Teil I

Die Hardware für die Roboter

In diesem Teil:

Kapitel 1

LEGO als Grundlage für unsere Roboter

Es ist eine gute Idee, LEGO-Komponenten zum Roboterbau zu verwenden. Ein aufgebauter Roboter kann immer wieder zerlegt werden und ein neuer erfunden und gebaut werden. Der eigenen Fantasie sind dabei grundsätzlich keine Grenzen gesetzt. Daher verwende ich in meinem Buch LEGO-Komponenten, um ein Roboterchassis zu erstellen. Ich lege dabei die Bauteile des LEGO-EV3-Baukastens zugrunde.

Leser, die diesen Baukasten nicht besitzen, aber über LEGO-Bausteine verfügen oder gewillt sind, sich diese anzuschaffen, können auch ganz entspannt sein. Denn ich gebe bei jedem Roboter eine Teileliste über die LEGO- oder Fremdkomponenten an, die ich für den jeweiligen Roboter verwende. Diese Teilelisten sind im Download zum Buch (`www.mitp.de/0310`) sowie in den einzelnen Kapiteln zu finden.

1.1 Roboter als Bausatz

Sie könnten sich natürlich, wenn Sie sich mit der Robotik beschäftigen möchten, auch einen Bausatz zu einem fertigen Roboter anschaffen, statt eigene Roboter aus LEGO zu bauen. Diese Roboter haben aber den Nachteil, dass es sich um fertige Maschinen handelt, die in ihren Freiheitsgraden eben deshalb beschränkt sind. Wenn man ihre Motoren und sonstigen Aktoren programmiert hat, ist es nicht mehr interessant, sich mit ihnen weiter zu beschäftigen.

Dazu hier einige Beispiele von Robotern, die ich mir unter anderem angeschafft habe (Abbildung 1.1 bis Abbildung 1.3).

Der YETI (Abbildung 1.1) ist ein einfacher Roboter, der über zwei Beine verfügt. Er ist mechanisch so aufgebaut, dass er mit zwei Servomotoren gesteuert werden kann. Um zu gehen, verlagert er sein Gewicht auf eines der beiden Beine und bewegt das andere vor. Danach wird das gegenüberliegende Bein belastet. Ich habe für den Roboter zusätzlich ein Vier-Segment-Display angeschafft und eingebaut. Damit kann er Meldungen oder seinen Status mitteilen.

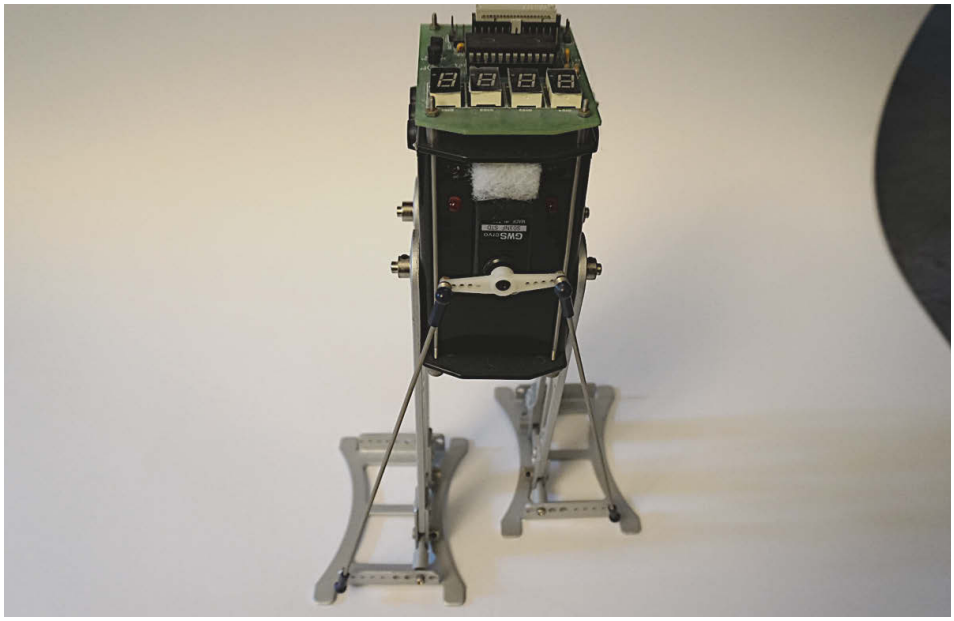

Abb. 1.1: Der YETI von AREXX Engineering

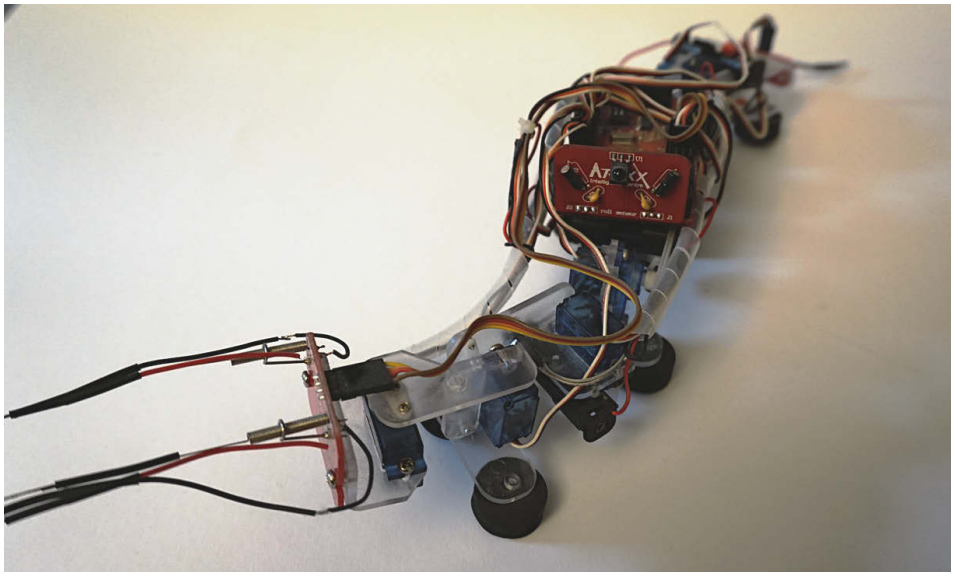

Abb. 1.2: Die Caterpillar von AREXX Engineering

Die Caterpillar ist ein wurmförmiger Roboter. Sein Körper ist in verschiedene Segmente geteilt, die sich durch Servomotoren jeweils auf und ab oder links und rechts bewegen lassen. Durch eine koordinierte Ansteuerung der Motoren ist der

Roboter in der Lage, sich fortzubewegen. Seine Sensoren sind zwei Antennen vorne und eine Antenne hinten sowie ein Rollsensor.

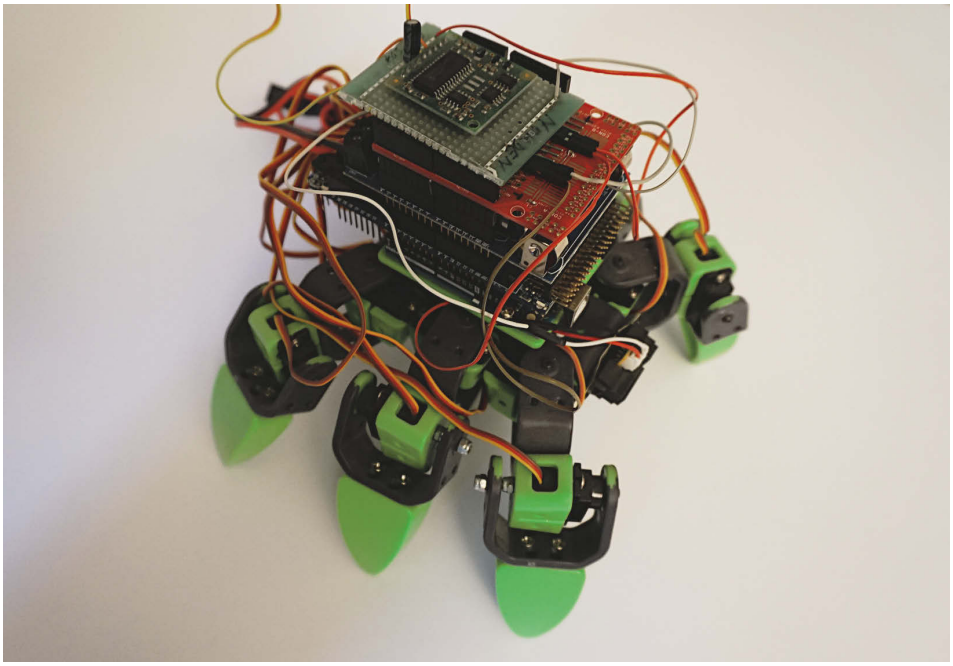

Abb. 1.3: Der ALLBOT von Velleman

Der ALLBOT ist ein sechsbeiniger Roboter, der entsprechend anspruchsvoll programmtechnisch gesteuert wird. Die sechs Beine müssen sich ja koordiniert bewegen, damit er sich vorwärtsbewegt. Dem Roboter habe ich einen Infrarotsensor (vorne am Chassis) spendiert, damit er Abstände zu einem Hindernis messen kann, sowie einen Kompass-Sensor (oberste Platine), der ihn befähigt, die Richtung zu bestimmen, in der er sich bewegt.

Natürlich kann man mit den vorgestellten Robotern vortrefflich experimentieren und diese auch mit zusätzlichen elektronischen Komponenten erweitern. Aber irgendwann hat man sie ausgereizt und dann stehen sie nur noch herum und sehen schön aus. Das ist schade.

Daher gehe ich in diesem Buch einen anderen Weg und verwende LEGO-Komponenten sowie elektronische Fremdkomponenten, um Roboter zu bauen. LEGO-Roboter haben den Vorteil, dass man sie, nachdem man sich ausführlich mit ihnen beschäftigt hat, zerlegen und dann einen neuen Roboter bauen kann. Wenn man LEGO für seine Roboter verwendet, wird dieses interessante Hobby niemals langweilig.

1.2 LEGO mit Elektronikkomponenten versehen

Das LEGO-Mindstorms-System bietet Computer, Motoren und Sensoren für den Bau von Robotern aus LEGO-Technikteilen an. Diese originalen LEGO-Bauteile sind im Vergleich zu Fremdkomponenten sehr teuer. Daher können sowohl die Originalteile als auch – als Alternative – elektronische Fremdkomponenten verwendet werden, wie Sie in den letzten drei Kapiteln zu den Roboterprojekten sehen können.

1.2.1 Folgende LEGO-Elektronikkomponenten werden verwendet

Für die Modelle im Buch verwende ich die folgenden LEGO-Komponenten.

Abb. 1.4: Der LEGO-Colorsensor

Abb. 1.5: Der LEGO-Infrarotsensor

Abb. 1.6: Der LEGO-Touchsensor

Abb. 1.7: Der Hitechnic-Kompass-Sensor

Abb. 1.8: Der mittlere Motor von LEGO EV3

Abb. 1.9: Der LEGO-EV3-Motor

Abb. 1.10: Der Hitechnic-Gyrosensor

1.2.2 Folgende Fremdkomponenten werden verwendet

Ein echter Elektronik-Bastler hat keine Angst vor Komponenten von Drittanbietern. Diese Bauteile haben den enormen Vorteil, dass sie teilweise nur einen Bruchteil der originalen LEGO-Komponenten kosten. Man muss allerdings bereit sein, auch mal etwas zu löten, da beispielsweise der Lichtsensor zusammengebastelt werden muss. Aber keine Bange, das kriegen Sie hin.

Ich verwende beim elektronischen Basteln einen 16-Watt-Lötkolben. Seine Leistung darf nicht zu hoch sein, damit die elektronischen Bauteile keinen Schaden nehmen. Dazu benötigt man noch eine Spule Lötzinn und wer es ganz komfortabel haben möchte, der legt sich noch eine Abisolierzange zu, fertig.

Als Fremdkomponenten setze ich die folgenden Bauteile ein.

Abb. 1.11: Der Lichtsensor (Fototransistor)

Abb. 1.12: Der Touchsensor (Mikroschalter)

Abb. 1.13: Die Raspberry-Pi-Kamera

Abb. 1.14: Getriebemotor plus Reifen und Encoder

Abb. 1.15: Die LED

Abb. 1.16: Der Piezo-Schallgeber

Teil III

Projekte

Sie haben jetzt einen Überblick über den Raspberry Pi sowie den BrickPi3 erhalten. Sie haben bereits etwas programmiert und ein paar Hardwarekomponenten eingesetzt. Das war sicherlich schon spannend, aber es geht natürlich noch mehr.

In den jetzt folgenden Kapiteln stelle ich jeweils einen Roboter vor. Ich gehe auf das Basteln des Roboters aus LEGO-Komponenten und in den letzten drei Kapiteln auf die Verwendung von Fremdkomponenten ein. Aber die Roboter werden natürlich auch programmiert.

Sie erfahren, wie Sie einen Roboter mit Motoren bewegen und Sie lernen die verschiedenen Sensoren kennen, mit denen ein Roboter gesteuert werden kann. Beispielsweise, wie ein Roboter mithilfe einer Kamera »sehen« kann.

Dabei lernen Sie die grundsätzliche Steuerung eines Roboters kennen, aber auch anspruchsvollere Anwendungen. Ich stelle Ihnen neuronale Netze vor, mit denen Roboter intelligent gesteuert werden können. Weiterhin arbeiten Sie sich in das Prinzip eines Expertensystems ein, das auch ein Instrument der Künstlichen Intelligenz ist.

Das alles soll Ihnen dieses spannende Gebiet näherbringen und so lernen Sie ganz nebenbei auch noch das Programmieren mit Python.

Projekte mit dem BrickPi3 und LEGO-Komponenten

In diesem Teil:

Kapitel 10

Wänden und Gegenständen ausweichen

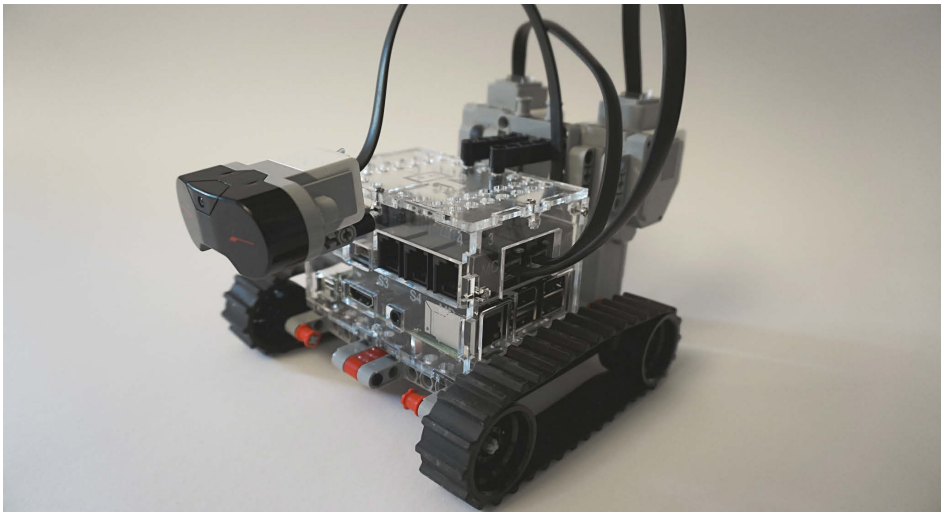

Abb. 10.1: Wänden und Gegenständen ausweichen

Der in diesem Kapitel vorgestellte Roboter hat die Aufgabe, Wänden und Gegenständen auszuweichen. Er verfügt über zwei Motoren, die ihn antreiben. Diese setzen dazu zwei Raupenriemen in Bewegung. Vorne am Roboter ist der Infrarotsensor befestigt, um den Abstand zu den Wänden und Gegenständen zu messen.

Starten Sie das Programm dieses Kapitels zunächst mit Tastatur und Bildschirm. Ziehen Sie dann die Kabel vom Raspberry Pi ab. Das sind: das HDMI-Kabel, das Tastaturkabel, das Mauskabel sowie das Kabel des Netzteils. Der Batterieschalter rechts neben dem Batterieanschluss muss dabei eingeschaltet und das Batteriepack angeschlossen sein. Wenn Sie den Roboter auf den Boden setzen und danach Ihre Hand weniger als fünf Zentimeter vor den Infrarotsensor halten, wird der Roboter eingeschaltet und fährt los. Trifft er auf eine Wand, stoppt er, setzt zurück und wendet ein wenig. Danach fährt er weiter geradeaus.

Das ist genau das Prinzip, nach dem die modernen Staubsaugerroboter funktionieren. Es fehlt nur noch einen Saugmotor unter dem Roboterchassis und fertig ist der Staubsaugroboter.

Sie können den Roboter ausschalten, indem Sie Ihre Hand wieder weniger als fünf Zentimeter entfernt vor den Infrarotsensor halten. Danach können Sie die erwähnten Kabel wieder einstecken.

Der Roboter in Aktion

youtu.be/
qCENsSaqnZU

Sie können ein Video mit dem Roboter im Einsatz sehen, wenn Sie dem QR-Code folgen.

10.1 Das LEGO-Modell

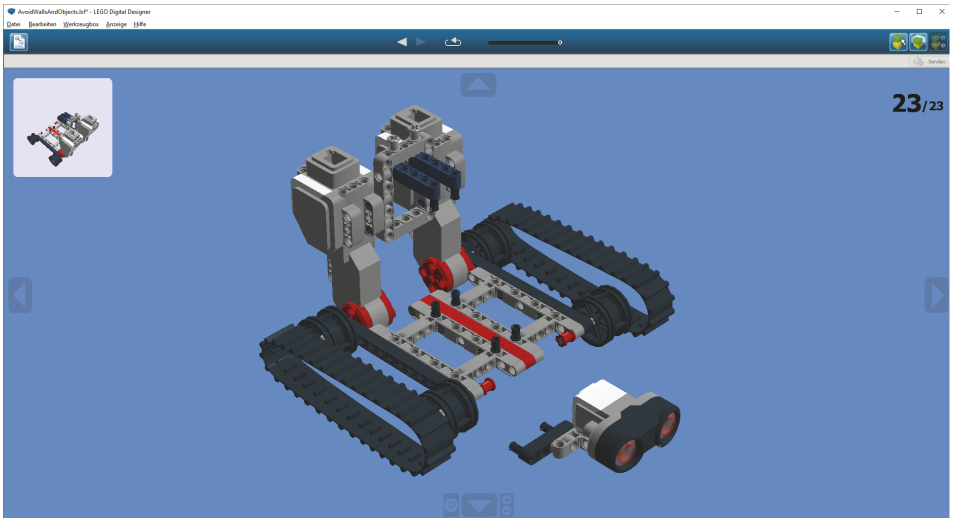

Abb. 10.2: Die Bauanleitung

Die Bauanleitung des Robotermodells rufen Sie mit dem LEGO Digital Designer auf. Wie Sie das Programm bedienen, habe ich in der Einleitung erläutert. Zum aktuellen Roboter gehört die Datei `AvoidWallsAndObjects.lxf`.

Den BrickPi3 platzieren Sie so, wie in Abbildung 10.1 dargestellt wird. Vorne am Gehäuse des BrickPi3 wird der Infrarotsensor von LEGO angebracht. Verbinden Sie ihn mit dem Port S1. Verbinden Sie den in Fahrtrichtung rechten Motor mit dem Port MB und den in Fahrtrichtung linken Motor mit MC. Die Maus und die

Tastatur verbinden Sie besser mit den oberen beiden USB-Ports, damit die Kabel nicht auf dem Riemen scheuern.

10.2 Die LED

Auf der Platine des BrickPi3 befindet sich links neben dem Batterieanschluss eine gelbe LED, die auch von den Programmen angesteuert werden kann. Bei unserem Roboter benutzen wir sie, um seinen Betriebsmodus anzuzeigen.

Ist der Roboter »ready to go« oder fährt nach vorne, blinkt die LED in einem Abstand von einer Sekunde. Fährt der Roboter rückwärts, blinkt sie mit einem schnellen und einem langsamen Blink, gewissermaßen als Warnung.

Zur Unterstützung der Steuerung der LED habe ich die Klasse `Led` entwickelt (siehe Listing 10.1). Diese wurde im Sinne der objektorientierten Programmierung (siehe Kapitel 9) erstellt.

```
#!/usr/bin/python
########################################
# class led                            #
# file: Led.py                         #
# Author: Th. Kaffka, Cologne, Germany #
# Date: 02.11.2020                     #
########################################

import brickpi3                         # ❶
import time
from threading import Thread

BP = brickpi3.BrickPi3()                # ❷
ok = True                               # ❸

class Led:

   def __init__(self):                  # ❹
      pass

   def __del__(self):                   # ❺
      BP.set_led(0)

   def ledOn(self):                     # ❻
      BP.set_led(100)
```

```
def ledOff(self):                    # ❼
   BP.set_led(0)

def blinkOn(self, time1, time2):     # ❽
   self.t = Thread(target=self.doBlink, args=(time1,time2,))
   self.t.setDaemon(True)
   self.t.start()

def blinkOff(self):                  # ❾
   global ok
   ok = False

def doBlink(self, time1, time2):     # ❿
   global ok
   ok = True
   while ok:                         # ⓫
      self.ledOn()
      time.sleep(time1)
      self.ledOff()
      time.sleep(time1)
      self.ledOn()
      time.sleep(time2)
      self.ledOff()
      time.sleep(time2)
```

Listing 10.1: LED-Steuerung (Led.py)

❶ Im Programm werden zunächst die nötigen Modulbibliotheken importiert. Neben der Bibliothek für den BrickPi3 und der Zeitbibliothek für die Zeitsteuerung wird die Bibliothek `threading` importiert. Diese dient dazu, das Blinken in einem parallelen Programmstrang zu verarbeiten. Das ist besser, da dann das Blinken unabhängig vom übrigen Programm verarbeitet wird und die LED gleichmäßig blinkt, während der Roboter gleichzeitig andere Aktionen durchführen kann.

Hinweis

Die Zeitsteuerung, die wir hier verwenden, ist das Statement `time.sleep()`. Das bewirkt, dass das Programm eine gewisse, angegebene Zeit anhält und danach weitermacht.

❷ Es wird ein Objekt in der Variablen `BP` erzeugt, das zur Steuerung des BrickPi3 dient.

❸ Dann wird die globale Variable `ok` auf `True` gesetzt.

❹ Der Konstruktor wird nur zu dokumentarischen Zwecken eingefügt. Ich will Ihnen damit zeigen, dass der Konstruktor existiert. Er verarbeitet hier aber nichts, daher das Statement `pass`.

❺ Im Destruktor wird die LED auf den Wert 0 (nicht leuchten) gesetzt. Dieser wird aufgerufen, wenn das Objekt gelöscht oder entfernt wird.

❻ In der Methode `ledOn` wird die LED auf 100% Leuchten gesetzt. Man kann der LED auch Werte zwischen 0 und 100 zuweisen, um sie beispielsweise schwächer leuchten zu lassen.

❼ Die Methode `ledOff` setzt die LED auf 0% Leuchten. Die LED ist dann aus.

❽ Die Methode `blinkOn` erhält zwei Parameter. Es handelt sich um zwei das Blinken steuernde Zeitparameter. Haben sie dieselben Werte, ist später das Blinken gleich lang. Sind deren Werte unterschiedlich, beispielsweise 1 und 0.1, dann blinkt die LED mit unterschiedlicher Geschwindigkeit, einmal lang, einmal kurz. Diese Methode startet als paralleles Programm (Thread) die Blinkmethode `doBlink` und übergibt ihr beide Parameter. Ich habe die Verarbeitung des Blinkens in einen *Thread* verpackt, damit das Hauptprogramm von dieser Steuerung entlastet wird und das Blinken immer mit denselben Zeiten stattfindet.

❾ Die Methode `blinkOff` setzt die globale Variable `ok` auf `False` und das Blinken hört auf.

❿ `doBlink` ist die Methode, die im Thread (also parallel) verarbeitet wird. Sie setzt zunächst die globale Variable `ok` auf `True`.

⓫ In einer Schleife, die ausgeführt wird, solange `ok` `True` ist, wird das Blinken mit den zwei Zeitparametern durchgeführt.

10.3 Der Infrarotsensor

Abb. 10.3: Der Infrarotsensor

Der Infrarotsensor (IR) hat die Aufgabe, Infrarotsignale zu erkennen. Wir benutzen ihn bei diesem Roboter, um den Abstand zu einem Gegenstand oder einer Wand zu bestimmen. Seine vom BrickPi3 verarbeiteten Werte liegen zwischen 100 (weit entfernt) und 0 (sehr nah). Er kann Gegenstände in einer Entfernung von bis zu 70 cm erkennen.

Auch für diesen Sensor habe ich eine Klasse erstellt, um ihn komfortabler handhaben zu können.

```
#!/usr/bin/python
########################################
# class lego infrared sensor           #
# file: LegoInfraRedSensor.py          #
# Author: Th. Kaffka, Cologne, Germany #
# Date: 21.10.2020                     #
########################################

import brickpi3                         # ❶
import time

BP = brickpi3.BrickPi3()

class LegoInfraRedSensor:

   def __init__(self, port):            # ❷
      self.port = port
      BP.set_sensor_type(port, BP.SENSOR_TYPE.EV3_INFRARED_PROXIMITY)

   def __del__(self):
      pass

   def getPort(self):                   # ❸
      return self.port

   def getDistance(self):               # ❹
      return BP.get_sensor(self.port)
```

Listing 10.2: IR-Sensor-Steuerung (LegoInfraRedSensor.py)

❶ Zuerst werden die Modulbibliotheken importiert und danach die globale Variable `BP` erstellt, in der sich das Objekt zur Steuerung des BrickPi3 befindet.

❷ In dem Konstruktor wird der Sensor mit seinem Port, der als Parameter übergeben wird, assoziiert. Dabei wird auch der Sensortyp definiert.

❸ Mit der `getPort`-Methode kann später auf die Portnummer zugegriffen werden.

❹ Mit dieser Methode wird die aktuelle Distanz zwischen dem IR-Sensor und dem Gegenstand oder der Wand ermittelt.