UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TECHNISCHE INFORMATIK

Übungsblatt 2

Arbitrierung von Reaktiven Verhalten

Vorlesung *Mobile Roboter*, Sommersemester 2022

Dozent: Prof. Dr.-Ing. Heiko Hamann
Ein Diskussionsforum zur Übung finden Sie unter: `https://moodle.uni-luebeck.de`.

In dieser Übung soll der Ihnen bereits aus der ersten Übung bekannte mobile Roboter Thymio II mit angeschlossenem Raspberry Pi (RPi) mit Python programmiert werden.

Unter Verwendung der horizontalen IR-Sensoren und der zum Boden gerichteten IR-Sensoren sollen drei reaktive Verhalten implementiert werden: Cruise, Avoid und Escape. Ein Arbiter soll diese drei Verhalten mittels Prorisierung zu einem Gesamtverhalten kombinieren, wobei Cruise die niedrigste und Escape die höchste Priorität haben soll.

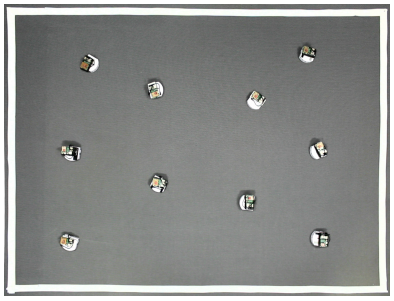

Abbildung 1: Arena mit Thymios

Der Roboter befindet sich dabei in einer Arena, die durch spiegelnde Linien begrenzt ist, vgl. Abbildung 1. Diese stellen virtuelle Wände dar. Der Roboter soll so lange geradeaus fahren (*Cruise*), bis er eine solche Wand durch die zum Boden gerichteten IR-Sensoren wahrnimmt. Dies entspricht einem Zusammenstoß mit der Wand, wodurch ein Fluchtverhalten (*Escape*) ausgelöst werden soll. Hindernissen, insbesondere anderen Robotern, soll der Roboter während seiner Fahrt durch die Arena ausweichen (*Avoid*).

Aufgabe 1 Vorbereitung zu Hause

Laden Sie sich die ZIP-Datei *Uebung2.zip* aus Moodle herunter und extrahieren Sie diese. Machen Sie sich mit dem Template *Ueb2_template.py* vertraut und nutzen Sie dieses, um Ihre Lösung zu implementieren.

Hinweise zum Aufbau des Programms

Sie sollen die drei reaktiven Verhalten Cruise, Avoid und Escape in separaten Klassen implementieren.

Jedes Verhalten soll die Variable *command* setzen. Diese spezifiziert in einem Array die Motorwerte für beide Differentialmotoren. Bei Bedarf können Sie weitere Variablen verwenden. Zudem soll jede Klasse zwei Methoden haben, vgl. Abbildung 2:

- *__init__*: Initialisierung der vom Verhalten benötigten Variablen, z.B. *command*.
- *action*: Setzt *command* auf die Soll-Motorwerte. Methode soll *True* zurückgeben, wenn das Verhalten ausgeführt werden soll und *False* andernfalls. So kann der Arbiter überprüfen, welche Verhalten gerade aktiv sind.

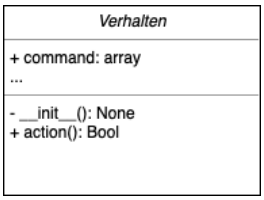

Abbildung 2: Beispiel des Aufbaus der Klasse eines Verhaltens

Beispielhaft ist im Template die Klasse Cruise vorgegeben, die allerdings noch vervollständigt werden muss. In der Implementierung soll Cruise die niedrigste Priorität haben und Escape die höchste.

Die Arbitrierung der Verhalten erfolgt in der Methode *Arbiter* durch einen kontinuierlichen loop (vgl. Template). Diese fragt durch die *action*-Methoden der einzelnen Klassen ab, welche Verhalten gerade aktiv sind und setzt die Motorwerte des aktiven Verhaltens mit der höchsten Priorität. Die einzelnen Verhalten dürfen dabei nicht blockieren (z.B. durch *time.sleep()*), da so die aktuellen Motorwerte nicht mehr vom Arbiter an die Motoren gesendet werden könnten!
Hinweis: Ballistische Verhalten müssen Sie so implementieren, dass diese sich intern ihren aktuellen Zustand merken können.

Die in Aseba verfügbaren Variablen finden Sie unter: `http://wiki.thymio.org/en:thymioapi`.

Implementierung

a) Der Roboter soll per Tastendruck gestartet und gestoppt werden können. Setzen Sie dazu die Variable *start* in der Methode *Arbiter* auf *True*, wenn der Knopf *button.forward* des Thymios gedrückt wird. Setzen Sie die Variable auf *False*, sobald der Knopf *button.center* gedrückt wird.

b) Die aktuellen Sensorwerte sollen in der Methode *Arbiter* abgerufen werden, wenn der Roboter gestartet wurde (*Start* ist *True*). Im Anschluss erfolgt die bereits vorgegebene Arbitrierung der Verhalten. Setzen Sie danach die Motorwerte des Thymios.

c) Implementieren Sie das Verhalten *Cruise* in der vorgegebenen Klasse. Es soll immer aktiv sein.

d) Implementieren Sie das Verhalten *Avoid* (Folie 2-31).

e) Implementieren Sie das Verhalten *Escape* (Folie 2-22).

f) Instantiieren Sie nun Objekte der einzelnen Verhalten und fügen Sie die Verhalten zum Array *behaviors* hinzu. Beachten Sie dabei die Reihenfolge, damit der Arbiter die Verhalten nach der vorgegebenen Priorität auswählt.

Testen des Codes

a) Öffnen Sie ein Terminal und starten Sie Webots:

```
webots
```

b) Öffnen Sie in Webots die Welt *Uebung2/worlds/Uebung_2.wbt*.

c) Öffnen Sie im Text Editor von Webots die Datei *Uebung2/controllers/thymio2_aseba_mr_1+2/main.cpp*. Drücken Sie nun *Shift+F7* (Clean) und im Anschluss *F7* (Build), um den Roboter-Controller *thymio2_aseba_mr_1+2* zu compilieren. Dieser Controller ermöglicht es Ihnen den gleichen Python-Code auf dem simulierten und dem echten Thymio ohne größere Anpassungen auszuführen.

d) Starten Sie die Simulation.

e) Öffnen Sie ein neues Terminal, navigieren Sie in den Ordner *Uebung2* und starten Sie Ihr Programm:

```
python Ueb2_template.py -s
```

f) Verbessern Sie den Python-Code bis das geforderte Verhalten ausgeführt wird.

Hinweise:

- Durch einen Doppelklick auf den Thymio öffnet sich sein *robot window*. Hier können Sensorwerte abgelesen und Knopfdrücke auf die fünf Knöpfe des Thymios simuliert werden. Ein geöffnetes *robot window* kann allerdings die Ausführungsgeschwindigkeit der Simulation deutlich verlangsamen. **Sie können zum Testen des Codes in der Simulation die Variable *start* dauerhaft auf *True* setzen.**
- Sie können die laufenden Programme und Prozesse im Terminal mit *Strg + C* beenden.

Aufgabe 2 Durchführung in der Übung

a) Starten Sie den Raspberry Pi und den Thymio, indem Sie diesen mit Strom versorgen. Verbinden Sie sich mit dem Raspberry Pi via SSH (Passwort: Thymio, HOSTNAME wurde in Übung 1 gesetzt)

```
ssh pi@HOSTNAME.local
```

b) Kopieren Sie Ihr Programm auf den Raspberry Pi, z.B. mit dem Befehl `scp`[1]. Testen Sie nun den Code auf dem Thymio.

```
python Ueb2_template.py
```

Sie können das Programm mit *Strg + C* beenden.

c) Schalten Sie nachdem Sie die Aufgaben erfolgreich bearbeitet haben den RPi mit dem Befehl

```
sudo shutdown now
```

aus. Warten Sie bis die gelbe LED aufhört zu blinken, um sicherzustellen, dass der RPi vollständig heruntergefahren ist. Entfernen Sie die SD-Karte und geben Sie diese bei den Betreuern ab.

[1] `https://www.raspberrypi.org/documentation/remote-access/ssh/scp.md`