CT Praktikum: Finite State Machine (Liftsteuerung)

1 Funktion

In diesem Praktikum implementieren Sie eine Finite State Machine (FSM) als Steuerung für eine einfache Liftanlage.

Das Modell in **Abbildung 1** zeigt den CT-Board-Warenlift, der den Keller eines Restaurants (untere LED-Reihe) mit der Küche (obere LED-Reihe) verbindet. Die Taster T0 und T1 dienen als Lift-Anforderungstasten, die Schalter S7 und S23 als Türöffnungsvorrichtungen auf den beiden Stockwerken.

Als erstes kontrollieren Sie das Öffnen und Schliessen der Lifttüren und implementieren die Fahrt zwischen den beiden Stockwerken.

Erweitern Sie dann das Basismodell um Sicherheitspausen vor der Fahrt, eine Signalsteuerung im oberen Stock sowie eine Gewichtskontrolle für die Beladung im Keller.

2 Lernziele

- Sie können eine Funktionsbeschreibung in ein einfaches UML-State-Diagramm umsetzen.
- Sie können ein UML-State-Diagramm in C implementieren.

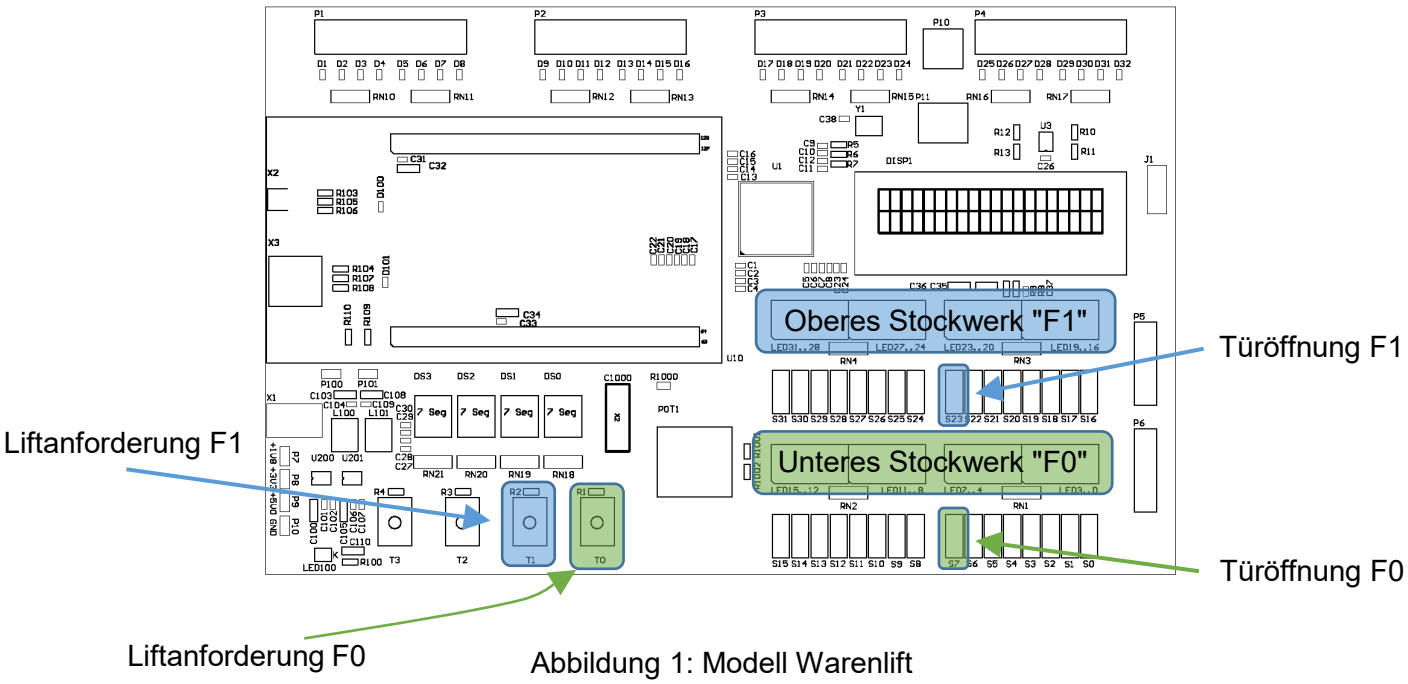

Abbildung 1: Modell Warenlift

3 Aufgaben

Die Funktionen der Liftsteuerung werden schrittweise umgesetzt.

3.1 Türkontrolle

In einem ersten Schritt soll eine einfache FSM gemäss **Abbildung 2** für den Türöffnungsmechanismus im Keller ("F0") implementiert werden:

- Studieren Sie die Funktion `eh_get_event()` im Modul `event_handler`. Wie werden die beiden Events `EV_DOOR0_CLOSED` und `EV_DOOR0_OPENED` generiert?
- Implementieren Sie die Funktionen `fsm_init()` und `fsm_handle_event()` im Modul `state_machine` gemäss dem Diagramm in Abbildung 2.

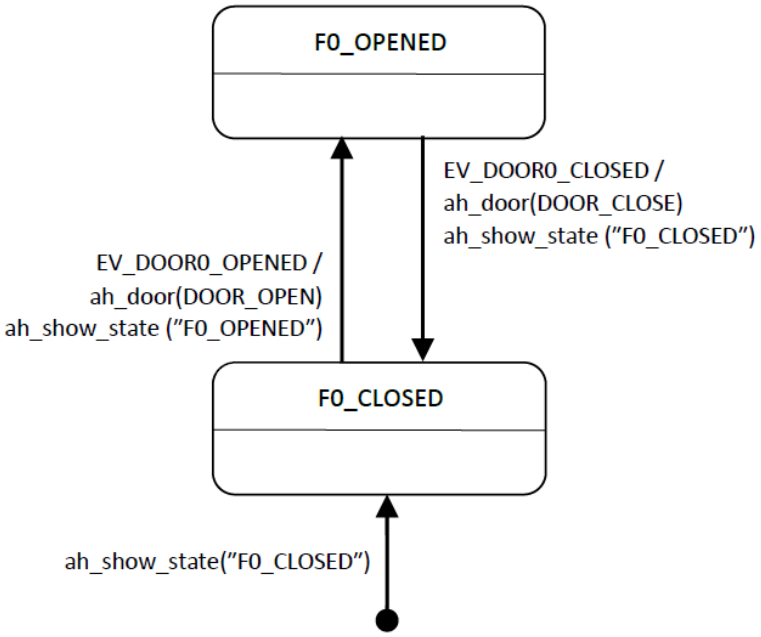

Abbildung 2: UML-State-Diagramm 'Türsteuerung'

Hinweise: Das Modul `action_handler` enthält eine Funktion `ah_show_state(char text[])`. Mit dieser Funktion können Sie Strings für das Debugging auf den LCD ausgeben. Zeigen Sie damit bei einer Transition den Folgezustand an, zum Beispiel: `ah_show_state(„F0_CLOSED");`

Die Aufrufe `ah_door(DOOR_OPEN)` und `ah_door(DOOR_CLOSE)` dienen der Animation des Warenlifts.

3.2 Liftfahrt

Im zweiten Schritt wird die Liftfahrt zwischen Keller und Küche ("F0") gemäss **Abbildung 3** implementiert, ebenso der Türöffnungsmechanismus im oberen Stock.

- Sehen Sie in der Funktion `eh_get_event()` nach, wie die Events `EV_BUTTON_F0` und `EV_BUTTON_F1` generiert werden. Sie dienen dazu, die Fahrt auszulösen, **falls die Türen geschlossen sind**.
- Die Events `EV_F0_REACHED` und `EV_F1_REACHED` signalisieren, wann der untere resp. der obere Stock erreicht sind **und der Liftmotor abgeschaltet werden muss**. Die Events werden automatisch von der Lift-Animation im Rahmenprojekt erzeugt.
- Erweitern Sie die State Machine so, dass die zusätzlichen States und Transitionen unterstützt werden.
- Funktioniert Ihr Lift ohne Crash?

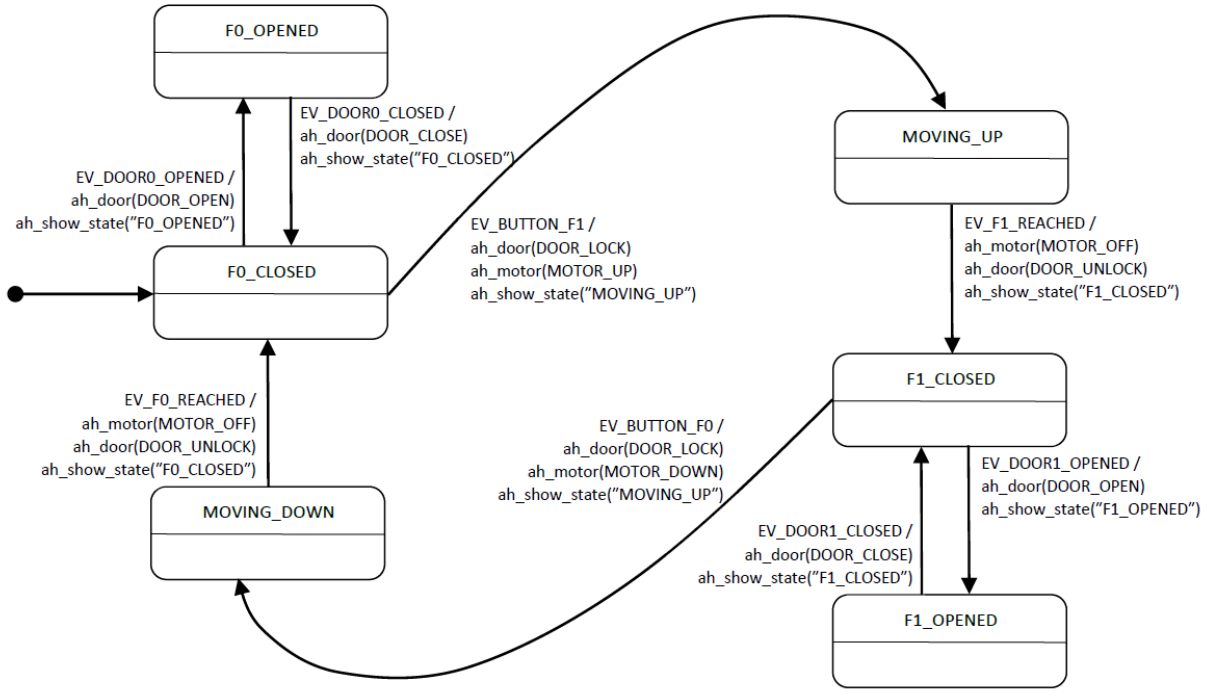

Abbildung 3: UML-State-Diagramm 'Liftfahrt'

3.3 Komfortfunktionen

Erweitern Sie nun das Liftmodell um **mindestens eine** der folgenden 3 Funktionen:

a) Kurze Sicherheitspause vor der Fahrt

Bevor der Lift nach oben oder nach unten losfährt, soll eine Sicherheitspause von 1.5s eingelegt werden, während der die Türen zwar verriegelt sind (`DOOR_LOCK`), aber der Lift noch stillsteht.

Benutzen Sie das vorgegebene Modul `timer`, um die notwendigen Verzögerungen zu implementieren; konsultieren Sie wiederum `eh_get_event()`, um herauszufinden, wie das zugehörige Event heisst, das Sie empfangen, wenn die Zeit abgelaufen ist.

Erweitern Sie ihr Programm um die fehlenden Funktionen:

- Erweitern Sie das UML-State-Diagramm und tragen Sie die neuen Übergänge ein.
- Den Timer können Sie mittels der Funktion `timer_start(duration)` starten, und mittels `timer_stop()` stoppen, falls der Timer abgebrochen werden soll.
  Der Funktionsparameter `duration` bezieht sich auf die Anzahl Perioden eines 100 Hz Signals (10ms-Schritte).

b) Signalisation, wenn der Lift im oberen Stock ankommt

Um das Küchenpersonal darauf aufmerksam zu machen, dass eine Lieferung aus dem Keller bereit steht, soll ein Blinklicht implementiert werden, das losgeht, wenn der Lift den oberen Stock erreicht, und stoppt, wenn die Lifttüre geöffnet wird.

Erweitern Sie ihr Programm folgendermassen:

- Erweitern Sie das UML-State-Diagramm und tragen Sie die neuen Übergänge ein.
- Auch hier wird ein Timer benötigt; beachten Sie dazu die Angaben oben unter a).
- Um das Blinklicht an- und abzuschalten, verwenden Sie die Aufrufe `ah_signal(SIGNAL_ON)` und `ah_signal(SIGNAL_OFF)`.

c) Gewichtskontrolle beim Beladen im unteren Stock

Es ist nun so, dass der Warenlift aus Sicherheitsgründen nicht mehr als 50 Kg transportieren darf. Wenn also der Lift **im Keller ist und die Lifttüre offensteht**, soll das Gewicht der Ladung angezeigt und überprüft werden; bei zu hohem Gewicht darf nicht losgefahren werden.

Als Gewichtseingabe wird das CT-Board Potentiometer verwendet, das so konfiguriert ist, dass Werte zwischen 0 und 63 erzeugt werden. Das aktuell eingestellte Gewicht sehen Sie auf der 7-Segment-Anzeige.

Erweitern Sie ihr Programm wie folgt:

- Erweitern Sie das UML-State-Diagramm und tragen Sie die neuen Übergänge ein.
- Verwenden Sie `eh_weight_control(WCTL_ENABLE, 50)`, um die Waage einzuschalten und die zugehörigen Events `EV_WEIGHT_TOO_HIGH` sowie `EV_WEIGHT_OK` zu empfangen.
  `eh_weight_control(WCTL_DISABLE, 0)`, schaltet die Waage wieder aus.
- Um eine Warnung anzuzeigen, verwenden Sie `ah_show_exception(WARNING, "Gewicht zu hoch!")`, und um die Warnung zu löschen `ah_show_exception(NORMAL, "")`.

4 Bewertung

Die lauffähigen Programme müssen präsentiert werden. Die einzelnen Studierenden müssen die Lösungen und den Quellcode verstanden haben und erklären können.

Bewertungskriterien	**Gewichtung**
Die Türkontrolle aus Aufgabe 3.1 ist funktionsfähig.	1/4
Der Lift fährt fehlerfrei gemäss Aufgabe 3.2.	1/4
Sie haben mindestens eine Komfortfunktion aus Aufgabe 3.3 implementiert.	2/4