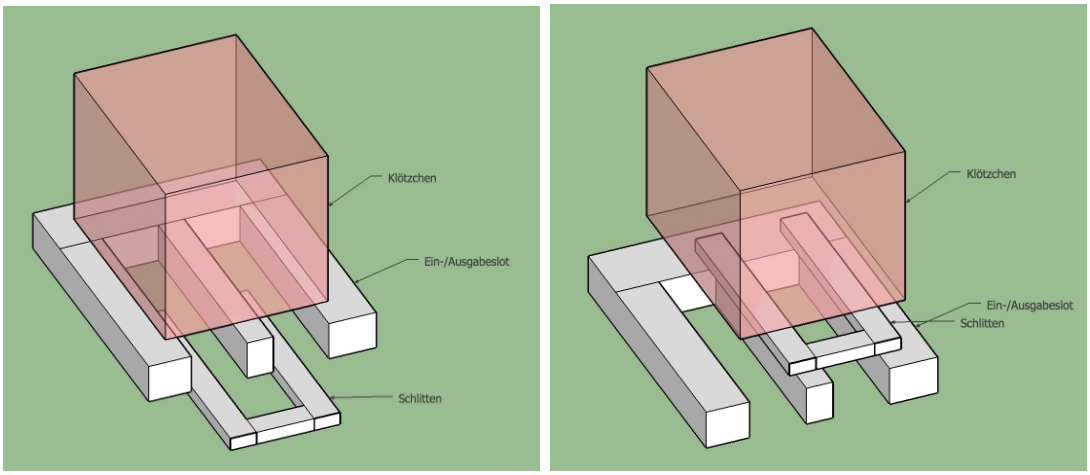

Abbildung 1: Aufnahme/Ablage an Ein-/Ausgabe-Slot

Zur Eingabe steht dem Bediener die Konsole zur Verfügung.

Folgende Befehle werden zur Verfügung gestellt:

- vsetspace[x][y] → Definiert einen Platz als schon belegt
- clearspace[x][y] → Definiert einen Platz als frei
- insert[x][y] → Holt ein Klötzchen vom Eingabe-Slot und legt es an gewünschter Position im Hochregallager ab
- remove[x][y] → Holt ein Klötuchen von der gewünschten Position ab und legt es an den Ausgabe-Slot

Sollte ein Befehl nicht möglich sein, da ein Hochregallagerplatz bereits belegt ist bzw. ein Fach aus dem ein Klötzchen geholt werden soll leer ist, wird der Anwender durch eine in der Konsole ausgegebene Fehlermeldung darauf aufmerksam gemacht und der Befehl nicht ausgeführt. Es können vor Abschluss eines Befehls weitere aufgegeben werden, welche sich in einer Warteschlange einreihen. Die Befehle "clearspace ünd "vsetspace"werden nicht an das Ende dieser gestellt, sondern nach Abschluss des letzten bereits angefangenen Auftrages ausgeführt, da für diese keine Bewegung des Turms nötig ist. Für den Fall, das zu diesem Zeitpunkt eine weitere Operation mit dem selben Regalfach bereits

in der Warteschlange ist, wird diese dann nicht ausgeführt und der Bediener über das Aussetzen dieses Auftrages mittels Konsolenausgabe informiert. Sobald kein weiterer Auftrag in der Warteschlange ist wird der Turm an die Position vor dem Eingabe-Slot gefahren und verweilt dort bis ein neuer Auftrag aufgegeben wird.
Die Visualisierung des Hochregallagers erfolgt ebenfalls in der Konsole, in welcher sowohl das Hochregal und dessen Belegung als auch die Position des Turms und dessen Auslegers, durch ASCI-Zeichen stilisiert dargestellt und nach jeder Änderung aktualisiert werden. Dabei liegt der Ursprung der Koordinaten und somit der Regalplatz (0,0) in der linken unteren Ecke. Die Darstellung des Ein- und Ausfahren des Auslegers wird unterhalb dargestellt. Dabei stellt die linke Position den zum Ein-/Ausgabe-Slot gefahrenen Arm da, die rechte Position die in das Regal hereingefahrene.

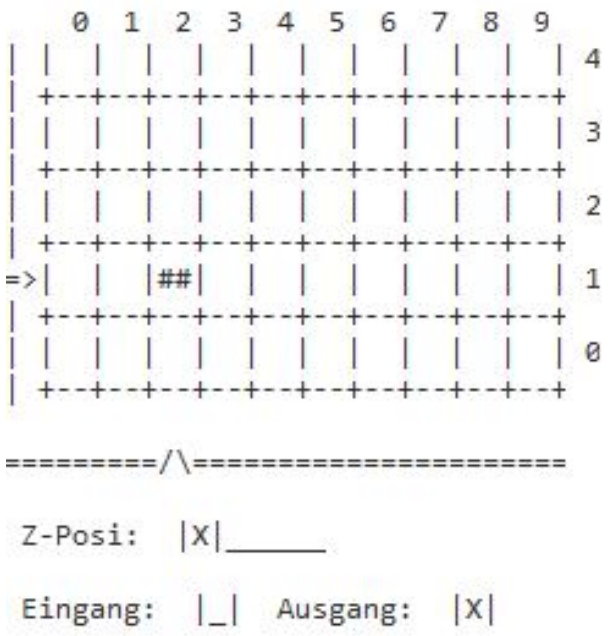

Abbildung 2: Ausgabevisualisierung in Konsole

2.2 Entwicklungsumgebung und Programmierung

Als Entwicklungs- und Testumgebung wurde Windriver Workbench 3.3 genutz und die Programmiersprache C verwendet. Es wurden für das Echtzeitbetriebssystem Vxworks Libaries inkludiert.

3 Spezifische Anforderungen

3.1 Funktionale Anforderungen

3.1.1 DFD1 Simulation

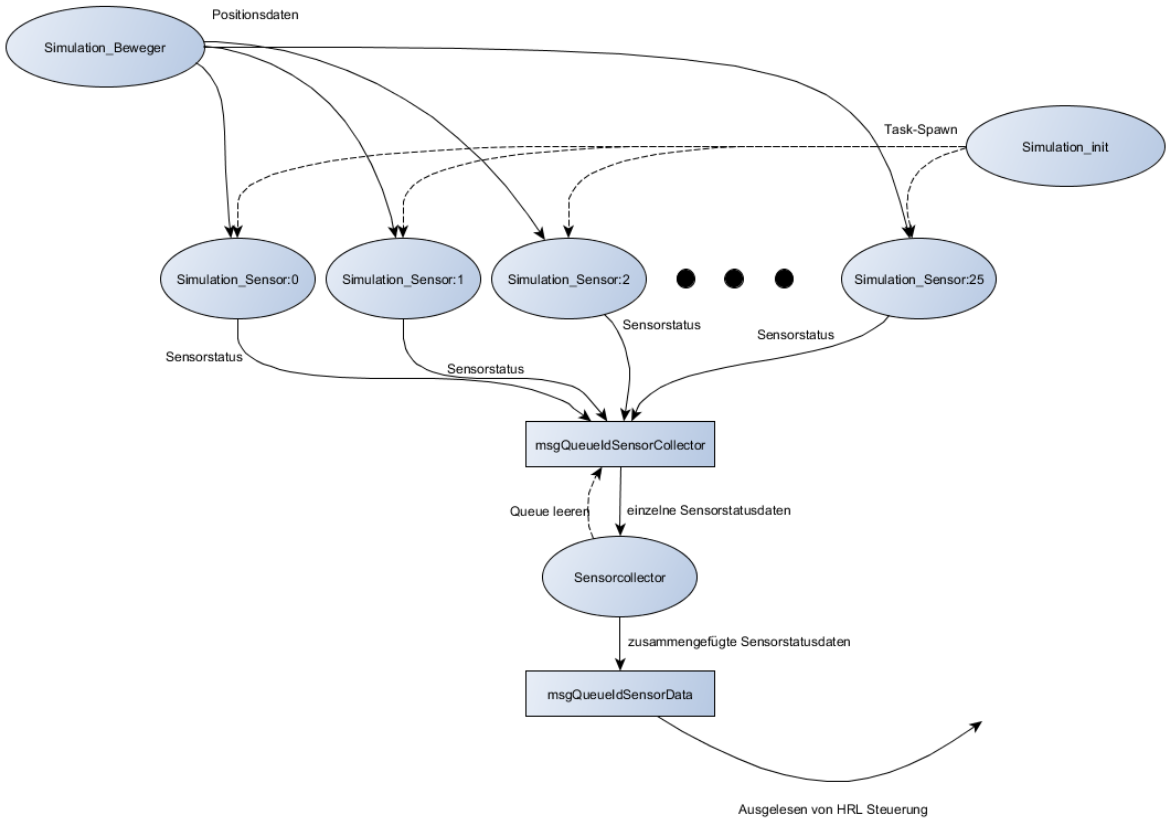

Abbildung 3: DFD1 Simulation

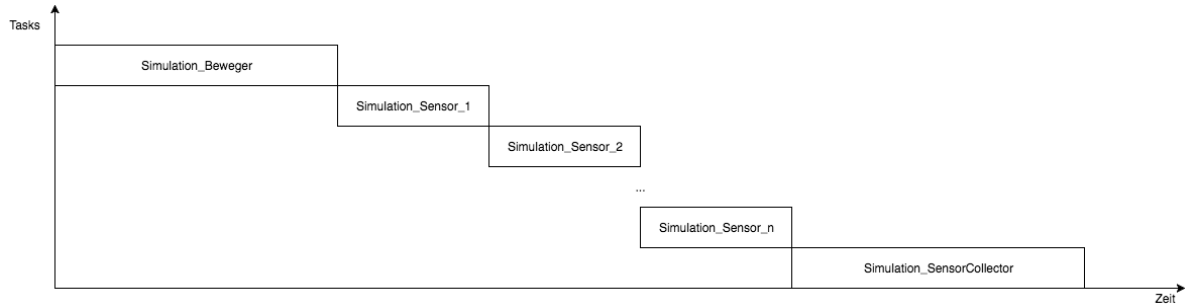

Abbildung 4: Gantt Diagramm der Task der Simulation

Alle Tasks in der Simulation haben eine Priorität von 200 und ändern diese nicht. Sie werden in einer festen Reihenfolge erzeugt und laufen dann in jedem Zyklus in dieser

Sequenz.

Beweger

Wie in Abb.4 zu sehen ist läuft der **Beweger** als erster Task eines Simulationsschrittes. Zuerst aktualisiert er die Aktorwerte aus der gloablen Variable **AktorBusData**. Diese entspricht den Busdaten TODO: entspricht was? und ist durch eine Semaphore geschützt. Da die Simulation minimal blockiert werden soll implementiert die Semaphore eine Prioritätsvererbung. Daraufhin „verschiebt“er den virtuellen Turm entsprechend der Aktordaten.

Sensorcollector

Der **Sensorcollector** läuft nach allen Sensoren und sammelt aus einer Message Queue die Einträge aller Sensoren. Wenn ein Sensor ausfällt bleibt der Wert des Sensors auf 1. Daraufhin schiebt er alle Sensorwerte, in einem Vektor, in eine Messega Queue auf die die Steuerung Zugriff hat.

Simulation Sensor x

Jeder Sensor bekommt eine id übergeben die seine Funktion und ihn eindeutig definiert.

1. **Tastsensoren** überprüfen ob die Turmposition mit der Sensorposition übereinstimmt.

2. **Lichtschranken** Simulieren entweder das eintreffen eines Klötzchens am Eingabeslot oder das Entfernen eines Klötzchens am Ausgabeslot, jeweils mit einem vordefinierten Delay. TODO: Das könnte besser sein

3. Der **Lichttaster** im Turm wird dann ausgelöst wenn die vorige y-Position des Turms im vorigen Schritt über (beim Ablegen eines Klötzchens) bzw. unter (beim Aufnehmen) der jetzigen Position ist und (beim Ablegen) ein Klötzchen im Turm ist bzw. (beim Aufnehmen) ein Klötzchen an der Aufnahmeposition ist.

Die jeweiligen Sensordaten schiebt der Task daraufhin in eine Message Queue für den Sensorcollector.

Taster X[0-9]	[gedrückt|gehalten]
Taster Y[0-9]	[gedrückt|gehalten]
Taster Z[0-3]	[gedrückt|gehalten]
Lichtschranke Eingabe	[unterbrochen|nicht unterbrochen]
Lichtschranke Ausgabe	[unterbrochen|nicht unterbrochen]
Turmlichtschalter	[bedeckt|nicht bedeckt]

Tabelle 1: Requirements Dictionary: Die Stati der Sensoren