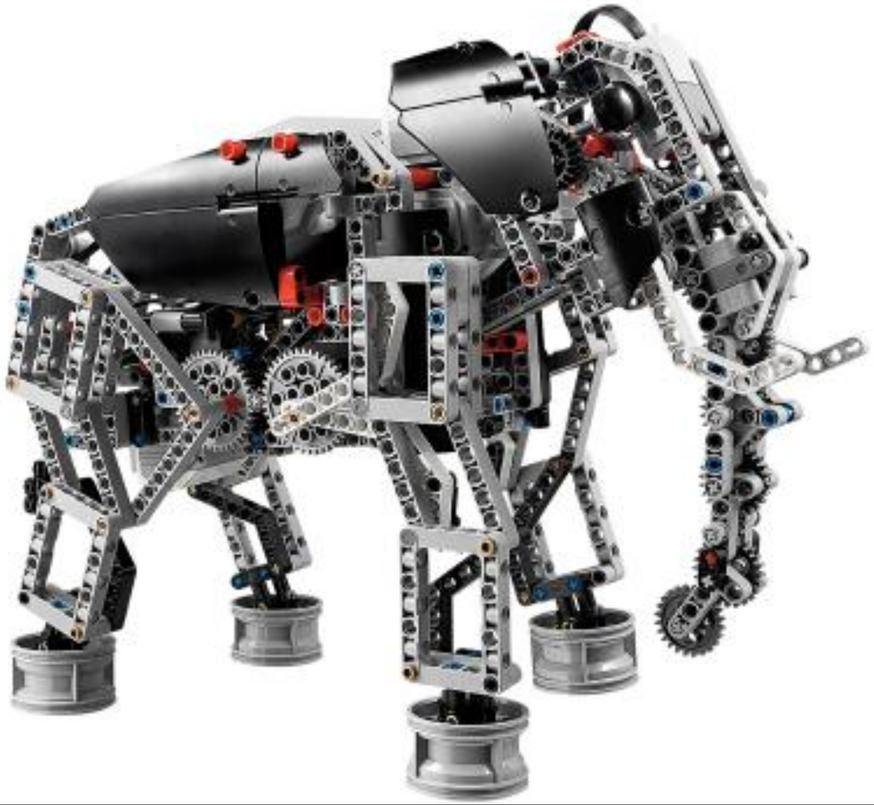


lego::lab

Raphaele Salvatore Licciardo
Projektarbeit WS1920
HotRod

Inhaltsverzeichnis

Auswahl

1. Kran
2. **Rennauto**
3. Bagger

Ideenfindung im laufe der Bauzeit

- Schnell wurde aus einem einfachen Rennauto, ein autonomes Alltagsfahrzeug.
- Ein Autonomes Fahrzeug sollte folgende Eigenschaften erfüllen
  - Hindernisse erkennen und abbremsen (ggf. ausweichen).
  - Dem vorausfahrenden Fahrzeug folgen.
  - Verkehrszeichen erkennen und darauf angemessen reagieren.

Lenkung

- Wie baut man etwas dynamisches mit starren Lego Steinen?

Problematik der Variante 1

- Das fordere Rad bzw. Die vorderen **Räder schleifen** in einer Kurve.

Variante 1: Fest verbaute Vorderräder

- Die Vorderräder werden fest verbaut
- Die Hinterräder bekommen separate Motoren.
- Die Lenkrichtung wird mit einer unterschiedlichen Motoren Geschwindigkeit der jeweiligen Seite bestimmt.
  - Der Linke Motor fährt schneller wie der Rechte Motor: das Auto fährt eine **Rechtskurve**
  - Der Rechte Motor fährt schneller wie der Linke Motor: das Auto fährt eine **Linkskurve**

Lenkung

- Wie baut man etwas dynamisches mit starren Lego Steinen?

Problematik der Variante 2

- Wie kann der Quellcode den Antriebsmotor und **Gleichzeitig** den Lenkmotor steuern?

= Der Antriebsmotor wird Zeitlich begrenz gestartet. Somit kann der Quellcode den Antriebsmotor starten und sich um den Lenkmotor kümmern.

Variante 2: Flexible Vorderachse

- Die Vorderräder werden mit einer Achse verbaut.
- Die Hinterräder bekommen einen Motor.
- Die Vorderrad Achse bekommt ein Lenkmotor der mittels Zahnräder mit der Achse verbunden ist.
  - Der Lenkmotor dreht sich rechts herum: das Auto fährt eine **Rechtskurve**
  - Der Lenkmotor dreht sich links herum: das Auto fährt eine **Linkskurve**

Gleichzeitigkeit

- Ein Quellcode kann nur eins nach dem anderen abarbeiten.
- Wie kann er sich gleichzeitig um den Lenkmotor und um den Antriebsmotor kümmern?

Variante 1: Multithreading

- Der Quellcode bekommt für jeden Motor, Sensor, ... einen eigenen Thread der sich nur darum kümmert.
- Somit ist eine Synchronität bzw. Gleichzeitigkeit in der Programmierung möglich.

Problematik der Variante 1

- Ein Lego Auto das Multithreading benötigt um Lenken und gleichzeitig Fahren (Gas und Bremse) zu können ist dann doch etwas zu viel.
- Das muss auch leichter gehen.

Gleichzeitigkeit

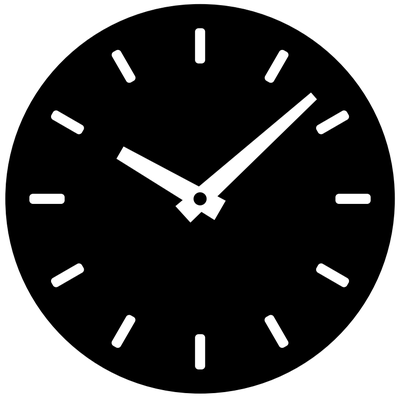
- Ein Quellcode kann nur eins nach dem anderen abarbeiten.
- Wie kann er sich gleichzeitig um den Lenkmotor und um den Antriebsmotor kümmern?

Problematik der Variante 2

- Zunächst keine.
- **Optimale Lösung**

Variante 2: Asynchronität

- Der Antriebsmotor bekommt eine Zeitliche Vorgabe wie lange dieser laufen soll.
- Danach ist die Abarbeitung des Vorwärtsfahrens abgeschlossen.
- Der Compiler kann sich ganz der Lenkung hingeben.



Follow-me

- Wie funktionieren moderne Autonome Fahrzeuge? Mit der Follow-me Funktion soll ein selbstfahrendes Fahrzeug simuliert werden, welcher genau dies darstellet.

Problematik der Variante 1

- Was passiert bei **Kurven**?

Variante 1: 1 Infrarot Sensor

- Der HotRod bekommt in Front (über der „Motorhaube“ einen Infrarot Sensor.
- Dieser sucht die Umgebung vor ihm nach Hindernissen ab und berechnet die Entfernung dazu.
- Die Geschwindigkeit wird so angepasst, dass es einen konstanten und gleichmäßigen abstand zu dem Objekt vor ihm gibt.
- Damit fährt das Auto dem vorausfahrenden Fahrzeug hinterher.

Follow-me

- Wie funktionieren moderne Autonome Fahrzeuge? Mit der Follow-me Funktion soll ein selbstfahrendes Fahrzeug simuliert werden, welcher genau dies darstellet.

Problematik der Variante 2

- Was passiert bei **scharfen Kurven**?
- Was passiert bei **mehreren Richtungsänderungen** außerhalb der Reichweite?

Variante 2: Pixy Kamera

- Die Pixy Kamera ist eine Kamera von Lego, welches mit einer „KI“ versehen ist.
- Sie legt um alle Objekte einen Kasten und merkt sich diese. Im Quellcode kann man diese Objekte ansprechen.
- Fährt das Fahrzeug nach Links, so versteht die Pixy Kamera dies und sendet ein entsprechendes Kommando.
- Selbst bei leichten Kurven, kann die Pixy Kamera nun das Auto tracken, und diesem folgen.

Follow-me

- Wie funktionieren moderne Autonome Fahrzeuge? Mit der Follow-me Funktion soll ein selbstfahrendes Fahrzeug simuliert werden, welcher genau dies darstellet.

Problematik der Variante 3

- Was passiert wenn ein Hindernis gar nicht das Auto ist (zum Beispiel ein Stuhl-Bein o. ä. Und das Auto **fälschlicherweise einlenkt**?

Variante 3: 3 Infrarot Sensoren

- Der HotRod bekommt in Front (über der „Motorhaube“ einen Infrarot Sensor.
- Der HotRod bekommt über den vorderen Räder einen Infrarot Sensor der in die jeweilige Richtung zeigt.
- Somit wird die Funktion von Folie 7 um die Kurven erweitert.
- Die beiden Sensoren zur Seite empfangen das kein Objekt in Sicht ist. Sobald der Abstand kleiner wird, hat sich das Auto in diese Richtung bewegt.
- Somit kann entsprechend eingelegt werden.

Follow-me

- Wie funktionieren moderne Autonome Fahrzeuge? Mit der Follow-me Funktion soll ein selbstfahrendes Fahrzeug simuliert werden, welcher genau dies darstellet.

Problematik der Variante 4

- Es gibt keine Probleme. Das Auto kann leicht einlenken, kann stark einlenken und gerade ausfahren.
- Doch es gibt nur 4 Steckplätze und diese (optimale) Lösung würde mindestens 5 Steckplätze benötigen.

Variante 4: 5 Infrarot Sensoren

- Der HotRod bekommt in Front (über der „Motorhaube“ einen Infrarot Sensor.
- Der HotRod bekommt über den vorderen Räder einen Infrarot Sensor der in die jeweilige Richtung zeigt.
- Ebenso werden Infrarot Sensoren Schräg zwischen den bisher angebrachten Sensoren angebracht.
- Die beiden Sensoren zur Seite empfangen das kein Objekt in Sicht ist. Sobald der Abstand kleiner wird, hat sich das Auto in diese Richtung bewegt.
- Somit kann entsprechend eingelegt werden.

Follow-me

- Wie funktionieren moderne Autonome Fahrzeuge? Mit der Follow-me Funktion soll ein selbstfahrendes Fahrzeug simuliert werden, welcher genau dies darstellet.

Problematik der Variante 5

- **Python unterstütz nur das Standardset** von Lego und nicht Hardware von dritten.
- Also wie Hi Technik Sensoren und Sender verwenden?

= Java verwenden

Variante 5: Hi Technik Sensor + Sender

- Der HotRod bekommt in Front (über der „Motorhaube“) einen Infrarot Sensor.
- Der vordere HotRod bekommt einen Anhänger mit einem Infrarot Ball, dieser sendet in alle Richtungen Infrarot Strahlen aus.
- Der hintere HotRod bekommt einen Hi Technik Infrarot Sucher. Dieser fängt Infrarot Strahlen ein und merkt sich den Eintrittswinkel.
- Dieser Eintrittswinkel wird von dem Lenkmotor angesteuert.

Follow-me

- Wie funktionieren moderne Autonome Fahrzeuge? Mit der Follow-me Funktion soll ein selbstfahrendes Fahrzeug simuliert werden, welcher genau dies darstellet.

Problematik der Variante 6

- Für diese Lösung (ideale Lösung) sind **minimal 5 Sensoren Steckplätze** von Nöten.
- Der EV3 Stein besitzt leider nur 4.
- Ein Port Multiplexer wird von Python hierbei nicht unterstütz.

Variante 6: 2 Farb- + 3 Infrarotsensoren

- Normale Fahrzeuge befinden sich auf Straßen mit markieren oder Bordsteinen und orientieren sich daran. Der Follow hält sich somit mit 2 Farbsensoren an einer Fahrbahnmarkierung.
- Ein weiterer Infrarot Sensor misst den Abstand nach vorne um eine Kollision zu vermeiden.
- Zwei weitere Infrarot Sensoren messen den Abstand zur Seite um eine Lenkung einzuleiten falls sie den Hot Rod wahrnehmen. Um fehlerhaftes Abbiegen zu vermeiden (Stuhlbein) darf nur abgebogen werden, wenn es die Fahrbahnmarkierungen zulassen.

Idee

- In der Realen Welt scannt ein Fahrzeug seine Umwelt bzw. die Objekte vor ihm auf Hindernisse und bremst Automatisch ab.

Umsetzung

- Das Lego EV3 Education Mindstorm Packet beinhaltet diverse Sensoren. Unter anderem auch Infrarot Sensoren.
- Ein Infrarot Sensor scannt den Abstand nach vorne (0 - 100, nah - fern).
- Dieser Abstand wird verarbeitet.

Quellcode

```
# Initialisiere den Infrarot Sensor
ir_sensor = InfraredSensor(Port.S4)

# Ermittle den Abstand nach vorne
distance_to_front = ir_sensor.distance()

# Verarbeite den Abstand
if distance_to_front < 50:

    # Fehlerbehandlung wie Alarm Signale
    troubleshooting_collision()

# Es ist kein Hindernis voraus und somit
# kann das Auto ohne Probleme fahren.
else:

    # Fahren
    drive()
```

Idee

- In der Realwelt hat ein Auto eine kleine Kamera in der Windschutzscheibe.
- Die Kamera oder auch dieser Sensor nimmt ein Objekt vor ihm war, und bildet den Abstand dazu. Dieser wird beibehalten.

Umsetzung

- **Python** unterstütz nur das Standardset von Lego. Somit ist es schwer eine solide Follow-me Funktion zu erstellen.
- **Java** unterstütz weitere Sensoren Typen, unter anderem die von Hi Technik.

Quellcode

```
public void followMe(HiTechnicIrSeekerV2 seeker) {
      // Ist der eintreffende Winkel kleiner 0
      if (seeker.getBeacon() < 0) {

            // So fahre eine Kurve
            driveCurve(speed, 0);

      // Ist der eintreffende Winkel größer 0
      } else if (seeker.getBeacon() > 0) {

            // so fahre eine Kurve
            driveCurve(0, speed);

      // Ist der eintreffende Winkel 0
      } else {

            // so fahre gerade aus
            driveForward(speed);
      }
}
```

Idee

- In der Realwelt hat ein Auto eine kleine Kamera in der Windschutzscheibe.
- Diese nimmt Farben oder sogar Ampel wahr. Je nach dem gemessen Farbwert, reagiert das Fahrzeug dementsprechend.

Umsetzung

- Das Lego EV3 Education Mindstorm Packet beinhaltet diverse Sensoren. Unter anderem auch Farbsensoren sowie eine Pixy Kamera.
- Im Beispiel wird der Standard Farbsensor von Lego verwendet.

Quellcode

```
# Initialisiere den Farb Sensor
c_sensor = ColorSensor(Port.S3)

# Ermittle die erkannte Farbe
color = c_sensor.color()

# Steht die Ampel auf Rot?
if color == Color.RED:
        # Bleibe stehen
        engine.stop()

# Es wird entweder keine Farbe erkannt oder
# die Ampel steht auf Grün bzw. Gelb.
# Beides steht für weiterfahren.
else:
        # Fahren
        drive()
```

Java

- Unterstützt deutlich weniger Library's.
- Läuft auf einer **Virtuellen Maschinen**. Somit ist das eigentliche Betriebssystem vorerst egal.
- Java's Library's können somit ohne Probleme auf dem EV3 Stein laufen.

Python

- Unterstütz **viele Library's**, da jeder eine erstellen und in den Packet Manager von Python hochladen kann.
- Diese sind nur für die gängigen Betriebssysteme verwendbar, je nach dem was sich der Entwickler gedacht hat.
- Das Miniatur Betriebssystem von dem EV3 Stein wird meistens nicht unterstütz.
- Funktionen wie eine Steuerung mit Xbox Controllern oder eine Car2Car Kommunikation mittels Bluetooth war somit nicht solide umsetzbar.

7. Quellcode - Hot Rod (Leader)

```
#!/usr/bin/env pybricks-micropython

from pybricks import ev3brick as brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor, InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import (Port, Stop, Direction, Button, Color, SoundFile, ImageFile, Align)
from pybricks.tools import print, wait, StopWatch
from pybricks.robotics import DriveBase

engine, wheels, infrared = Motor(Port.D), Motor(Port.A), InfraredSensor(Port.S4)
speed_of_engine, speed_of_wheels, time_of_wheels = 2000, 500, 250

def display(text, pos, img):
    brick.display.clear()
    brick.display.image(img)
    brick.display.text(text, pos)

brick.sound.beep(1500, 1000, 50)
brick.sound.file(SoundFile.HELLO)
display("HotRod", (60, 10), ImageFile.UP)

while True:
    distance_to_front = infrared.distance()
    if distance_to_front < 50:
        engine.stop()
        wheels.stop()
        brick.sound.file(SoundFile.ERROR_ALARM)
        brick.light(Color.RED)
        display("HotRod", (60, 10), ImageFile.KNOCKED_OUT)
    else:
        brick.light(Color.GREEN)
        display("HotRod", (60, 10), ImageFile.UP)
    pressed_key = infrared.buttons(1)
    for button in pressed_key:
        if button == 128:
            engine.run(speed_of_engine)
        elif button == 2:
            engine.stop()
        elif button == 512: #left
            wheels.run_time(speed_of_wheels, time_of_wheels)
        elif button == 8: #right
            wheels.run_time(-1 * speed_of_wheels, time_of_wheels)
        elif:
            engine.stop()
            wheels.stop()

    if brick.battery.voltage() < 7000:
        brick.sound.beep()
```

Fazit

- Der HotRod erfüllt seine Aufgabe gut. Er ist ein Ferngesteuertes Fahrzeug.
- Dem HotRod fehlt jedoch der Rückwärtsgang.
- Der kleine Freund sollte nicht dem Ball, sondern dem Fahrzeug folgen.
- Der kleine Freund wäre optimal hätte er einen Steckplatz mehr.