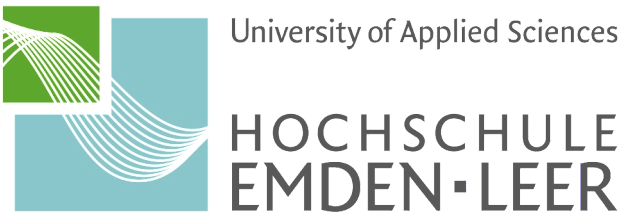
Parallele und verteilte Systeme

Nebenläufige Programme in Go



Gruppenmitglieder: Johannes Theiner
Semester: Sommersemester 2020
letzte Änderung: 20. Januar 2022

Zuerst werden die Datentypen *Colour* und *CardinalDirection* als Enum angelegt, in Auflistung 1 ist beispielhaft *Colour* definiert.

```
1
```

Auflistung 1: Definition Datentyp Colour

In der Routine einer Apmel wird mit der Gegenrichtung synchronisiert bevor die angezeigte Farbe geändert wird und bevor ein Richtungswechsel initiiert wird (Auflistung 2).

```
1
```

Auflistung 2: Implementation TrafficLight

Die *sync* Funktion definiert die Synchronisation der gegenüberliegenden Ampeln (Auflistung 3). Hierfür wird entweder eine Nachricht auf dem Kanal für die Achse gesendet oder empfangen (Auflistung 3).

```
1
```

Auflistung 3: Implementation Synchronisierung

In der *handover* Funktion wird die Übergabe zwischen nebeneinander liegenden Ampeln definiert. Dazu wird eine Nachricht an die rechts/links liegende Ampel gesendet, die bei Empfang dieser, die eigene Routine fortführt. Erst wenn eine solche Nachricht wieder von der anderen Seite empfangen wird fährt die Routine fort (Auflistung **??**).

```
1
```

Auflistung 4: Implementation der Übergabe

Zur Kommunikation zwischen den Ampeln werden mehrere Channel verwendet. Je einer pro Achse und zwei zur Kommunikation zwischen nebeneinander liegenden Ampeln.

Zusätzlich werden zwei Funktionen *getAxisChannel* und *getNextChannel* definiert die die jeweils zugehöhrigen Channel zurückgeben. (Auflistung 5)

```
1
```

Auflistung 5: Definition der Channel

Schlussendlich wird das System wie in Auflistung reflst:main gestartet.

```
1
```

Auflistung 6: Hauptfunktion

Ein Durchlauf des Systems ist in Auflistung 7 zu sehen.

```
1
```

Auflistung 7: Beispielhafte Konsolenausgabe