Zimmermann, Frank

Working Paper

Benutzerhandbuch leJOS Statemachine Toolkit

Arbeitspapiere der Nordakademie, No. 2008-02

Provided in Cooperation with:
Nordakademie - Hochschule der Wirtschaft, Elmshorn

Suggested Citation: Zimmermann, Frank (2008) : Benutzerhandbuch leJOS Statemachine Toolkit, Arbeitspapiere der Nordakademie, No. 2008-02, Nordakademie - Hochschule der Wirtschaft, Elmshorn

This Version is available at:
https://hdl.handle.net/10419/38585

ARBEITSPAPIERE DER NORDAKADEMIE

ISSN 1860-0360

Nr. 2008-02

Benutzerhandbuch leJOS Statemachine Toolkit

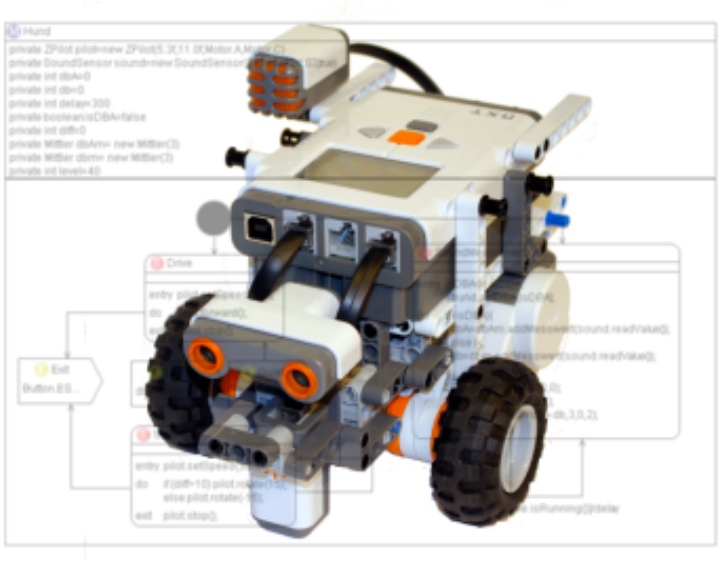
Prof. Dr. Frank Zimmermann

Februar 2008

Eine elektronische Version dieses Arbeitspapiers ist verfügbar unter:
http://www.nordakademie.de/index.php?id=ap

NORDAKADEMIE
HOCHSCHULE DER WIRTSCHAFT

Köllner Chaussee 11
25337 Elmshorn
http://www.nordakademie.de



leJOS
Statemachine Development Toolkit

[1]

Modellierung von Zustandsautomaten für Lego Mindstorm NXT

Frank Zimmermann
ft.zimmermann@web.de
FH Nordakademie

26. Februar 2008

[1]Titelfoto: Werner Welzin

Inhaltsverzeichnis

Kapitel 1

Installation

"Murphy, O'Malley, Sod and Parkinson are alive and well - and working on your project. "

—ohne Autor

1.1 Installationsvoraussetzungen

Bevor mit Eclipse Programme erstellt werden können, muss Lejos auf dem NXT ge"flash"t worden sein. Das Plugin beinhaltet derzeit noch keine Funktionalität, um leJOS auf den Brick herunterzuladen.

Die Lejos Plugins basieren auf Eclipse Europa. Dies ist auf der Eclipse Homepage[1] frei herunterladbar.

Zur Zeit ist das Plugin unter Win32 Plattformen mit Bluetooth oder USB Anbindung, unter Mac OSX mit Bluetooth lauffähig. Es wird die BlueCove Java Schnittstelle für Bluetoothzugriffe verwendet. Das bedeutet, dass der auf dem PC installierte BluetoothStack von BlueCove unterstützt werden muss. Mit Windows XP SP2 bietet Microsoft einen Bluetoothstack an, der funktionieren sollte. Abbildung 1.1 zeigt meine Installation.

[1] http://www.eclipse.org

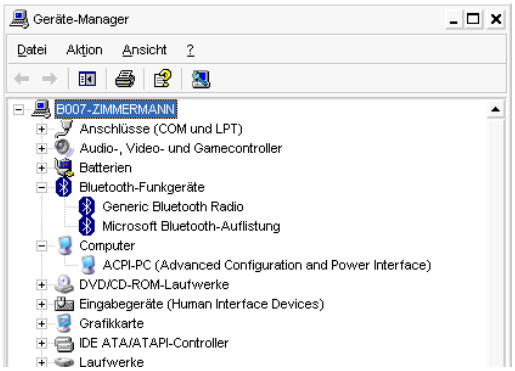


Abbildung 1.1: Beispiel für installierten Bluetoothstack

1.2 Plugin installieren

Das Plugin nutzt GMF, deshalb müssen die zugehörigen Features vorher installiert werden: Zur Installation nutzen Sie **HELP≻SOFTWARE UPDATE≻FIND AN INSTALL ...≻SEARCH FOR NEW FEATURES TO INSTALL** Wählen Sie den Europa Discovery Site aus und drücken Sie auf Finish. Wählen Sie die Ihnen angenehmsten Server aus. Klappen Sie die Europa Discovery Site „Models and Model Development“ und wählen Sie das Graphics Modeling Framework aus. Wählen Sie mit SelectRequired auch die erforderlichen Plugins aus dem Europa Discovery Site aus. Installieren Sie diese Plugins.

Zusätzlich zu den GMF Plugins wird nun auch oAW benötigt.Nutzen Sie den Updatesite

http://www.openarchitectureware.org/updatesite/milestone/site.xml

um das oAW classic feature zu installieren. Tipp: Klappen Sie in der Produktauswahl oAW Classic auf, dann können Sie das oAW Classic Feature allein auswählen. Es benötigt zwei weitere features von dieser updatesite.

Installieren Sie das leJOS Plugin von:

http://fermat.nordakademie.de/update/site.xml

Kapitel 2

Zustandsautomaten

"Vertrauen ist gut - Kontrolle besser."

—Lenin, fälschlich zugeschrieben

Zum Lösen einer komplexen Kontrollaufgabe kann es erforderlich sein, diese Aufgabe so zu strukturieren, dass sie überschaubar bleibt. Ein Roboter, der eine komplizierte Bewegung durchführt, muss aber jederzeit unterbrochen werden können, weil er gegen ein Hindernis stößt oder weil ein Sensor einen Abgrund gefunden hat. Wenn der Programmierer sich um alle möglichen Ereignisse selbst kümmern muss, dann wird das Programm schnell unübersichtlich. Es wird eine Instanz benötigt, die eine übergeordnete Kontrolle ausübt und im Bedarfsfall eine Unterbrechung der Arbeit vornimmt. Das Zusammenspiel zwischen übergeordneter Instanz und eigentlicher Arbeit muss geregelt sein. Dazu ist ein Zustandsautomat (Statemachine) ein geeignetes Mittel.

2.1 Das Metamodell

In einem Metamodell ist der grundlegende Aufbau des Anwendungsbereichs beschrieben. Es enthält jene Modellelemente, deren Abstraktion sinnvoll zur Vereinfachung der Anwendung beiträgt. Im Falle einer Steuerung eines NXT Robots ist die größte Schwierigkeit darin zu sehen, verschiedene quasiparallel ablaufende Anwendungen miteinander zu koordinieren. Es sollte also Aufgabe des Metamodells sein, diese Komplexität vor dem eigentlichen Programmierer so weit wie möglich zu verbergen. Es ist die Aufgabe der technischen Experten, sich um die Umsetzung der Synchronisationsaufgaben zu kümmern.

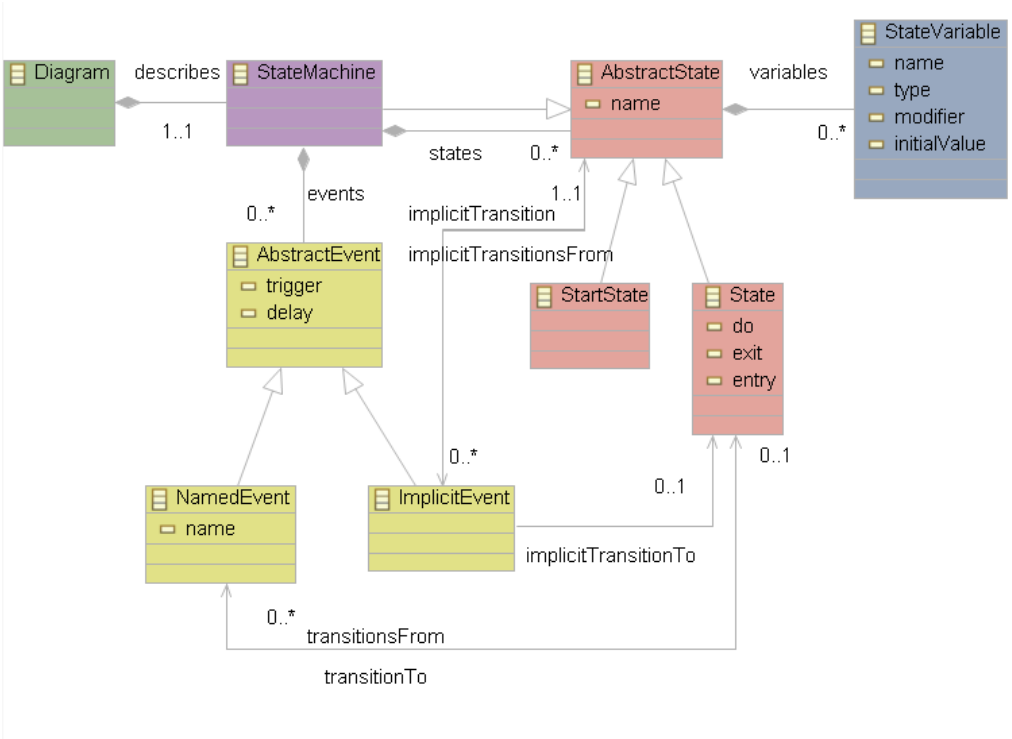


Abbildung 2.1: Metamodell des Zustandsautomaten

2.2 Zustandsautomaten

Zustandsautomaten werden in der Informatik häufig zur Beschreibung von Abläufen verwendet. Zustandsautomaten eignen sich deshalb besonders zu diesem Zweck, weil sie besonders gut in Zustandsdiagrammen dargestellt werden können. Sie enthalten einerseits Zustände, in denen sich das System befinden kann und andererseits Zustandsübergänge, die durch definierte Ereignisse ausgelöst werden. Durch diese Diagramme ist es möglich, Abläufe zu visualisieren und damit übersichtlicher und leichter verständlich zu machen. Eine reduzierte Fehlerrate ist die Folge.

Neben der genannten Anwendung finden Zustandsautomaten noch eine weite Verbreitung in der Beschreibung von Protokollen und in der theoretischen Informatik als endliche Automaten.

Abbildung 2.1 zeigt das Metamodell des Zustandsautomaten. Das Wurzelelement ist „Diagram“. Ein Diagramm enthält in der momentanen Version genau einen Zustandsautomat. Eine Version des Plugins mit mehreren „Zustandsau-

tomaten“ ist sicher eine sinnvolle Erweiterung. Ein Zustandsautomat (Statemachine) hat im Kern zwei Aufgaben:

1. Ausführen der Aktionen der Zustände

2. Wechseln zwischen den Zuständen

In den generierten Anwendungen werden diese Aufgaben von zwei unterschiedlichen Threads erledigt. Das Programm braucht sich jedoch um die Synchronisation dieser beiden Threads nicht zu kümmern.

In dem Plugin kann man einen Zustandsautomaten über das Kontextmenü des modelle- Verzeichnisses des leJOS Projekts mit **NEW≻NEW≻OTHER≻LEJOS STATEMACHINE DIAGRAMS≻STATEMACHINE DIAGRAM** hinzufügen. Das Diagramm zeigt zunächst einen Fehler, da erst ein Zustandsautomat im Diagramm angelegt werden muss. Da die graphische Modellierung typischerweise viel Platz benötigt, empfiehlt es sich, das Editorfenster zu maximieren und den Problems View an das Editorfenster anzudocken. Die Palette des graphischen Editors auf Abbildung 2.2 bietet Werkzeuge zum Hinzufügen

- eines Zustandsautomaten zum Diagramm

- von Zustandsvariablen in den Variablenbereich der Zustände und des Zustandsautomaten

- von Zuständen, Ereignissen, einem Startzustand und Kontrollflüssen zwischen Ereignissen und Zuständen im Modellierungsbereich des Zustandsautomaten.

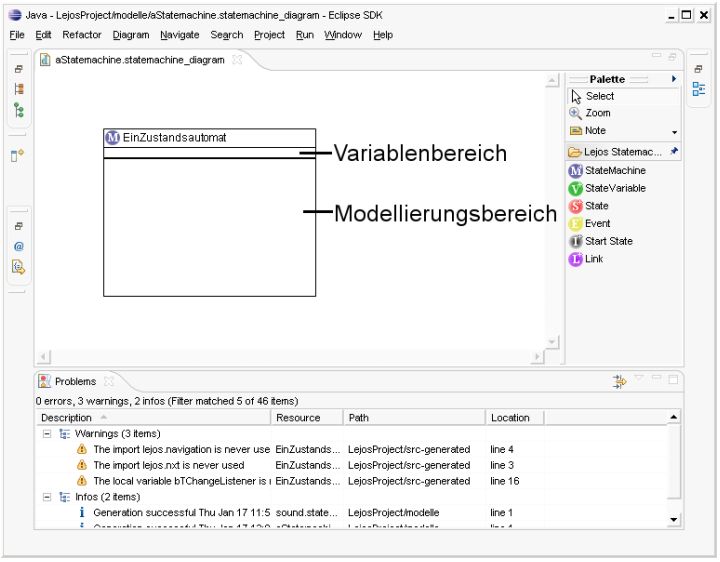


Abbildung 2.2: Zustandsautomat im Editorfenster

2.2.1 Zustände

Begriff 2.2.1
Zustand

ZUSTAND:
Ein Zustand (State) modelliert eine besondere Situation, in der sich ein Objekt befindet. Diese Situation wird durch die Aufgabenstellung beschrieben. Beispielsweise gibt es bei einem Robot, der auf einer schwarzen Linie fahren soll, zwei sofort in die Augen fallende Zustände: „Auf der Linie“ und „Neben der Linie“.
Zustände sind zunächst statisch, das bedeutet, dass sie eine Situation beschreiben, nicht ein Verhalten. Dem Zustand können aber drei Verhaltensspezifikationen, zum Beispiel in Form von Java Programmanweisungen, zugeordnet werden:

1. ein Eintrittsverhalten, das ausgeführt wird, wenn der Zustandsautomat in den Zustand eintritt (engl. entry behaviour)
2. ein Austrittsverhalten, das ausgeführt wird, wenn der Zustandsautomat den Zustand verlässt (engl. exit behaviour)
3. ein Verlaufsverhalten, das ausgeführt wird, während sich der Zustandsautomat im Zustand befindet (engl. do activity)

Alle drei Verhaltensspezifikationen werden gemeinsam als das Verhalten des Zustands bezeichnet.

Diese Definition eines Zustands orientiert sich an den Vorstellungen von Zuständen, die sich in der standardisierten Modellierungssprache UML[1] finden.

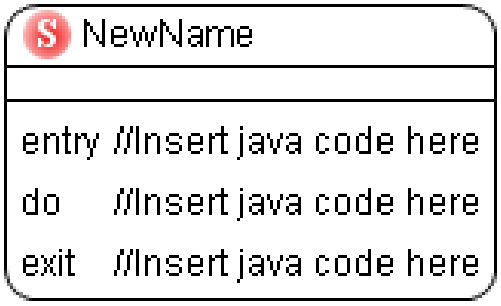


Abbildung 2.3: Zustand mit Verhaltensspezifikation

Da Zustände statische Situationen beschreiben, sollte ihr Name eigentlich eine Nominalphrase sein, die diese Situation kurz und präzise beschreibt. „AufDerLinie“ ist ein gutes Beispiel. Häufig steht jedoch das do-Verhalten im Vordergrund, so dass aus der Nominalphrase auch schon mal eine Verbalphrase „FahreAufLinie“ wird.

Graphisch wird ein Zustand meistens als Rechteck mit abgerundeten Ecken dargestellt, leicht unterschiedliche Darstellungsformen sind aber auch möglich, siehe Beispiele in der Abbildung rechts.

Zur Veranschaulichung ein Beispiel. Für die ersten Zustandsautomaten benötigen wir einen Zustand mit dem Namen „Welcome“. Als Verlaufsverhalten wird

```
1 LCD.drawString("Hallo Welt");
2 LCD.refresh();
```

verwendet. Das Verlaufsverhalten muss eine vollständige Java Anweisung sein.

Damit eine Statemachine ausgeführt werden kann, muss festgelegt werden, welcher Zustand zuerst ausgeführt werden soll. Dazu benutzen wir einen Startzustand, der in der UML ein Pseudozustand genannt wird. Pseudo- deshalb, weil dieser Zustand von der Statemachine eigentlich niemals eingenommen wird. Er dient nur zur Markierung des Anfangs der Verarbeitung. Der Startzustand wird dann mit einem Pfeil mit dem ersten Zustand verbunden.

Der Zustandsautomat startet in dem Welcome Zustand und führt das Verlaufsverhalten aus. Das ist in diesem Fall die berühmte „Hello, world!“ Ausgabe. Ist

[1] http://de.wikipedia.org/wiki/Zustandsautomat_(UML)

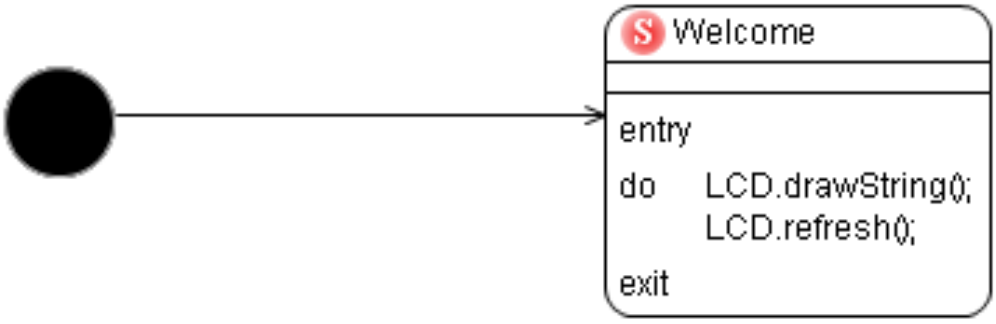


Abbildung 2.4: Einfacher Automat mit einem Zustand.

diese Ausgabe beendet, so soll ein Folgezustand gesucht werden. Es ist aber keiner definiert, und deshalb bricht der Zustandsautomat an dieser Stelle ab und beendet sich. Möchte man die Verhalten anderer Zustände ausführen, so benötigt man Zustandsübergänge, die durch Events ausgelöst werden.

Begriff 2.2.2
Abbruch

ABBRUCH:
Liegt nach der Verarbeitung eines Zustands kein Folgezustand vor, so beendet sich die Verarbeitung.

2.2.2 Ausführen von Zustandsautomaten

Das Hochladen und Starten des Zustandsautomaten kann aus dem Kontextmenü des Zustandsautomaten heraus erfolgen. Es stehen drei Varianten zur Auswahl. Siehe hierzu Abbildung 2.5.

Mit Upload, Start und Animate wird der Zustandsautomat auf den Brick hochgeladen und falls die Option run automatically after upload (siehe Abbildung 2.6) aktiviert ist, wird der Automat auch gestartet. Ist zusätzlich noch Generate BT Animation Code aktiv, wird davon ausgegangen, dass der Robot animiert werden soll (Einfärben des aktiven Zustands im Diagramm). Eine entsprechende Kommunikation wird aufgebaut. Der Upload und das Starten des Programms kann auch mit einem USB Kabel erfolgen. Die Voraussetzungen hierfür entnimmt man der leJOS Dokumentation. Bei Run and Animate wird kein Upload durchgeführt. Und bei Animate Only muss das Programm vorher auf dem Brick gestartet sein.

Erfahrungsgemäß ist die Reihenfolge, in der man vorgeht sehr wichtig. Insbe-

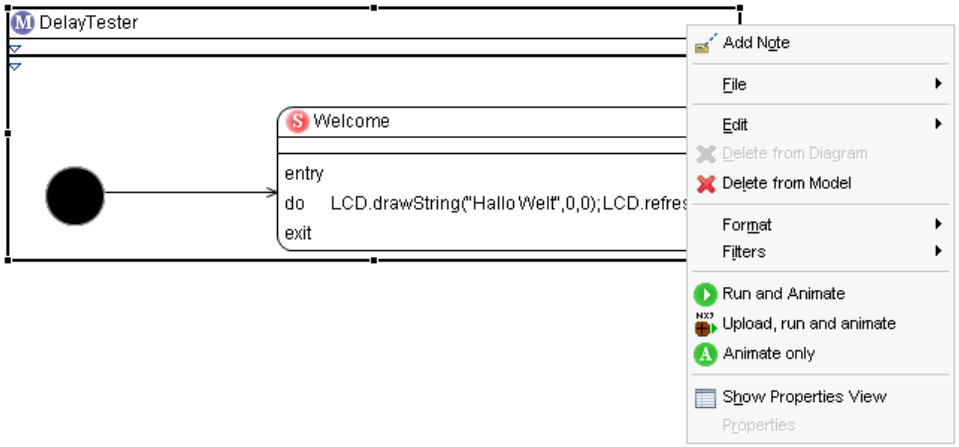


Abbildung 2.5: Hochladen und Starten eines Zustandsautomaten

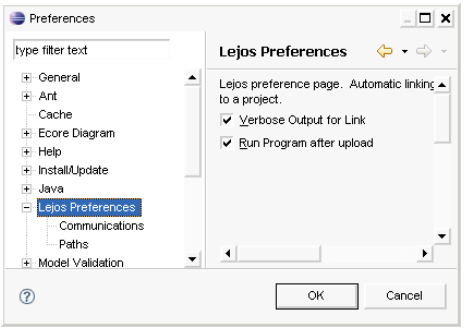


Abbildung 2.6: leJOS Einstellungen

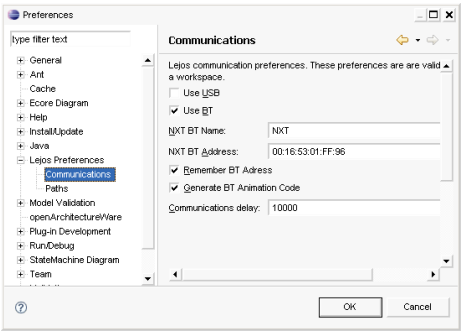


Abbildung 2.7: Kommunikationseinstellungen

sondere muss der Brick eingeschaltet sein, bevor der Upload gestartet wird. Verfügt das System über keine Bluetoothschnittstelle, so muss ein USB Bluetoothstick eingesteckt sein, bevor die Kommunikation gestartet wird. Bedauerlicherweise ist die Kommunikation nicht narrensicher, und nach einem Reihenfolgefehler ist zu empfehlen, sowohl den Brick als auch Eclipse neu zu starten.

Im Einzelnen bedeuten die Einstellungen folgendes

Einstellung	Bedeutung	Empfehlung
Directory name for Models	Name des Modell Verzeichnisses	Default models belassen
Directory name for Java Generation	Name des Verzeichnisses für generierte Java Klassen	Default src-gen belassen
Directory name for leJOS linked Files	Name des Verzeichnisses für gelinkte leJOS Dateien	Default bin belassen. Zum Überprüfen Navigator View benutzen.
Verbose Output for Link	Steuert, ob im Job die leJOS Method Records ausgegeben werden	Für Debugging auf Defaultwert true belassen.
Run Program after upload	Automatisches Starten nach dem Upload	true oder false, je nach Bedarf
Use USB	USB für Download oder Programmstart verwenden. Nur für Win. Keine Animation.	Kabellos rocks :-)
Use BT	BT für Download, Programmstart oder Animation verwenden. Keine Animation.	Default ausgewählt belassen.
NXT BT Name	Lesbarer Name für NXT Bricks	NXT belassen. Ändern des Namens ist Zukunftsmusik.
NXT BT Address	Adresse wird beim ersten Kontakt automatisch ermittelt	Nur eintragen, wenn bekannt. Für Animation erforderlich.
Remember BT Adress	Kenntnis der Adresse verkürzt Uploadzeit um Faktor 2	Nur löschen, wenn der NXT gewechselt wird.
Generate BT Animation Code	Animationscode mitgenerieren	je nach Bedarf true oder false
Communications delay	Verzögerung zwischen Programmstart und Animationsstart in ms	Default 10000 belassen, nur mit Vorsicht vermindern.

2.2.3 Explizite Events

Um einen Robot zu steuern, der einem Hindernis ausweicht, sind nun zwei Zustände erforderlich: ein Zustand „normale Fahrsituation“ und einer, der dem Ausweichmanöver entspricht. In der Abbildung 2.8 ist ein Automat mit diesen

Zuständen dargestellt.

EVENT:
Ein Event wird durch eine Bedingung beschrieben, die zu einem beliebigen Zeitpunkt auftreten kann. Diese Bedingung wird auch Trigger genannt. Das Framework überprüft alle Trigger des aktiven Zustands, solange er sein Verlaufsverhalten ausführt. Ist die Trigger Bedingung eines Events erfüllt, so wird das Verlaufsverhalten des aktiven Zustands unterbrochen.

Begriff 2.2.3
Event

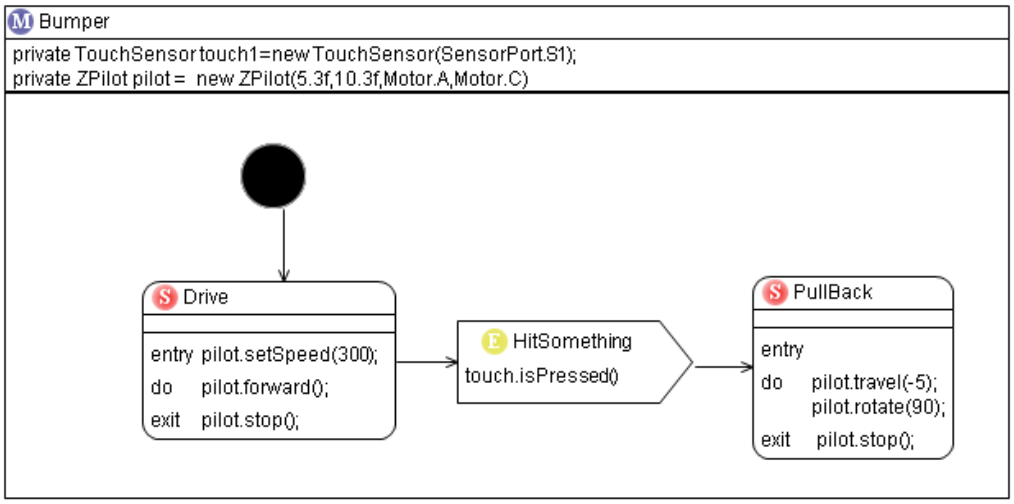


Abbildung 2.8: Durch Touch Sensor ausgelöstes Event

Das Verlaufsverhalten des zweiten Zustands „PullBack“ wird den Robot um eine festgesetzte Strecke zurücksetzen und dann um 90 Grad drehen. Der erste Zustand hat nun noch eine Besonderheit im Verlaufsverhalten. Die Anweisung „pilot.forward()“ läßt den Robot vorwärts fahren und nichts könnte ihn aufhalten. Hier kommt das Event HitSomething ins Spiel. Wenn der TouchSensor ausgelöst wird, hat das zur Folge, dass die Trigger Bedingung des Events erfüllt ist. Das Framework wird das Verlaufsverhalten des Drive Events unterbrechen und die Kontrolle an den Pull Back Zustand übergeben. Verlaufsverhalten können also sehr gut eine Endlosschleife haben. Diese kann durch Events unterbrochen werden. Nach Ausführung des Verhaltens des PullBack Zustands ist kein Folgezustand modelliert, deshalb wird gemäß 2.2.2 die Verarbeitung an dieser Stelle beendet.

Dies ist jedoch nicht sinnvoll, da nach dem Ausweichmanöver die Fahrt fortgesetzt werden sollte.

2.2.4 Implizite Events

Soll nach der vollständigen Durchführung des Verhaltens eines Zustands ein anderer angesteuert werden, so geschieht dies durch ein implizites Event.

Begriff 2.2.4
Implizites Event

IMPLIZITES EVENT:
Ein implizites Event löst einen Zustandsübergang aus, wenn das Verhalten des aktiven Zustands beendet wurde, ohne dass ein Event das Verlaufsverhalten beendet hat. Das Auslösen des Zustandsübergangs kann an eine Bedingung (Guard Condition) geknüpft sein.

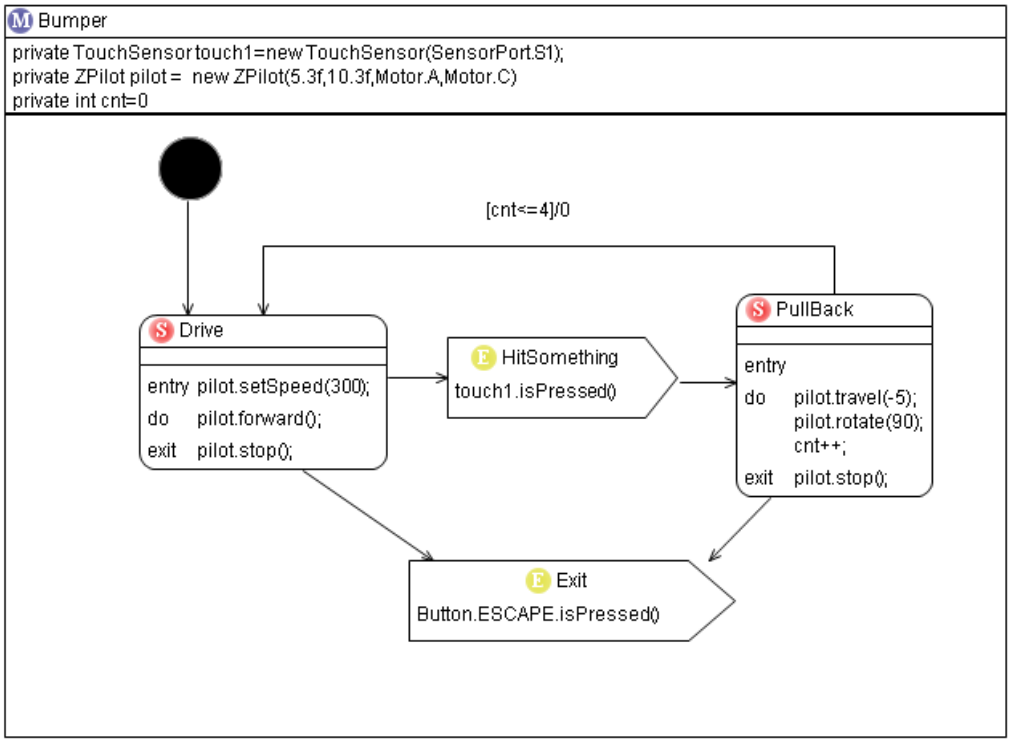


Abbildung 2.9: Das fertige Bumper Car

Nach Beendigung des Verhaltens des Zustands „PullBack“ wird durch ein implizites Ereignis die Transition in den „Drive“ Zustand ausgelöst. Dieser ist an die Guard Condition $cnt <= 4$ geknüpft, was zu Folge hat, dass sich die Maschine nach dem Entdecken von Hindernissen in allen vier Himmelsrichtungen abschaltet.

Schließlich wurde noch die Möglichkeit des Abbruchs der Verarbeitung aus je-

dem Zustand heraus modelliert. Es ist guter Stil, ein Programm kontrolliert zu beenden.

2.3 Beispiele

2.3.1 Fahren auf der Linie

Es ist eine Standardaufgabe, einen Robot zu bauen, der einer Linie folgen kann. Auch diese scheinbar langweilige Aufgabe kann auf viele verschiedene Weisen modifiziert werden. Man verwendet unterbrochene Linien oder Linien, auf denen Hindernisse stehen, die Umfahren werden müssen, oder man versucht die Zeit zu minimieren, die für eine gegebene Strecke benötigt wird.

Die von Lego mitgelieferte Versuchsstrecke zeigt schnell ihre Grenzen. Da das Papier nicht eben aufliegt, reflektiert es einfallendes Licht unterschiedlich. Selbst mit angeschalteter Lichtquelle des Lichtsensors werden beim Überfahren des scheinbar weißen Papiers sehr viele unterschiedliche Intensitätswerte gemessen. Das macht ein Kalibrieren des Sensors unausweichlich.

Für meine Robots verwende ich lieber einen flachen hellen Untergrund (Tisch oder weiß beschichtete Hartfaserplatte aus dem Baumarkt) und eine Rolle Isolierband. Das dünne Isolierband hat den Vorteil, dass es ohne große Mühe auch in engen Kurven geklebt werden kann, ohne dass Falten geworfen werden. Wer will, kann es auch in verschiedenen Farben erstehen, ich für meinen Teil habe mich bisher mit dem schwarzen begnügt.

2.3.2 Katzenjäger

Nützliche Erfindungen sind selten. Beim Katzenjäger handelt es sich um eine solche. Im Flur meines Büros tauchte eines Tages ein Katze auf, die angeblich einem Nachbarn gehörte, der aber zu alt war, um sich um die Katze zu kümmern. Zunächst hatten wir noch die Hoffnung, diese Katze loswerden zu können, wenn man nur unfreundlich genug zu ihr war. Aber leider hat der Hausmeister angefangen, sie zu füttern. Danach war aussichtslos, dass diese Katze das Haus jemals wieder verlässt. Wenn man nun in die Küche kommt, muss man damit rechnen, dass die Katze auf dem Stuhl liegt und nur mit handfesten Argumenten davon zu überzeugen ist, dass Stühle Menschen gehören.

Inzwischen kann man sicher sein, dass es keine 5 Minuten dauert, bis die Katze

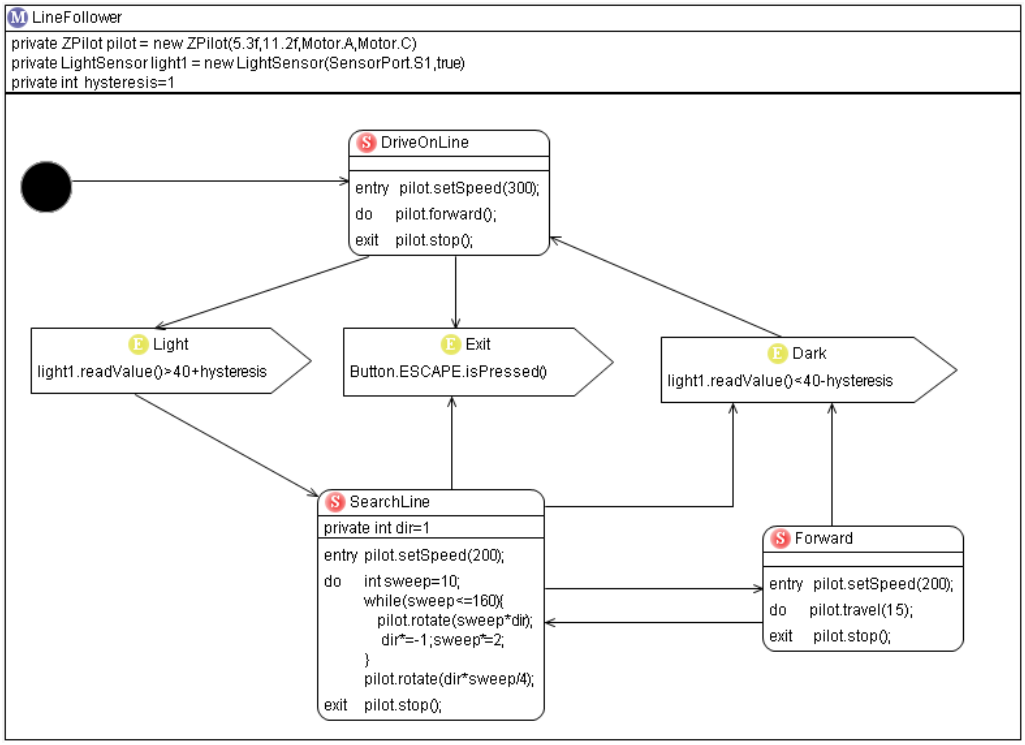


Abbildung 2.10: Fährt auf unterbrochenen Linien

hinten wieder drin ist, wenn man sie vorn rausgeschmissen hat. Diese Katze ist jedenfalls Ursache für die Erfindung dieses Robots. Den Tierschützern zur Beruhigung: Lebendige Katzen sind für den Katzenjäger nicht sichtbar. Das Fell absorbiert die Ultraschallwellen so gut, dass die Katze „schwarz“ ist. Also muss ein Katzenersatz her: Luftballons eignen sich hierfür hervorragend.

Dieser Versuch bedarf einiger Vorbereitungen. Verbrauchsmaterial pro Versuch: 2 mittelgroße Luftballons, 1 cm doppelseitiges Klebeband. Benötigt werden weiterhin: 1 handelsübliche Nadel für die Pinwand und ein Stift zum Aufmalen des Katzengesichts.

Für diesen Versuch werden die Katzen also durch Luftballons gedoubelt (siehe Abbildung 2.11). Man sollte nicht zu kleine verwenden, denn der Ultraschallsensor benötigt auch hier eine Mindestgröße. Die Luftballons sollten gut aufgeblasen sein, damit sie leicht zum Platzen gebracht werden können. Man befestigt sie mit einem doppelseitigen Klebeband (Teppichband) am Fußboden, damit sie nicht beim leisesten Windhauch davonfliegen. Das doppelseitige Klebeband klebt natürlich nicht am Knoten, sondern muss oben am Ballon angebracht werden. Zur Dekoration kann man noch ein Katzengesicht auf dem Ballon malen,

muss aber dabei beachten, dass der Ballon „verkehrtherum“ aufgeklebt wird.

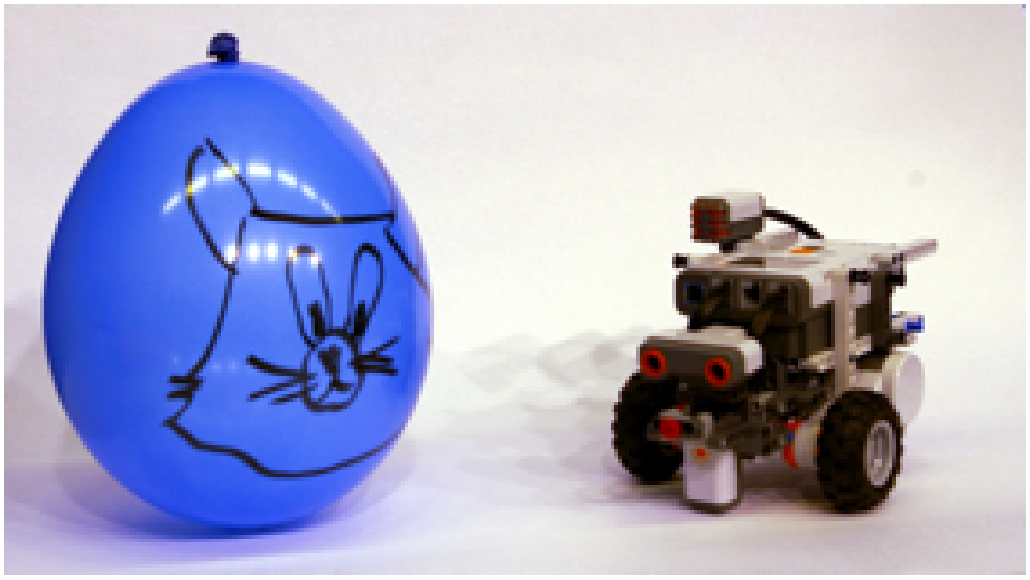


Abbildung 2.11: Katzenjäger in Aktion

Als Waffe für den Katzenjäger haben sich handelsübliche Pinnadeln bewährt. Sie lassen sich leicht so modifizieren, dass sie in die Löcher der Lego Leisten passen. Mit einer Feile oder einer Bohrmaschine als Drehbankersatz ist das Plastik schnell so weit abgetragen, dass die Nadeln wunderbar passen.

Abbildung 2.12: Pinnadeln modifiziert und original

Bei der Programmierung des endlichen Automaten ist zunächst das Problem zu behandeln, dass der Ultraschallsensor nur alle 200 ms abgefragt werden darf. Am zuverlässigsten scheint die Lösung, den Schallsensor in einem unabhängigen Teilautomaten laufen zu lassen, der nur alle 200 ms eine Messung vornimmt. Dieser Teilautomat hinterlegt die gemessenen Werte in einer Variablen.

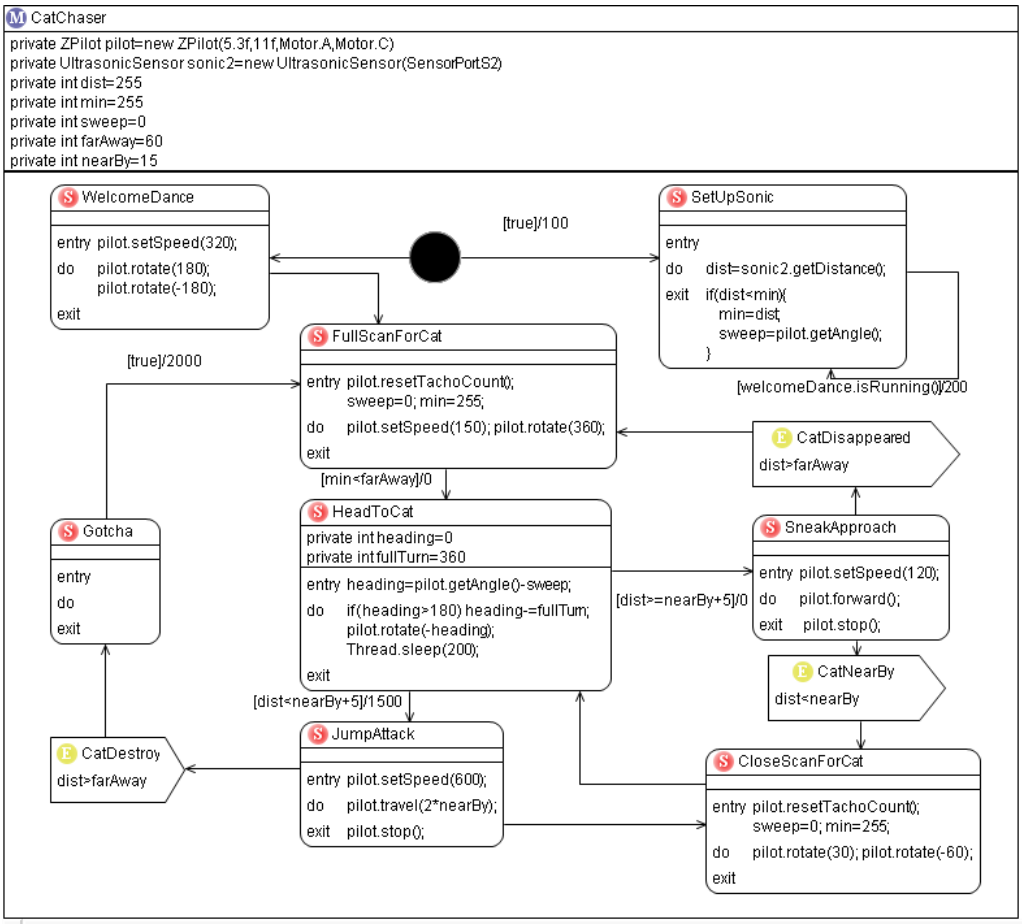


Abbildung 2.13: Jagd nach Katzenart: Anschleichen und Anspringen

2.3.3 Agility mit dem Hund

Aufgabe ist es, dem Robot die Kommandos „Links“ und „Rechts“ beizubringen. Er soll über diese Steuerung durch einen Hindernisparcour zu einem Ziel gebracht werden.

Von einem sprachgesteuerten Roboter zu sprechen, ist in diesem Fall doch ein wenig übertrieben. In der Tat soll der Robot auf Geräuschsignale reagieren. Aber es müssen nur hohe und tiefe Töne unterschieden werden. Etwa wie ein Hund, der auch nicht die Worte seines Herren versteht, sondern an der Tonlage erkennt, ob er gelobt oder getadelt wird. Das Wort „Links“ enthält den Vokal i, der langgedehnt sehr hoch ausgesprochen werden kann. So mancher Hundebesitzer kennt das Problem, einen Hund mit dem Kommando „sitz“ zum Sitzen zu

bringen. Erfahrenere Hundebesitzer bringen ihrem Hund das Kommando „Siieeeehhhtz“ bei. Spöttisch gesprochen „sie-tzen“ sie ihren Hund. Aber der Hund hat es deutlich leichter, sich an dieses übertriebene „i“ zu gewöhnen. Ähnlich verhält es sich mit unserem Robot. Man spricht also „Liiieeehhhnks“. Ähnlich spricht man „Rechts“ eher als „Röööhhhchts“, dann kann man eine tiefe Stimmlage erzeugen. Diese Unterschiede sollen aufgrund verschiedener dbA/db Messungen erkannt werden.

Experimente ergeben, dass die Unterschiede in den dbA/db Messungen nicht besonders stabil sind. Auch eine laute/leise menschliche Stimme beeinflusst den Unterschied. Deshalb benötigt man einen Glättungsmechanismus, der mehrere Messwerte nacheinander mittelt.

```
1  public class Mittler {
2   private int current = 0;
3   private int mittel = 0;
4   private int[] messwerte;
5
6   public Mittler(int messpunkte) {
7    messwerte = new int[messpunkte];
8    for (int i = 0; i < messwerte.length; i++) messwerte[i] = 0;
9   }
10
11  private int berechneMittel() {
12   int sum = 0;
13   for (int i = 0; i < messwerte.length; i++) sum += messwerte[i];
14   return mittel = sum / messwerte.length;
15  }
16
17  public int addMesswert(int messwert) {
18   messwerte[current] = messwert;
19   current = (current + 1) % messwerte.length;
20   return berechneMittel();
21  }
22
23  public int getMittel() {
24   return mittel;
25  }
26 }
```

Die grundsätzliche Idee beim Aufbau des Zustandsautomaten ist es, einen Teilautomat zu haben, der für die Geräuschmessungen verantwortlich ist, und einen zweiten Teilautomaten zu verwenden, um die Motoren zu steuern.

Der Teilautomat für die kontinuierlichen Geräuschmessungen benutzt den Mittler, um geglättete Werte für den Schalldruck zu haben. Er benutzt eine Verzögerungsschleife, um die Messungen mindestens 200 ms auseinander zu halten.

Der Teilautomat besteht aus einem Zustand, der fährt, und einem Zustand, der die Kommandos ausführt. Zwischen diesen Zuständen wird abhängig vom aktuellen Schalldruck (in dbA) hin- und hergeschaltet. Im Kommando ausführenden Zustand muss zwischen hohen und tiefen Geräuschen unterschieden werden. Der Robot fährt nach links, wenn die Differenz zwischen den Schalldrücken klein und nach rechts, wenn diese Differenz groß ist.

Der zugehörige Zustandsautomat ist in Abbildung 2.14 angegeben.

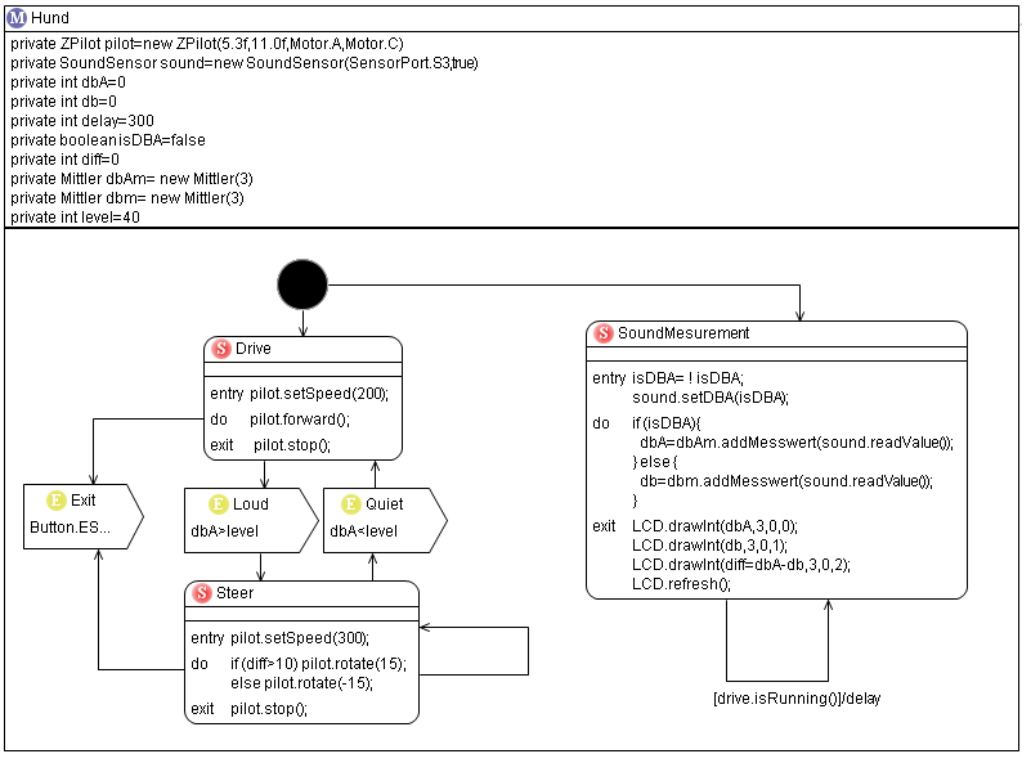


Abbildung 2.14: Ein Hund hört auf die Kommandos rechts und links

Typeset 26. Februar 2008