EV3-Programmierung mit TigerJython (Python)

1 Einführung

Im HaPra (Hardwarepraktikum) werden neben den EWB-Versuchen auch Roboterversuche durchgeführt. Hierzu werden programmierbare Lego Mindstorm EV3-Roboter genutzt, bei denen die Programmiersprache Python eingesetzt wird. Eine kleine Befehlsreferenz finden Sie in dieser Anleitung. Damit sollten Sie bei den Versuchen zurechtkommen. Weitergehende Informationen erhalten Sie z.B. aus dem Buch "Python for Informatics" (http://www.pythonlearn.com/book_007.pdf) oder auch z.B. im Internet unter „https://www.python.org/". Das genannte Buch ist in der Universitätsbibliothek als Download verfügbar.

Die Software „TigerJython" und einige Zusatzinformationenn finden Sie unter http://www.jython.ch/.

Um die Versuche durchführen zu können, erhalten Sie einen EV3-Roboter mit einer vorinstallierten SD-Karte, auf der Ihre Programme gespeichert werden können. Da der Roboter auch von anderen Gruppen verwendet wird, werden ihre Programme auf der SD-Karte bei jedem Einschalten wieder gelöscht. Zum Bearbeiten Ihrer Programme, sollten Sie ihre Versuche auf Ihrer Home-Partition (H:\) zwischenspeichern.

Lassen Sie die Roboter und Sensoren zusammengebaut. Falls Sie weitere Teile benötigen, wenden Sie sich an die Tutoren. Wenn Sie Sensoren oder den Roboter umbauen, bauen Sie diese am Ende jeder HaPra-Sitzung wieder zurück. Stellen Sie sicher, dass bei Beendigung Ihrer HaPra-Sitzung alle Materialien vollständig sind, da diese auch von anderen Gruppen verwendet werden (s. Bauanleitungen am Ende der Anleitung).
Fahren Sie am Ende jeder Sitzung die Rechner herunter und schalten Sie die Bildschirme und Roboter aus.

Hinweis: Im HaPra werden die Programme auf den Robotern bei jedem Bootvorgang automatisch gelöscht. Dafür wurde die Original-SD-Karte umprogrammiert.

2 TigerJython und EV3

2.1 Programm erstellen

Öffnen Sie zunächst auf Ihrem Desktop das Programm TigerJython. Warten Sie mit der Bearbeitung, bis in der Überschriftzeile das „Loading“ rechts neben TigerJython verschwunden ist (Abb. 2.1.1). Es dauert ca. 10 Sekunden, bis das Programm geladen ist. Nun können Sie auch schon loslegen. Geben Sie Ihren Programmcode ein oder klicken Sie auf das 2.Icon (Öffnen), um eine Datei zu laden. Ihr fertiges Programm können Sie mit dem 3.Icon (Speichern) in Ihrem Home-Directory (auf dem Server unter H:\) abspeichern. Im Beispiel (Abb. 2.1.2) ist ein Programm gegeben, das den EV3 eine Sekunde lang vorwärts (Zeile 7) fahren lässt. Es wird mit dem 4. Icon ausgeführt.

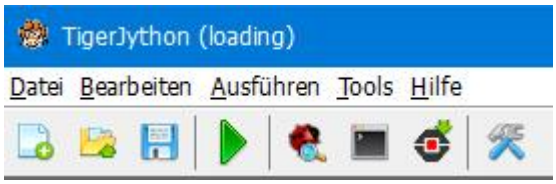


Abbildung 2.1.1

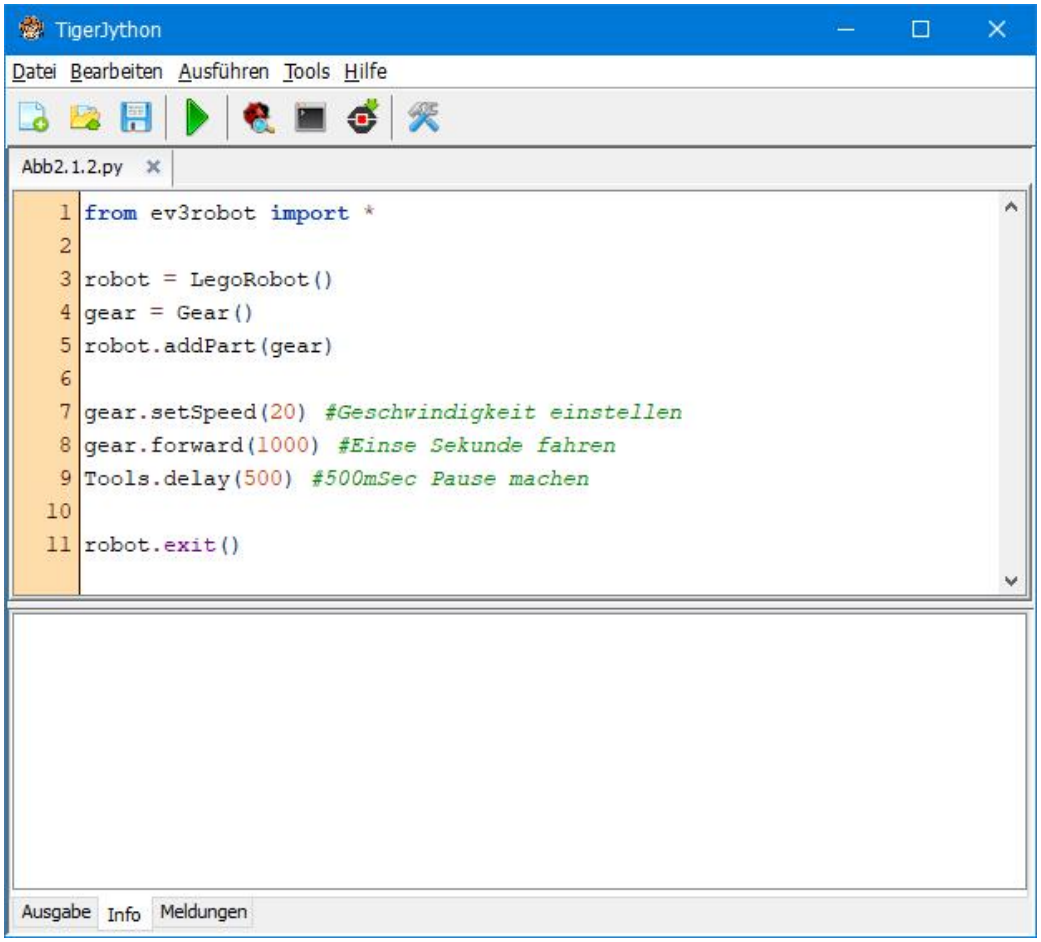


Abbildung 2.1.2

2.2 Programm am EV3 testen

Roboter mit Original-SD-Karte:
Schliessen Sie den EV3 mit einem USB-Kabel am Computer an. Schalten Sie den EV3 ein (Abb 2.2.1) (EV3-Enter) und warten Sie (bis zu ca. 65 sec), bis er betriebsbereit ist. Drücken Sie EV3-Enter, um „Run default“ (Abb 2.2.2) zu starten. Es wird das Programm „brickgate“ gestartet (ca. 28 sec), was die Pythonbefehle verarbeiten kann (Abb 2.2.3). Nun können Sie ihr Programm testen. Mit angeschlossenem USB-Kabel dient dazu das 4. Icon ist (Abb. 2.2.4) (das Aktuelle Programm ausführen) oder die Funktionstaste F5. Es erscheint ein Fenster (Abb 2.2.5) mit der IP-Adresse des EV3. Bestätigen Sie die IP (10.0.1.1) durch Drücken der Computer-Enter-Taste. Ihr Programm wird nun zum EV3 übertragen und dort ausgeführt. Wenn Ihr Programm syntaxfehlerfrei ist, reagiert der EV3 entsprechend ihrem Programmcode.

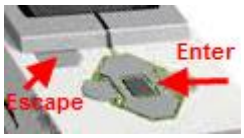


Abbildung 2.2.1

Abbildung 2.2.2

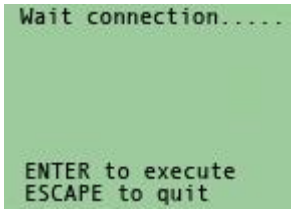


Abbildung 2.2.3

Roboter mit umprogrammierter HaPra-SD-Karte:
Man muss den Roboter lediglich einschalten. Er erscheint dann ein einziger Eintrag mit „Jython“. Bearbeitungsbeispiele sind in Abb. 2.2.5 und 2.2.6 zu sehen.

Abbildung 2.2.4

Abbildung 2.2.5

Abbildung 2.2.6

Wollen Sie ein Programm ohne angeschlossenes USB-Kabel betreiben, so müssen Sie Ihr Programm auf den EV3 laden. Dazu dient das 7. Icon (Abb. 2.2.7) (Hinunterladen/Ausführen) oder die Tasten Shift-F7. Danach wählen Sie auf dem EV3-Display mit den Cursortasten Ihr Programm aus und starten es mit der EV3-Enter-Taste. Auch hier dauert es etwas, ehe der EV3 ein Programm ausführt. Um ein auf den EV3 geladenes Programm zu beenden, drücken Sie die EV3-Escape-Taste. Wenn Sie das „brickgate“ beenden, müssen Sie es erneut starten (Run Default / Run Jython), um ein Programm am EV3 starten zu können.

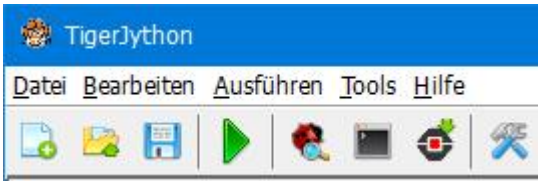


Abbildung 2.2.7

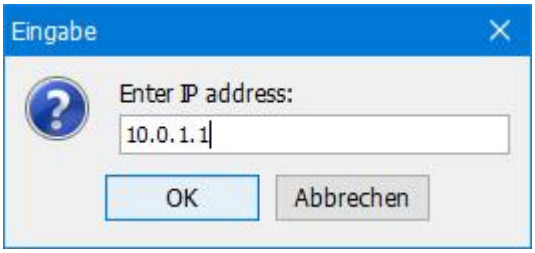


Abbildung 2.2.8

3 Programmierung der EV3 Roboter mit Python

Schliessen Sie die Motoren an die Ports A und B an. Für die Sensoren stehen die Ports 1-4 auf der gegenüberliegenden Seite zur Verfügung.

3.1 Motoren

Die Motoren können wie folgt gesteuert werden:

```
from ev3robot import * #EV3 Roboter verwenden
robot = LegoRobot()    #Roboter definieren
gear = Gear()          #Fahrwerk definieren
robot.addPart(gear)    #Fahrwerk am Roboter anschliessen

gear.setSpeed(20)      #Geschwindigkeit festlegen
gear.forward(500)      #500 ms vorwärts fahren (Millisek)
gear.backward (500)    #500 ms rückwärts fahren

gear.left (500)        #500 ms links drehen
gear.right (500)       #500 ms rechts drehen

gear.forward()         #dauernd vorwärts fahren
Tools.delay(500)       #500 ms warten
gear.stop()            #anhalten

robot.exit()           #Programm beenden
```

3.2 Text- und Tonausgabe

```
print ("Hallo")           #auf dem Rechner
robot.setVolume (10)      #Lautstärke anpassen
robot.playTone(440, 200)  #Frequenz 440 Hz, 200 ms Dauer
```

3.3 Warten und Zeit

Einfache Methode:

```
Tools.delay(500)       #500 ms warten
```

Um mit der internen Uhr des EV3 arbeiten zu können muss das time-Paket importiert werden:

```
from ev3robot import * #EV3 Roboter verwenden
import time            #Time-Paket importieren

t = time.clock()       #t = aktuelle Zeit als Dezimalzahl
```
(t in Sekunden nach dem Roboterstart mit 9 Stellen hinter dem Komma)

Um **Zeitdifferenzen** z.B. **beim Versuch „Klopfen“** zu erfassen, kann man die Differenz zwischen 2 Zeitmessungen bilden (t1-t2).

3.4 Zufallszahlen

Um mit Zufallszahlen arbeiten zu können, muss das random-Paket importiert werden:

```
from random import randint #Random-Paket importieren
z= randint (1, 100)        #z = Zufallszahl [1,100]
```

3.5 Funktionen definieren, Schleifen

Oft ist es sinnvoll, wiederkehrende Befehle in Funktionen auszulagern.

```
def ausgabe():           #Funktion ausgabe definieren
  global i               #i ist globale Variable
  print "Zahl", i        #„Zahl" und i ausgeben


for i in xrange(0, 5):   #for-Schleife von 0..4, nicht 5!
  ausgabe()              #ausgabe aufrufen


i = 0                    #i = 0
while i  < 5:            #while-Schleife von 0..4
  ausgabe()              #ausgabe aufrufen
  i = i + 1              #i erhöhen
```

Die Variable i wird auch ausserhalb von „ausgabe“ benutzt. Sie ist innerhalb von „ausgabe“ als global zu definieren.
Die for-Schleife durchläuft die Werte von 0 bis 4, obwohl eine 5 angegeben ist.
Die while-Schleife ist ähnlich wie die for-Schleife

3.6 Sensoren

Die Sensoren können an die Sensoranschlüsse S1..S4 angeschlossen werden. Die Initialisierung erfolgt entsprechend mit S1..S4 (siehe Beispiele mit S1).

Wie Sie die Sensoren an Ihrem Roboter anbringen, können Sie in den Aufbauanleitungen weiter hinten nachlesen.

Bei den Beispielen ergibt die Anweisung `while not robot.isEscapeHit():` eine Endlosschleife. Das Programm wird durch Drücken der Rechner-ESC-Taste beendet, wenn das Programm vom Rechner aus gestartet wird (4. Icon) und mit der EV3-ESC-Taste wenn das Programm vom EV3 gestartet wird.

3.6.1 Touch-Sensor (EV3 und NXT)

```
#EV3-Touchsensor an Port1
from ev3robot import*
robot = LegoRobot()

ts = TouchSensor(SensorPort.S1)
robot.addPart(ts)

#die ESC-Taste beendet das Programm
while not robot.isEscapeHit():
  if ts.isPressed():
     print "Taste gedrückt"
  else:
     print "Taste losgelassen"
  Tools.delay(500) #kleine Pause
robot.exit()
```

Beim NXT Touch-Sensor ist die Initialisierung anders als beim EV3 Touch-Sensor:

```
#Nxt-Touchsensor an Port1
ts = NxtTouchSensor(SensorPort.S1)
```

3.6.2 Sound-Sensor (NXT)

```
#Nxt-Soundsensor an Port1
from ev3robot import*
robot = LegoRobot()

s = NxtSoundSensor(SensorPort.S1)
robot.addPart(s)

#die ESC-Taste beendet das Programm
while not robot.isEscapeHit():
  if s.getValue() > 30:
     print "laut"
  else:
     print "leise"
  Tools.delay (500)

robot.exit()
```

Der Sound-Sensor stammt vom NXT-Roboter und muss deshalb mit `NxtSoundSensor` initialisiert werden. Der Wert „> 30“ dient nur als Beispiel und kann angepasst werden.
Die Rückgabewerte liegen zwischen 0 und 150.

3.6.3 Ultrasonic-Sensor (EV3 und NXT)

```
#EV3-Ultraschallsensor an Port1
from ev3robot import*
robot = LegoRobot()

us = UltrasonicSensor(SensorPort.S1)
robot.addPart(us)

#die ESC-Taste beendet das Programm
while not robot.isEscapeHit():
  print us.getDistance()
  Tools.delay (500)

robot.exit()


#Nxt-Ultrasonicsensor an Port1
us = NxtUltrasonicSensor(SensorPort.S1)
```

Die Methode `getDistance()` gibt die Entfernung zu einem Gegenstand zurück. Falls kein Gegenstand detektiert wird, wird 255 zurückgegeben. Ansonsten liegen die Rückgabewerte zwischen 3 und 150 (3 cm und 150 cm).

3.6.4 Licht-Sensor (EV3 und NXT)

```
#EV3-Lichtsensor an Port1
from ev3robot import*
robot = LegoRobot()

li = LightSensor(SensorPort.S1)
robot.addPart(li)

#die ESC-Taste beendet das Programm
while not robot.isEscapeHit():
  print li.getValue()
  Tools.delay (500)

robot.exit()
```

Die Rückgabewerte liegen zwischen 0 (keine Reflexion) und 1500 bei einem weissem Hintergrund.

```
#Nxt-Lichtsensor an Port1

li = NxtLightSensor(SensorPort.S1)
robot.addPart(li)
```

Die Rückgabewerte liegen zwischen 400 und 700.

3.6.5 Color-Licht-Sensor (EV3)

```
#EV3-Colorsensor an Port1
from ev3robot import*
robot = LegoRobot()

cs = ColorSensor(SensorPort.S1)
robot.addPart(cs)

#die ESC-Taste beendet das Programm
while not robot.isEscapeHit():
  print cs.getColorStr()
  Tools.delay(500)
robot.exit()
```

Die Methode `getColorStr()` liefert die entsprechenden Farben als string zurück.

BLACK BLUE GREEN YELLOW RED WHITE

3.6.6 Gyro-Sensor (EV3)

```
#EV3-Gyrosensor an Port1
from ev3robot import*
robot = LegoRobot()

g = GyroRateSensor(SensorPort.S1)
robot.addPart(g)

while not robot.isEscapeHit():
  print g.getValue()
  Tools.delay (500)

robot.exit()
```

Es wird die Drehbeschleunigung gemessen. Rechtsdrehung liefert negative Werte (0 bis -500), eine Linksdrehung positive Werte (0 bis +500).
Eine konstante Drehbewegung liefert eine 0, da es sich dabei nicht um eine Beschleunigung handelt.

```
g = GyroAngleSensor(SensorPort.S1)
```

Ausgehend von irgendeiner Startposition wird als Rückgabewert der Drehwinkel gemessen und zurückgegeben. Eine Rechtsdrehung um 90° liefert -90. Zwei komplette Linksdrehungungen liefern 2x360 = +720.

4 Tipps und Tricks

- Reset des Roboters: Die **ESC-Taste** und die **Mittlere Taste a**uf dem EV3-Roboter drücken. Wenn auf dem Bildschirm die Meldung „Starting“ erscheint, die Tasten loslassen.

- Wenn Sie bei einem der Versuche nicht weiterkommen, versuchen Sie an verschiedenen Stellen des Programms Textausgaben zu erzeugen, um Ihr Problem zu lösen.

- Wenn Sie unvorhergesehene Fehlermeldungen bekommen, überprüfen Sie, ob die Motoren an den Ports A (links) und B (rechts) und die Sensoren an den Ports 1-4 angeschlossen sind. Bei mehreren Sensoren des gleichen Typs überprüfen Sie, ob Sie die Input-Ports (S1..S4) richtig angegeben haben.

- Wenn sich eines Ihrer Programme nicht ohne Verbindung zum Rechner auf dem Roboter starten lässt, überprüfen Sie, ob Sie das USB-Kabel angeschlossen haben.

- Wenn die Verbindung zum Roboter nicht klappt, starten Sie TigerJython neu und warten Sie etwas, bis der Rechner die Verbindung erkannt hat. Das dauert einige Sekunden. Möglicherweise müssen Sie einen anderen USB-Port am Rechner verwenden.

- Bei Python sind Einrückungen Teil der Syntax. Überprüfen Sie bei Fehlern zunächst, ob alle Zeilen richtig eingerückt sind.

5 Liste der Sensoren

5.1 EV3 Sensoren:

- Touch Sensor - Berührungssensor
- Ultrasonic Sensor - Ultraschallsensor (Entfernung)
- Color Sensor - Farb- und Lichtsensor
- Gyro Sensor - Gyroskop

5.2 NXT Sensoren:

- Sound Sensor - Schalldruckmessgerät (Mikrofon)
- Light Sensor - Lichtsensor (Helligkeitsmessung)

5.3 Drittanbietersensoren - HiTechnic:

Wie diese Sensoren angesteuert werden, ist nicht bekannt.

- Magnetic Compass Sensor - Kompass
- Accelerometer Sensor - Beschleunigungssensor
- Color Sensor - Farbsensor
- Gyroscopic Sensor - Gyroskop

6 Bauanleitungen

6.1 Materialien

Folgende Materialien befinden sich in der Roboterkiste:

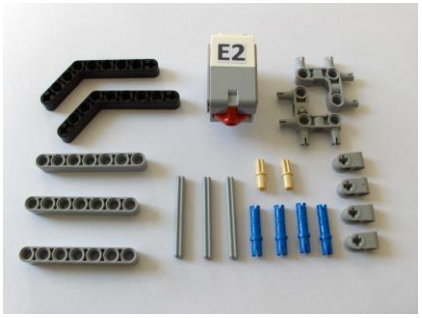
- EV3-Roboter
- USB-Kabel
- 2 Sensorkabel
- Touchsensor
- Ultraschallsensor
- Farbsensor
- Soundsensor
- Gyrosensor
- Lego-Anleitung
- Stromkabel
- NXT-Lichtsensor ist extern
- Mikrofon ist extern

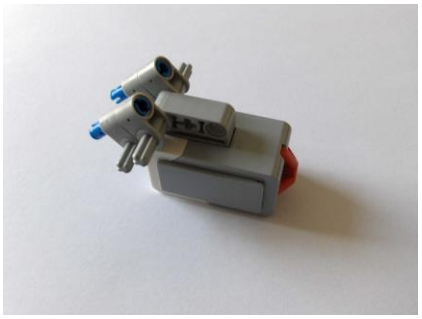
Stellen Sie sicher, dass die Materialien in der Roboterkiste stets vollständig sind.

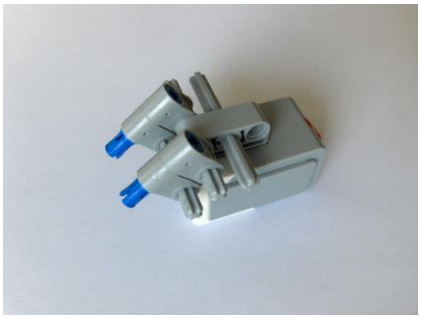
6.2 EV3-Roboter

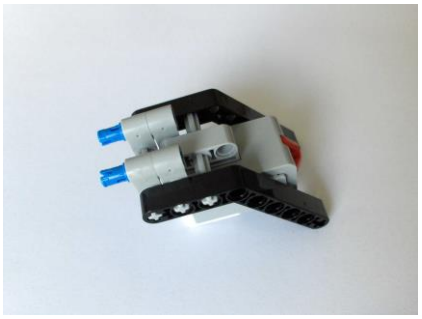
Falls Sie etwas an dem Aufbau des Roboters geändert haben, befolgen Sie bitte die Aufbauanleitung (S.7 bis S. 38) von Lego, die sich in ihrer Roboterkiste befindet, um den Roboter wieder in seinen Ausgangszustand zu bringen.

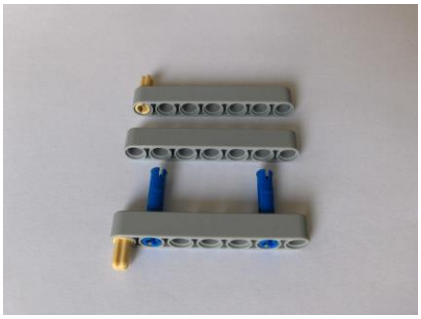
6.2.1 Touchsensor

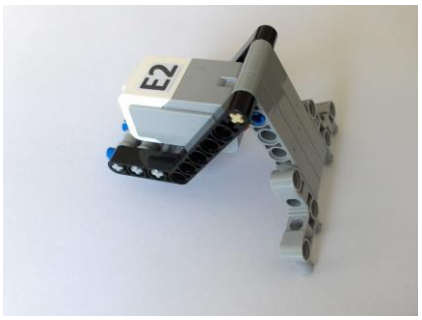




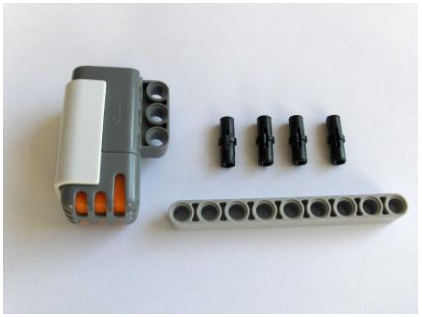


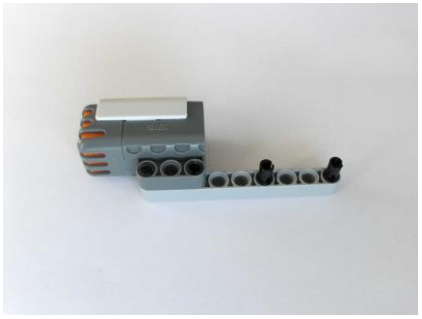


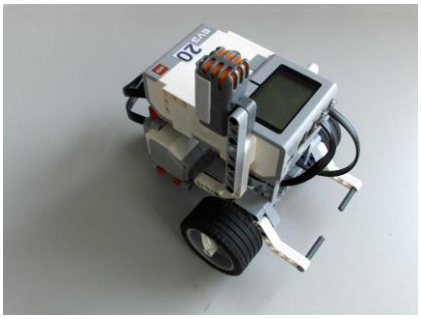




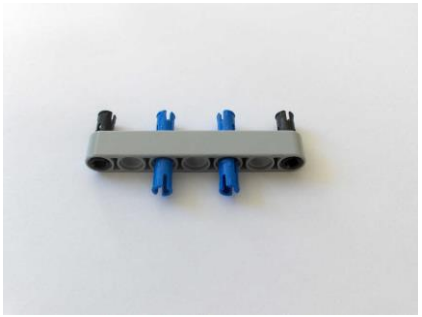
6.2.2 Soundsensor, Mikrofon

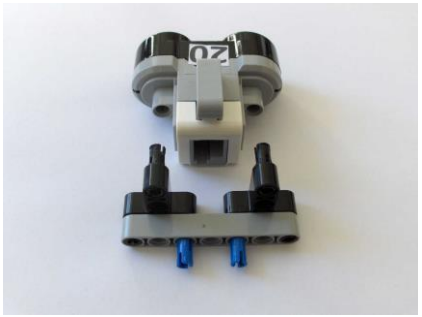


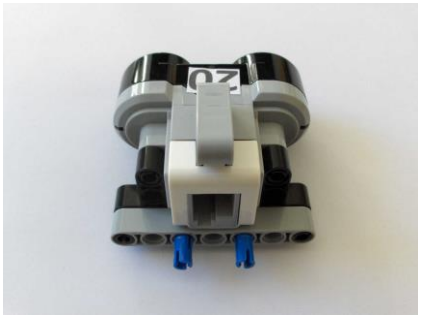


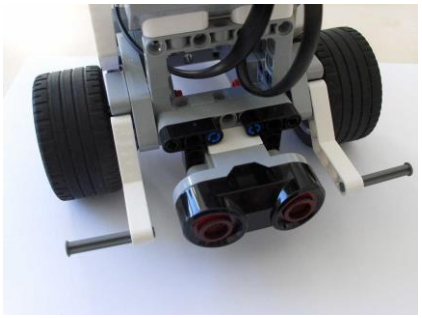


6.2.3 Ultraschallsensor

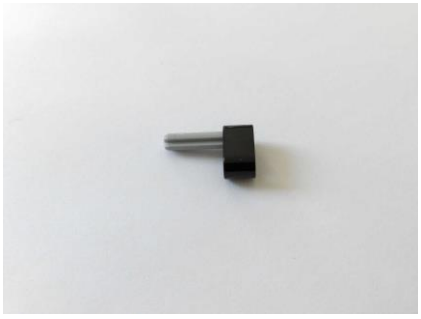


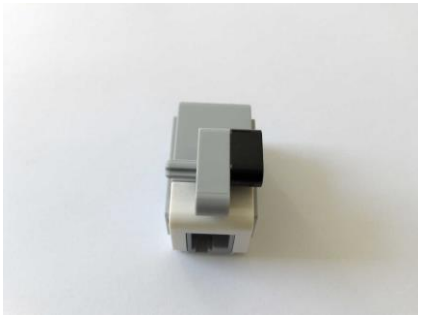


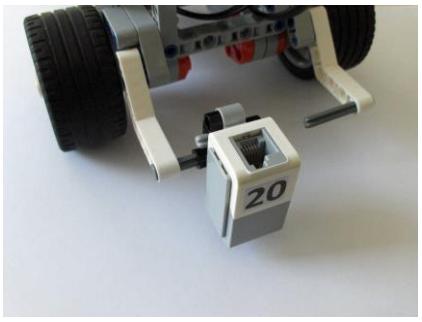




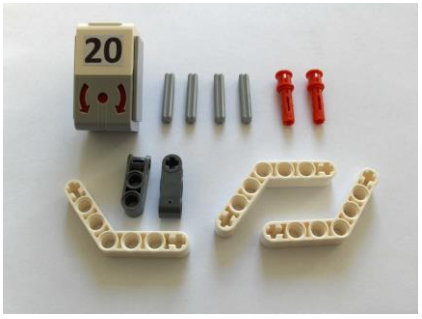
6.2.4 Licht-Farbsensor

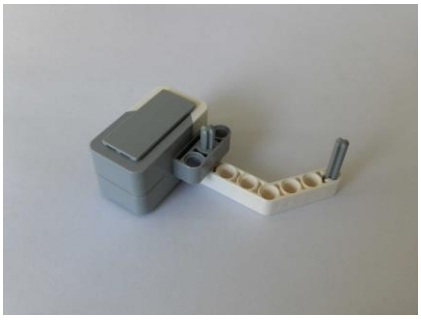


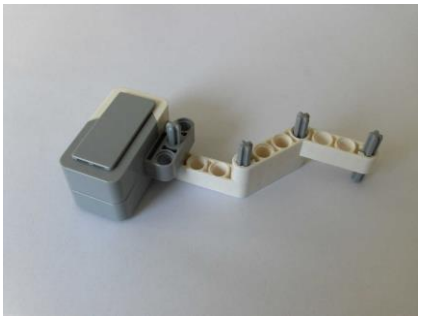


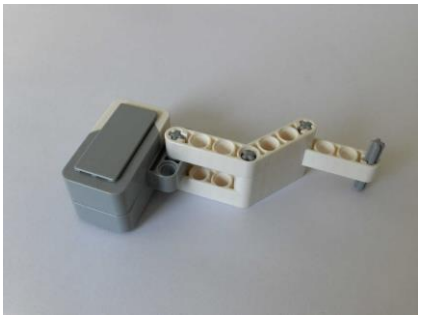


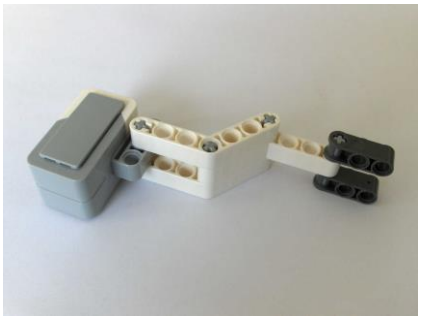
6.2.5 Gyrosensor

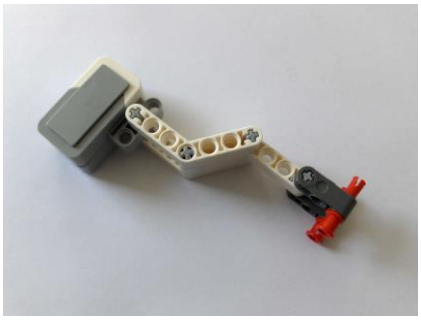


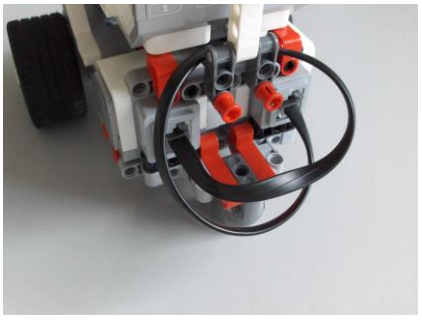


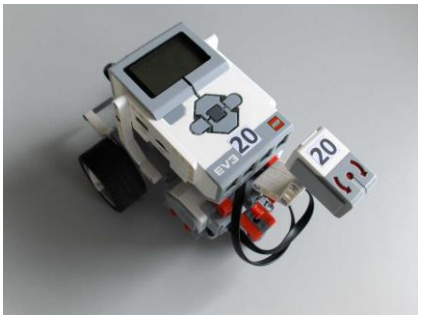












7 Befehlsreferenz für Tigerjython

Die Befehle wurden nicht einzeln überprüft und dienen als Hilfe zum Experimentieren.

LegoRobot:

Funktion	Action
LegoRobot()	erzeugt Roboter (ohne Motoren) und stellt die Verbindung zum Roboter her
addPart(part)	fügt eine Komponente zum Roboter
clearDisplay()	Löscht die Anzeige [*Simulationsmodus:* Status bar]
drawString(text, x, y)	Schreibt text an Position x, y [*Simulationsmodus:* in Status bar, (x, y) irrelevant]
isEscapeHit()	True, falls ESCAPE-Taste gedrückt wurde [*NXT und Simulationsmodus:* Tastaturtaste *Escape*]
isEnterHit()	True, falls ENTER-Taste gedrückt wurde [*NXT und Simulationsmodus:* Tastaturtaste *Escape*]
isLeftHit()	True, falls LEFT-Taste gedrückt wurde [*NXT und Simulationsmodus:* Tastaturtaste *Cursor-Left*]
isRightHit()	True, falls RIGHT-Taste gedrückt wurde [*NXT und Simulationsmodus:* Tastaturtaste *Cursor-Right*]
isUpHit()	True, falls UP-Taste gedrückt wurde [*NXT und Simulationsmodus:* Tastaturtaste *Cursor-Up*]
isDownHit()	True, falls DOWN-Taste gedrückt wurde [*NXT und Simulationsmodus:* Tastaturtaste *Cursor-Up*]
playTone(frequency, duration)	spielt einen Ton mit geg, Frequenz (in Hz) in Länge (ms) [*Simulationsmodus:* nicht vorhanden]
setVolume(volume)	setzt die Lautstärke für alle Tonausgaben (0..100)
playSampleWait(tag, volume)	dasselbe, aber Funktion blockierend, bis Datei fertig abgespielt. [*Simulationsmodus:* nicht verfügbar]
setLED(pattern)	setzt EV3-LEDS: 0: aus, 1: grün, 2: rot, 3: rot hell, 4: grün blinkend, 5: rot blinkend, 6: rot blinkend hell, 7: grün doppelblinkend, 8: rot doppelblinkend, 9: rot doppelblinkend hell
exit()	stoppt den Roboter und beendet die Verbindung
isConnected()	gibt True zurück, falls die Verbindung unterbrochen ist, oder das Simulationsfenster geschlossen wurde
reset()	*Simulationsmodus:* setzt den Roboter an die Startposition/Startrichtung

Gear:

Gear()	erzeugt ein Fahrwerk mit 2 Motoren an Port A, B
backward()	fährt rückwärts (nicht-blockierende Methode)
backward(ms)	fährt während gegebener Zeit (in ms) rückwärts (blockierende Methode)
isMoving()	gibt True zurück, wenn das Fahrwerk in Bewegung ist
forward()	fährt vorwärts (nicht blockierende Methode)
forward(ms)	fährt während gegebener Zeit (in ms) vorwärts (blockierende Methode)
left()	dreht links (nicht blockierende Methode)
left(ms)	dreht während gegebener Zeit (in ms) links (blockierende Methode)
leftArc(radius)	fährt auf einem Linksbogen mit geg. Radius (nicht- blockierende Methode)
leftArc(radius, ms)	fährt während gegebener Zeit (in ms) auf einem Linksbogen (blockierende Methode)
right()	dreht rechts (nicht-blockierende Methode)
right(ms)	dreht während gegebener Zeit (in ms) rechts (blockierende Methode)
rightArc(radius)	fährt auf einem Rechtsbogen mit geg. Radius (nicht blockierende Methode)
rightArc(radius, ms)	fährt während gegebener Zeit (in ms) auf einem Rechtsbogen (blockierende Methode)
setSpeed(speed)	setzt die Geschwindigkeit
stop()	stoppt das Fahrwerk
getLeftMotorCount()	gibt momentanen Zählerstand für den linken Motor zurück [nicht für Sim]
getRightMotorCount()	gibt momentanen Zählerstand für den rechten Motor zurück [nicht für Sim]
resetLeftMotorCount()	setzt den Zähler des linken Motors auf 0
resetRightMotorCount()	setzt den Zähler des rechten Motors auf 0

Tool:

Tools.delay(ms)	wartet ms MilliSekunden

Motor:

Motor(MotorPort.port)	erzeugt einen Motor am Motorport A, B, C oder D
backward()	dreht den Motor rückwärts
forward()	dreht den Motor vorwärts
setSpeed(speed)	setzt die Geschwindigkeit
isMoving()	gibt True zurück, wenn der Motor in Bewegung ist
stop()	stoppt den Motor
getMotorCount()	gibt den momentanen Stand des Zählers zurück [nicht für Sim]
resetMotorCount()	setzt den Zähler auf 0 [nicht für Sim]
rotateTo(count)	setzt Zähler auf 0, bewegt Motor bis Zählerstand count und stoppt (blockierend) [nicht für Sim]
rotateTo(count, blocking)	wie rotateTo(count), aber mit blocking False nicht blockierend [nicht für Sim]
continueTo(count)	wie rotateTo(count), aber Zähler nicht auf 0 gesetzt [nicht für Sim]
continueTo(count, blocking)	wie rotateTo(count, blocking), aber Zähler nicht auf 0 gesetzt [nicht für Sim]
continueRelativeTo(count)	wie continueTo(count), aber count ist Inkrement [nicht für Sim]
continueRelativeTo(count, blocking)	wie continueTo(count, blocking), aber count ist Inkrement [nicht für Sim]

LightSensor:

LightSensor(SensorPort.port)	erzeugt einen Lichtsensor am SensorPort S1, S2, S3, S4
LightSensor(SensorPort.port, bright = onBright, dark = onDark)	dasselbe mit registrierten Event-Callbacks onBright(port, value), onDark(port, value) beim Überqueren des Triggerlevels
LightSensor(SensorPort.port, True, bright = onBright, dark = onDark)	dasselbe mit registrierten Event-Callbacks onBright(port, value), onDark(port, value) beim Überqueren des Triggerlevels
activate(bool)	schaltet den LED eines NXT LightSensors ein/aus
getValue()	gibt den Wert des Lichtsensors zurück (Zahl zwischen 0 und 1000)
setTriggerLevel(level)	setzt den Triggerlevel (default: 500)

ColorSensor:

ColorSensor(SensorPort.port)	erzeugt einen Colorsensor am Port S1, S2, S3, S4
getColor()	gibt die gemessene Farbe zurück als Color-Objekt mit den Methoden getRed(), getGreen() und getBlue(), welche den RGB-Wert 0 - 255 liefern
getColorID()	gibt eine Farbidentifikationszahl zurück: 0: undefiniert, 1: schwarz, 2: blau, 3:grün, 4: gelb, 5: rot, 6: weiss
getColorStr()	liefert die Farbe als String (BLACK, BLUE, GREEN, YELLOW, RED, WHITE und UNDEFINED)
getLightValue()	gibt die Helligkeit (im HSG Modell) der gemessenen Farbe zurück

TouchSensor:

TouchSensor(SensorPort.port)	erzeugt einen Berührungssensor am SensorPort S1, S2, S3, S4
TouchSensor(SensorPort.port, pressed = onPressed, released = onReleased)	dasselbe mit registrierten Event-Callbacks onPressed(port), onReleased(port)
isPressed()	gibt True zurück, falls der Touchsensor gedrückt ist

SoundSensor:

SoundSensor(SensorPort.port)	erzeugt einen Soundsensor am Port S1, S2, S3, S4 (nur 1 Instanz erlaubt) Für den EV3 kann der Soundsensor des NXT verwendet werden. Muss aber mit **NxtSoundSensor(SensorPort.port)** erzeugt werden.
SoundSensor(SensorPort.port, loud = onLoud, quiet = onQuiet)	dasselbe mit registrierten Event-Callbacks onLoud(port, value), onQuiet(port, value) beim Überqueren des Triggerlevels. Beim EV3: NxtSoundSensor(SensorPort.port, loud = onLoud, quiet = onQuiet)
getValue()	gibt die Lautstärke zurück (0 - 150)
setTriggerLevel(level)	setzt den Triggerlevel (default: 50)

UltrasonicSensor:

UltrasonicSensor(SensorPort.port)	erzeugt einen Ultraschallsensor.
UltrasonicSensor(SensorPort.port, far = onFar, near = onNear)	dasselbe mit registrierten Event-Callbacks onFar(port, value), onNear(port, value) beim Überqueren des Triggerlevels
getDistance()	gibt die Entfernung zurück
setTriggerLevel(level)	setzt einen Triggerlevel (default: 10)