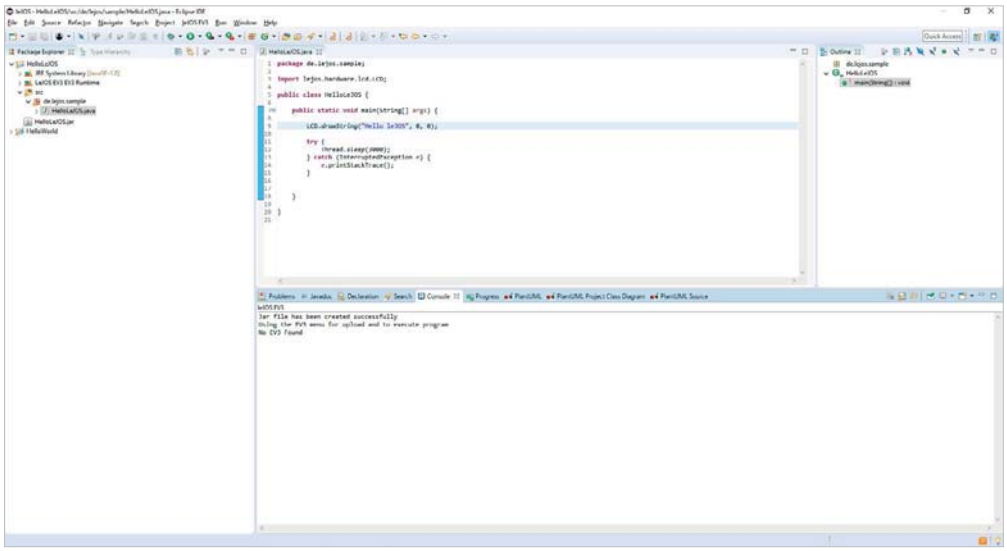


Abb. 3–33 // Eclipse-IDE: Ausgabe ohne gekoppelten EV3-Brick

Nach erfolgter Kopplung (siehe Kapitel 3.4) ist die Ausgabe folgendermaßen:

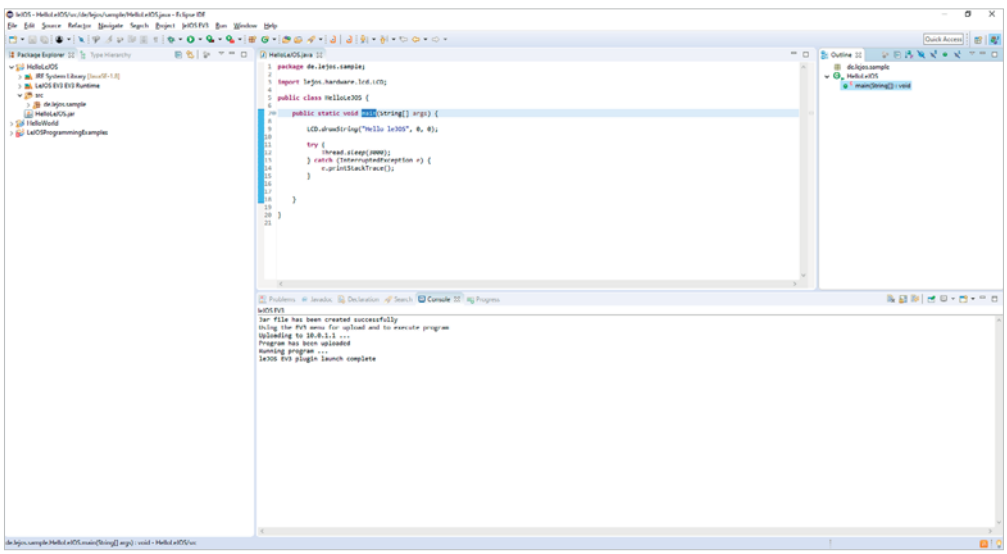


Abb. 3–34 // Eclipse-IDE: Ausgabe mit gekoppeltem EV3-Brick

Grundsätzlich lässt sich die Eclipse-IDE auch mit einem deutschen Sprachpaket ausstatten, sodass alle Menüs und Hinweistexte in Deutsch erscheinen. Dies lässt sich (wie auch das leJOS-Plug-in) über »Help« – »Install New Software« installieren. Nachdem allerdings für leJOS keine deutsche Lokalisierung existiert, wird von diesem Installationsschritt abgesehen.

3.3.4 Kurze Einführung in die Objektorientierung mit Java

In diesem Kapitel wird eine kurze Einführung zur objektorientierten Programmierung gegeben. Dabei wird auf einige wenige der wichtigsten Begriffe eingegangen. Es sollen hier nur die wichtigsten Grundlagen vermittelt werden, die für die direkte Programmierung des EV3 notwendig sind. Es ist nicht das Ziel, eine vollständige Vermittlung der Prinzipien der Objektorientierung zu versuchen, da hierfür ganze Bücher gefüllt werden.

Bei der prozeduralen Programmierung, wie sie zum Beispiel bei Small Basic verwendet wird, wird das Programm inhaltlich durch Prozeduren (sprich Unterprogrammen) strukturiert, die nacheinander gerufen werden können.

Im Gegensatz dazu geht die Objektorientierung davon aus, dass ein Gesamtsystem sich durch Objekte beschreiben lässt, die sich durch Eigenschaften und Aktivitäten/Funktionen auszeichnen. Durch die Verknüpfung von Objekten wird dabei ein Gesamtsystem aufgebaut.

Ein sehr beliebtes Beispiel ist dabei ein Auto. Ein Auto besteht aus verschiedenen Teilen, die Eigenschaften haben und durch Aktivitäten das Gesamtsystem beeinflussen. Das Auto hat dabei einen Motor und mehrere Räder. Ein Rad kann sich drehen und wird durch die Aktivität des Motors beeinflusst. Die Türen haben die Möglichkeit, sich zu öffnen, und haben als sichtbare Eigenschaften eine Farbe. Diese wenigen Beispiele sollen verdeutlichen, dass sich im Grunde alle Systeme als Verbindungen von Objekten darstellen lassen, die sich gegenseitig beeinflussen beziehungsweise sich gegenseitig bedingen.

Die Objektorientierung erfordert somit auch eine andere Art und Weise der Herangehensweise an die Erstellung eines Programms im Vergleich zu einer prozeduralen Programmiersprache.

In Java ist auch ein prozeduraler Programmierstil möglich. Bei der Umsetzung mit leJOS wird in diesem Buch sogar öfter explizit dieser Stil angewendet, um die Vergleichbarkeit der Lösungen in den drei Programmiersprachen besser zu verdeutlichen. Erst in den letzten Kapiteln wird stärker auf die Objektorientierung eingegangen, und die vorgestellten Lösungen werden auch mit entsprechenden Diagrammen erläutert.

In der Objektorientierung sind einige wichtige Kernbegriffe vorhanden, die die Grundlagen bestimmen.

- Klasse

  Eine Klasse stellt die allgemeine Definition eines Objekts dar und beinhaltet die Eigenschaftsbeschreibungen und Aktivitäten in verallgemeinerter Form, ohne dass diesen konkrete Werte zugewiesen sein müssen. Diese sind als Programmcode vorhanden. Eine Klasse kann nicht direkt ausgeführt werden.

  Ein Auto stellt dabei eine Beschreibung und Definition dar, das sowohl Eigenschaften (zum Beispiel Farbe) als auch Aktivitäten (zum Beispiel Starten des Motors) haben kann.

  Eine Klasse sollte in Java mit einem Großbuchstaben beginnen.

Es gibt einen wichtigen Hinweis für die folgenden Szenarien mit unterschiedlichen Umgebungen: Jede EV3-Software auf dem PC erfordert eine exklusive Verbindung. Das heißt, dass mit dem EV3-Brick immer nur genau 1 Programm auf dem PC verbunden sein kann. Oft stören bereits geöffnete Programme für die EV3-Programmierung, da zum Beispiel beim USB-Modus die Verbindung sofort zwischen Entwicklungsumgebung und EV3-Brick hergestellt wird. Dies führt zu sehr seltsamen Effekten (zum Beispiel nicht gefundene Kommunikationsendpunkte). Somit muss bei der Nutzung unterschiedlicher Programme mit Zugriff auf den EV3-Brick beachtet werden, diese nie parallel zu starten. Dabei zählt der EV3Explorer ebenfalls als eigenständiges Programm, was eine parallele Nutzung von Small Basic mit EV3 Basic und dem EV3Explorer unmöglich macht. Das EV3 Control Center von leJOS kann jedoch parallel zur Eclipse-IDE genutzt werden.

3.5 Bau des Beispielroboters

Der Beispielroboter ist ein sehr einfaches Modell, das einen fahrenden Roboter mit ein oder zwei Farbsensoren und einem Infrarotsensor kombiniert. Dieser Roboter besteht aus 58 Teilen, ist in weniger als einer halben Stunde aufgebaut und kann auch gut im Unterricht der HTML-basierten Anleitung folgend gebaut werden. Solange nur ein Farbsensor verwendet wird, reicht die Ausstattung der EV3 Home Edition vollständig aus.

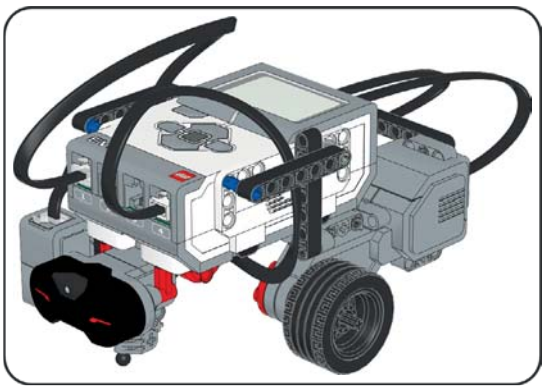
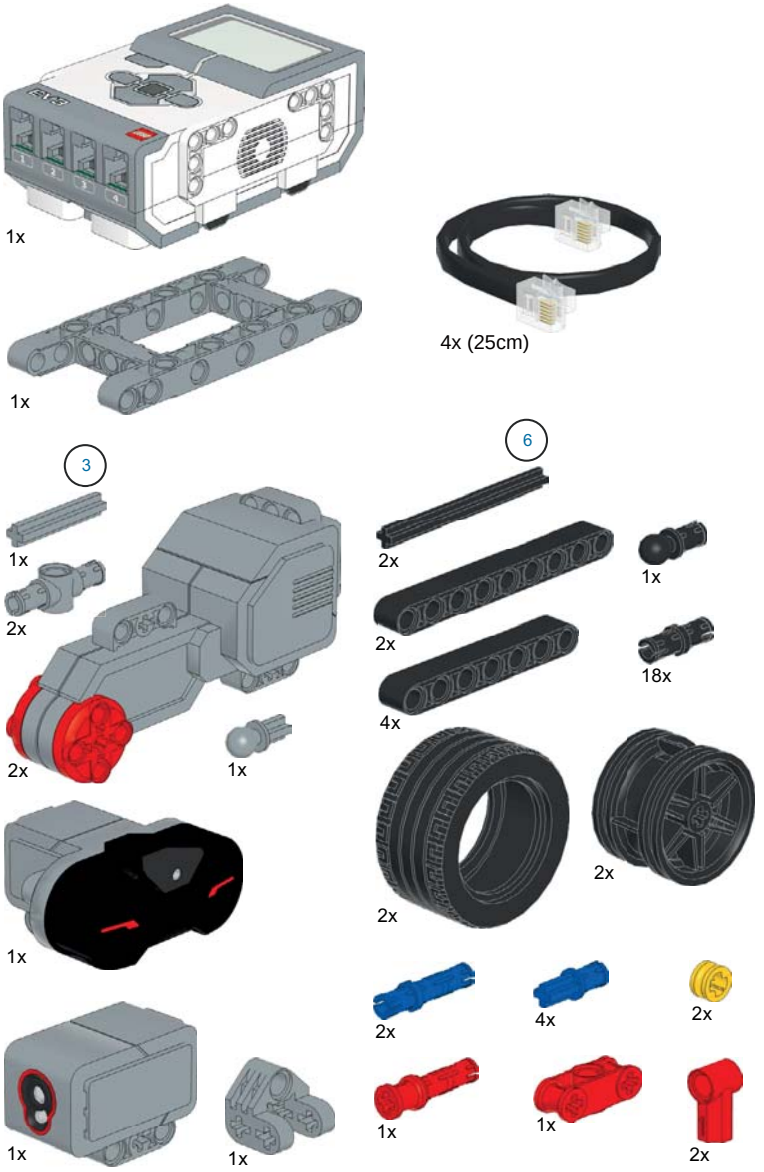


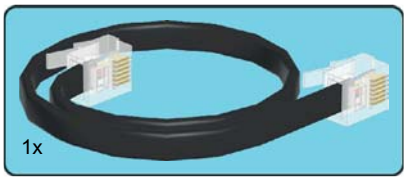
Abb. 3–58 // Gesamtansicht des Beispielroboters

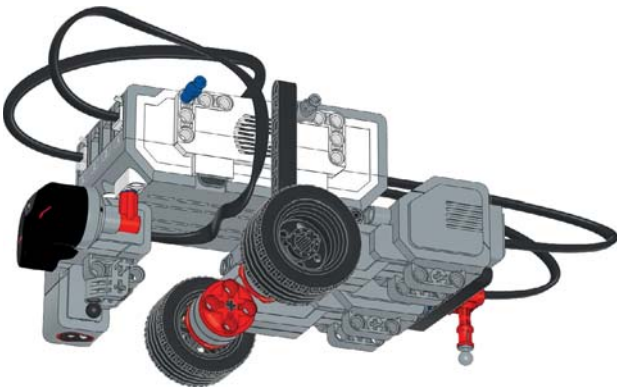
Dieser Roboter kann für typische Beispielszenarien wie Fahren, Hinderniserkennung, Ausweichen, Sound- und Displayanzeige genutzt werden und ist auch für erweiterte Problemlösungen wie die Linienverfolgung geeignet. Für die Teilnahme an Wettbewerben ist dieses Basismodell nur bedingt geeignet.

Eine Variante der folgenden Bauanleitung des Beispielroboters liegt im LDR- und LXF-Format vor und kann so direkt im LEGO Digital Designer geladen oder in alternative Programme wie das Virtual Robotic Toolkit importiert werden. Wie auch die HTML-Version stehen diese Dateien auf *dpunkt.de/programmev3* zum Download bereit.

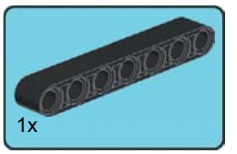

1x
4x (25cm)
1x
3
1x
2x
2x
1x
6
2x
1x
2x
18x
4x
2x
2x
1x
2x
4x
2x
1x
1x
1x
2x

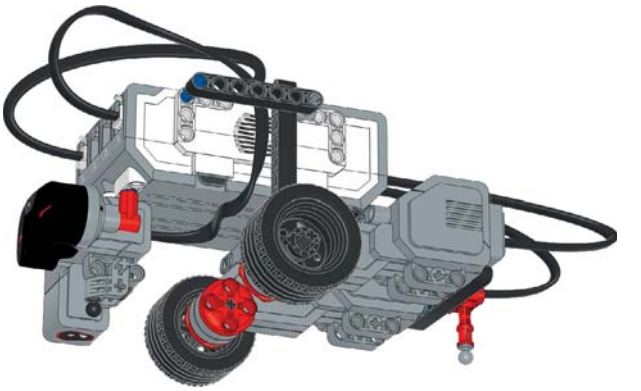
24





25





Hinweis für die korrekten Anschlüsse

Die Motoren müssen an die Ports B und C, der Farbsensor an Port 1 und der Infrarotsensor an Port 4 angeschlossen werden, damit die Beispielprogramme wie abgebildet und beschrieben korrekt funktionieren.

5 Weiterführende Themen der Programmierung

In diese Kapitel sollen weiterführende Themen der EV3-Programmierung betrachtet werden, die zum Beispiel bei Wettbewerben eingesetzt werden können (Linienverfolgung beim RoboCup Junior oder den WRO-Wettbewerben, Kommunikation für kooperative Roboter).

5.1 Linienverfolgung mit einem oder zwei Farbsensoren

In diesem Kapitel wird die klassische Betrachtung von neuen Programmierelementen verlassen, denn es kommen die verschiedenen bereits besprochenen Programmierelemente zum Einsatz, um einen Roboter einer Linie folgen zu lassen.

Dafür muss der Algorithmus für eine Linienverfolgung betrachtet werden. Grundlegend ist eine Linienverfolgung ein vergleichsweise einfaches Unterfangen, wenn der zugrunde liegende Algorithmus verstanden wird. Im Detail – vor allem bei Wettbewerben – können die Linienverfolgungen allerdings komplexer ausfallen, um zum Beispiel der Linie sanfter zu folgen oder Ähnliches. Dies wird in diesem Buch nur ansatzweise diskutiert.

Der entscheidende Algorithmus zum Folgen einer Linie basiert dabei auf der Annahme, dass der Roboter mit einem Farb-/Helligkeitssensor der Kante einer Linie folgt. Es wird nie direkt einer Linie gefolgt (das heißt, der Sensor ist genau auf der Linie), da der Roboter nicht entscheiden könnte, ob er nach links oder rechts ausgleichen muss, wenn der Sensor das Verlassen der Linie anzeigt. Wenn sich der Sensorwert allerdings auf eine Linienkante bezieht, bedeutet dies, dass der Roboter im Fall der Linienfarbe in die eine Richtung und im Fall der Untergrundfarbe in die andere Richtung ausgleichen muss. Der Roboter fährt somit »Schlangenlinien« beziehungsweise ruckelt an der Kante der Linie entlang.

Das folgende Schaubild verdeutlicht dieses Vorgehen am Beispiel einer schwarzen Linie auf weißem Grund:

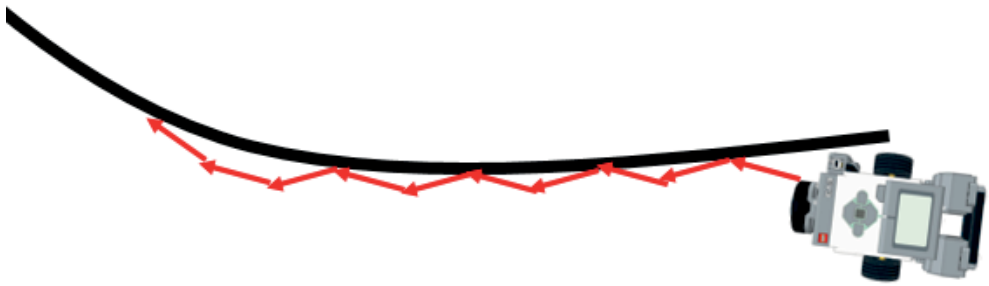


Abb. 5–1 // Linienverfolgung mit einem Farbsensor

Damit dies problemlos funktioniert, muss der Farbsensor circa 1 cm vom Boden entfernt sein, um die Farbinformation korrekt zu verarbeiten.

Eleganter wird die Linienverfolgung, wenn nicht die Linienfarbe für das Folgen herangezogen wird, sondern Helligkeitswerte genutzt werden. Dabei muss man wissen, dass die Sensoren von LEGO die Helligkeitswerte an einer Schwarz-Weiß-Grenze aufgrund des reflektierten Lichts berechnen. Wenn somit im Erfassungsbereich des Sensors mehr Schwarz als Weiß ist, wird daraus zum Beispiel ein 40-%-Helligkeitswert. Dieser Helligkeitswert kann somit für den Lenkausschlag genutzt werden – je weiter sich der Helligkeitswert vom Richtwert 50 % entfernt, desto stärker muss in die eine oder andere Richtung gelenkt werden. Dieses Vorgehen wird auch »proportionaler Linienverfolgungsalgorithmus« genannt und liefert wesentlich eleganter wirkende Fahrbewegungen, weil weniger harte Kurven auftreten.

Bei der Verwendung von Helligkeitswerten ist meist eine Kalibrierung vor der sinnvollen Nutzung der Sensoren notwendig, um den absoluten Schwarz- und Weiß-Wert zu bestimmen.

Aus Sicht der Programmierung wird dieser Algorithmus durch eine Schleife ausgedrückt, die bei jedem Schleifendurchlauf aufgrund der Farbwerte in die eine oder andere Richtung lenkt.

In vielen Anwendungsfällen ist allerdings die Linienverfolgung in der beschriebenen Art und Weise mit einem Sensor nicht ausreichend. Beispielsweise können Lücken in einer Linie oder Kreuzungen mit nur einem Sensor nicht erkannt werden. Bei einer Lücke würde der Roboter bei einer Verfolgung an der rechten Kante der Linie einfach eine Linkskurve machen, bis er wieder auf die Linie trifft und somit zurückfährt (180-Grad-Drehung).

Die Lösung liegt im Einsatz von zwei Farbsensoren, die die Linie zwischen sich einschließen. Dabei muss der Abstand der Sensoren von der Linienbreite abhängig sinnvoll gewählt werden. Der große Vorteil beim Einsatz von zwei Sensoren ist, dass damit Lücken in der Linie überbrückt werden können und bei einer geraden Linie ein absolut gerades Fahren ohne weiteren Ausgleich möglich ist. Bei einer schwarzen Linie auf weißem Untergrund würde somit der Roboter bei zwei weißen Rückmeldungen vorwärtsfahren und bei einem schwarzen Farbwert in die entsprechende Richtung lenken. Das folgende Schaubild verdeutlicht auch dieses Vorgehen ansatzweise.

Abb. 5–2 // Linienverfolgung mit zwei Farbsensoren

Wenn beide Sensoren Schwarz melden, ist die Wahrscheinlichkeit hoch, dass eine Kreuzung gefunden wurde. Dies kann allerdings auch bei einer rechtwinkligen Abzweigung passieren, wenn beide Sensoren einen Teil der Abbiegung der Linie erfassen (je nach Layout des Roboters). Die Reaktion darauf hängt von den äußeren Anforderungen ab und lässt sich pauschal nicht beantworten. Soll der Roboter zum Beispiel an einer Kreuzung immer rechts abbiegen oder soll er eine Kreuzung immer überfahren oder muss er weitere Informationen sammeln, um dies zu entscheiden. Beim RoboCup-Junior-Rescue-Line-Wettbewerb wird die Abbiegerichtung mit einem grünen Marker vor der Kreuzung gekennzeichnet. Bei Verwendung eines proportionalen Linienverfolgungsalgorithmus ist die gleichzeitige Auswertung einer weiteren Farbe (zum Beispiel das grüne Abbiegekennzeichen) relativ schwer mit dem gleichen Sensor, der auch für die Linienverfolgung genutzt wird, umzusetzen. Die LEGO-Farbsensoren können entweder die Helligkeit oder das Umgebungslicht messen oder eine Farberkennung machen. Ein permanentes Umschalten der verschiedenen Modi ist zwar möglich, liefert aber oft aufgrund der Trägheit der Sensoren nicht die gewünschten Ergebnisse. Somit müsste für derartige Anwendungsfälle in Kombination mit proportionalem Linienverfolgungsalgorithmus ein dritter Farbsensor für die Erkennung der Farbwerte genutzt werden.

Es gibt Spezialsensoren für das Folgen einer Linie, die aber bei vielen Wettbewerben nicht zugelassen sind und deshalb in diesem Buch auch nicht behandelt werden.

Für die Linienverfolgung sind die folgenden Aufgaben in diesem Kapitel zu lösen. Für die Aufgaben mit zwei Farbsensoren ist ein zusätzlicher Farbsensor notwendig, der in einem für die Breite der Linie sinnvollen Abstand zum ersten Sensor am Roboter anzubringen ist.

ID	Aufgabenstellung
A511	Linienverfolgung mit einem Farbsensor
A512	Linienverfolgung mit zwei Farbsensoren
A513	Proportionaler Linienverfolgungsalgorithmus mit einem Farbsensor
A514	Proportionaler Linienverfolgungsalgorithmus mit zwei Farbsensoren