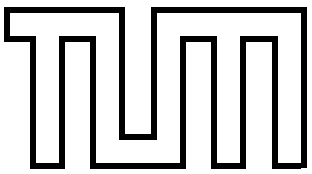


Technische Universität
München

Fakultät für Informatik
Forschungs- und Lehreinheit Informatik VI

Übung zur Vorlesung Echtzeitsysteme

Aufgabe 3 – EasyLab

Dr. Christian Buckl
buckl@in.tum.de

Simon Barner
barner@in.tum.de

Michael Geisinger
geisinge@in.tum.de

Stephan Sommer
sommerst@in.tum.de

Wintersemester 2008/09

Aufgabe 3: Modellbasierte Entwicklung mit EasyLab

Ziel

In der Vorlesung wurde kurz das modellbasierte Entwicklungswerkzeug EasyLab vorgestellt, mit dessen Hilfe ein pneumatischer Zylinder programmiert wurde. Heute erhalten Sie selbst die Möglichkeit, Programme in EasyLab zu entwickeln und nutzen dabei unter anderem den *synchronen Datenfluss* als „Modellierungssprache“, den Sie auch bereits aus der Vorlesung kennen. Das modellierte Programm wird einen mobilen Roboter steuern, dessen Bewegungen wir der Einfachheit halber nur in einer 3D-Szene simulieren werden.

EasyLab-Kurzeinführung

EasyLab ist ein modellbasiertes Entwicklungswerkzeug, das zurzeit am Lehrstuhl Informatik VI entwickelt wird. Es ist unter anderem für die Modellierung von Programmen für mechatronische Systeme und Roboter geeignet. Das Programm ist auf allen Rechnern im Praktikumsraum vorinstalliert und kann unter *Start* → *Programs* → *EasyLab* gestartet werden.

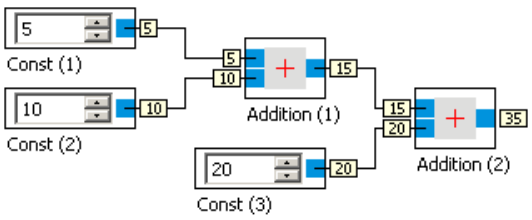
EasyLab unterstützt den folgenden Workflow:

1. **Hardware-Modell**:
   Auf welchem Gerät kommt das Programm später zur Ausführung?
2. **Software-Modell**:
   Was soll das Programm bei Ausführung tun?
3. **Simulation** und/oder **Codegenerierung**:
   Ausführung des Programms in EasyLab oder auf dem Zielsystem.

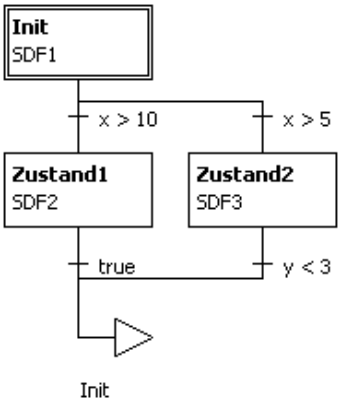
Heute werden wir uns auf die Entwicklung des *Software-Modells* sowie auf die *Simulation* der modellierten Anwendung beschränken. Das für die mobile Roboterplattform benötigte *Hardware-Modell* liegt bereits vor.

EasyLab unterstützt die folgenden zwei Modellierungssprachen:

1. **Synchroner Datenfluss**
   Der synchrone Datenfluss (*SDF*) ist bereits aus der Vorlesung bekannt. Programme bestehen aus Funktionsblöcken (Aktoren), die Ein- und Ausgangsports besitzen. Über Verbindungen können zwischen den Aktoren Daten ausgetauscht werden. SDF-Programme werden zyklisch ausgeführt.

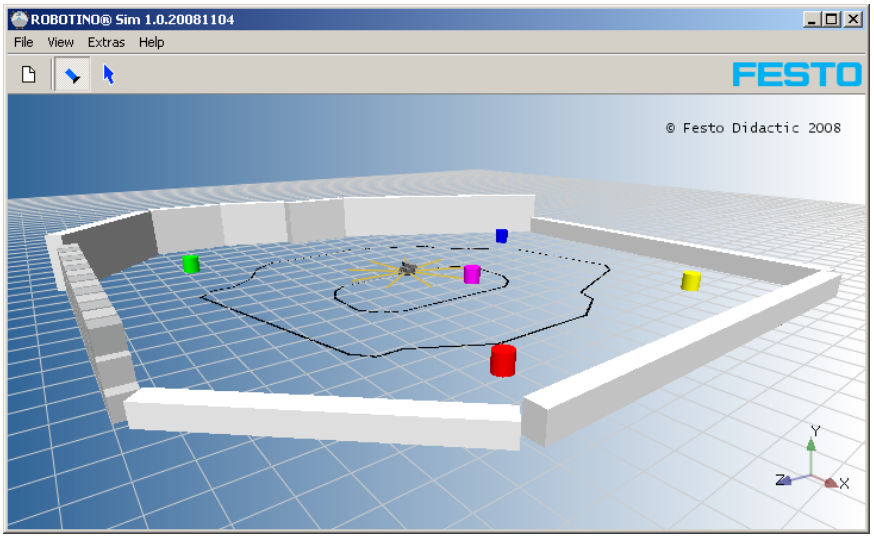


2. **Zustandsfluss-Diagramm**
   Das Zustandsfluss-Diagramm (englisch *sequential flow chart*, SFC) ist ähnlich wie ein endlicher Automat aufgebaut. Es werden damit Zustände modelliert, in denen sich die Anwendung befinden kann. Jedem Zustand ist ein SDF-Programm zugeordnet, das ausgeführt wird, so lange der jeweilige Zustand aktiv ist. Der Initialzustand ist durch eine doppelte Umrandung gekennzeichnet. Um zwischen verschiedenen Zuständen umzuschalten, werden so genannte *Transitionsbedingungen* angegeben. Transitionsbedingungen können sich auf Variablen beziehen, die im SDF direkt *vor* dem jeweiligen Übergang definiert werden (z.B. $x > 10$). In EasyLab definiert man Variablen, indem man einen *Global Variable Writer*-Aktor verwendet und diesen entsprechend der Variable benennt. Weitere SFC-Elemente sind alternative Verzweigungen (auf einen Zustand können mehrere Zustände folgen) und Sprünge (die mit dem Namen des Zielzustandes versehen sind).



RobotinoSim-Kurzeinführung

Die mobile Roboterplattform Robotino besitzt 3 Motoren für den Antrieb, eine Kamera und 9 Infrarot-Abstandssensoren. Die Anwendung RobotinoSim simuliert die Bewegungen des Roboters in einer 3D-Szene und liefert ein „simuliertes“ Kamerabild und die (virtuellen) Messwerte der Sensoren an EasyLab zurück. Somit ist es möglich, ein in EasyLab modelliertes Programm zu testen, ohne wirklich einen echten Roboter an das System angeschlossen zu haben.



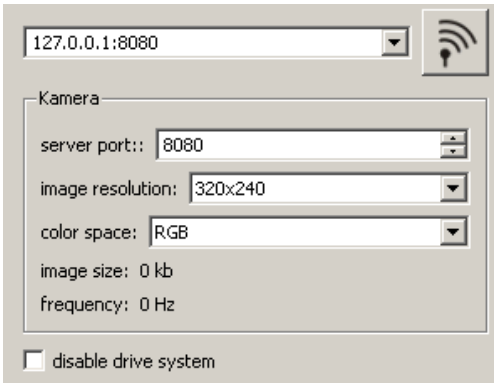
Aufgabe

Mit einem in EasyLab entwickelten Programm soll die mobile Roboterplattform Robotino gesteuert werden. Unter `\\atknoll133\Uebung` finden Sie die Datei `robotino.easy`, die bereits das passende Hardware-Modell enthält. Kopieren Sie die Datei zunächst auf Ihr Netzlaufwerk und öffnen Sie diese dann in EasyLab.

a) Starten Sie das Programm RobotinoSim, das Sie unter *Start* → *Programs* → *RobotinoSim* finden und betrachten Sie die abgebildete Szene. Was ist die Bedeutung des von *Subtraction1* in `robotino.easy` berechneten Wertes? Auf der nächsten Seite finden Sie Kurzreferenz zu den Aktoren.

b) Welche Aktionen wird der Roboter ausführen, wenn das Programm gestartet wird?

c) Überprüfen Sie Ihre Vermutung, indem Sie eine Simulation des Programms durchführen. Öffnen Sie dazu in EasyLab mittels des Menübefehls *View* → *Show Device Manager* den Gerätemanager und stellen Sie sicher, dass das Gerät *Robotino* wie folgt konfiguriert ist. Danach klicken Sie auf den Knopf mit der Antenne.



Nach erfolgreicher Verbindung (grünes Antennensymbol) starten Sie die Simulation in EasyLab und beobachten die Aktionen des Roboters in RobotinoSim.

d) Durch die vorherigen Aufgaben haben Sie einen Eindruck davon gewonnen, wie Datenflussprogramme in EasyLab programmiert werden. Erstellen Sie nun ein Programm bei dem der Roboter so lange in eine Richtung fährt, bis er sich einem Objekt nähert. Ist dies der Fall, so soll sich der Roboter ein Stück um sich selbst drehen und weiterfahren, bis er auf die nächste Begrenzung stößt. Ihr Programm sollte so robust sein, dass der Roboter nie ein Objekt berührt.

Um diese Aufgabe zu lösen, verwenden Sie am besten ein SFC als Hauptprogramm, das aus mehreren Zuständen besteht. Überlegen Sie sich zunächst, welche Zustände das sein könnten und entwerfen Sie dann für die Zustände je ein Datenflussprogramm. Schließlich geben Sie Bedingungen an, durch die die Zustandsübergänge gesteuert werden.

Neue Aktoren können Sie in ein SDF-Programm per Drag & Drop aus dem Fenster *Devices and Actor Variants* in das SDF ziehen. Nicht mehr benötigte Aktoren und Verbindungen können durch Markieren und Drücken der *Entf*-Taste gelöscht werden.

Simulieren Sie Ihr Programm, um zu überprüfen, ob das Ergebnis dem erwarteten Verhalten entspricht. Sollte Robotino in der Simulation die Begrenzung durchbrechen, können Sie die Szene in RobotinoSim mittels *File* → *Reset scene* zurücksetzen. Danach müssen Sie sich von EasyLab aus neu verbinden.

Tipp: Wählen die in RobotinoSim unter *Extras* → *Options...* unter *Debug* → *Sensor* die Option *Show distance sensors*, um sich die simulierten Abtastbereiche der Abstandssensoren anzeigen zu lassen.

Kurzreferenz zu den Aktoren

Robotino → Antriebssystem → Motor Ansteuerung der Motoren der mobilen Roboterplattform (1. Eingang).

Robotino → Antriebssystem → Omnidrive Umrechnung von Geschwindigkeit in X- und Y-Richtung (1. und 2. Eingang) sowie Winkelgeschwindigkeit (3. Eingang) in Steuersignale für die drei Motoren (Ausgang 1 bis 3).

Robotino → Kollisionserkennung → Abstand Abstandssensoren. Nummer 1 zeigt frontal nach vorne (in Richtung der Kamera), die restlichen Sensoren sind gleichmäßig gegen den Uhrzeigersinn verteilt (Nummer 2 – 9).

Robotino → Image System → Kamera Liefert das Bild der Kamera am Roboter.

Generic → Bildverarbeitung → Bildinformation Liefert Breite und Höhe des Bildes.

Generic → Bildverarbeitung → Linienerkenner Liefert X-Position einer Linie, die im Y-Wertebereich *searchstart* bis *searchstart* + *searchheight* im Bild gesucht wird.

Generic → Globale Variablen → Global Variable Reader Gibt den Wert einer (globalen) Variable zurück. Der Aktor muss so benannt werden, wie die Variable heißt.

Generic → Globale Variablen → Global Variable Writer Setzt den Wert einer (globalen) Variable. Der Aktor muss so benannt werden, wie die Variable heißt.