Projektaufgabe zu „Wissensverarbeitung“, WS17/18, IIOb15

Georg Ringwelski

Stand: 7.9.18

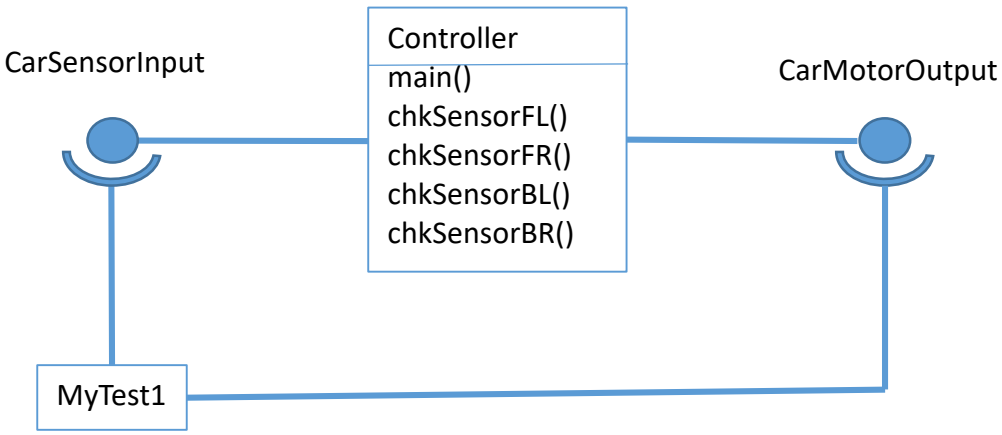
Die Projektaufgabe wird semesterbegleitend durchgeführt und ist Gegenstand der Prüfungsleistung. **Die Prüfungsleistung (PL) besteht aus 4 Teilprüfungen**, in denen jeweils 5 bzw. 10 Punkte zu erreichen sind. Die Benotung des Moduls errechnet sich aus der Summe der im Semester gesammelten Punkte nach dem üblichen Schlüssel (>=50% → bestanden etc). **Die u.g. Termine zur Präsentation der Teilleistungen sind verbindlich**, verspätete Präsentationen werden nur bei Vorlage einer ärztlichen Bescheinigung akzeptiert. **Alle Leistungen werden in Einzelarbeit erbracht**, bei der Präsentation von Plagiaten bekommen beide Präsentierenden keine Punkte.

Im Projekt entwickeln Sie die Software für ein autonom fahrendes Auto, das chaotisch aber kollisionsfrei gemeinsam mit anderen Fahrzeugen im Roboterlabor herumfährt. Zunächst entwickeln Sie die Software mit Simulator am Rechner. Sie erstellen also als Eingabe alternativ zu den Lego Sensoren eine GUI und mehrere Testprogramme. Als Ausgabe erstellen Sie alternativ zu den Lego Motoren ebenfalls eine GUI. Beim Deployment auf den NXT und EV3 werden dann die Motoren als Ausgabeeinheiten verwendet und die Sensoren oder die Testprogramme (hardware in the loop) zur Eingabe.

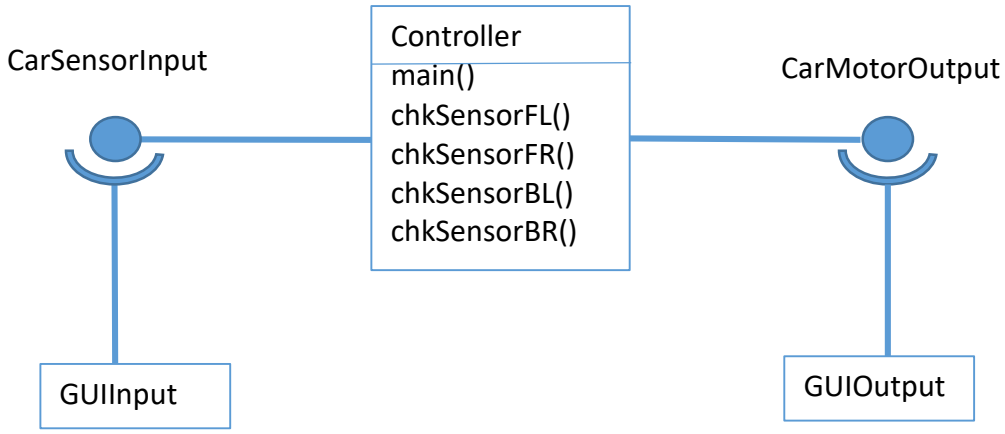
Aufgaben

1. (Bis Mitte November, ist nicht Bestandteil der Prüfungsleistung) Bauen Sie jeweils zu zweit mit Lego Technik ein Fahrzeug nach dieser Vorlage:
   http://www.nxtprograms.com/NXT2/race_car/
   Lassen Sie dabei den Farbsensor und die Fernsteuerung weg und bauen statt dessen Ultraschallsensoren an. Zusätzlich können Sie auch einen Touch-Sensor anbauen. Die nötigen Teile dafür stellt Ihnen Herr Fiebiger zur Verfügung. Überlegen Sie, welche Sensoren Sie an welcher Stelle sinnvoll anbringen könnten. Diskutieren Sie das auch in der Gruppe.

2. (bis KW 43, 5 Punkte für PL) Entwickeln Sie in Java ein nebenläufiges Programm „Controller“, das alle 500ms die u.g 4 Methode **asynchron** aufruft. In den Methoden lesen Sie Daten über die Schnittstelle CarSensorInput ein und rufen jeweils eine (zunächst beliebige) Methode der Schnittstelle CarMotorOutput auf. Um auszuprobieren, ob das funktioniert realisieren Sie eine Klasse, die beide Schnittstellen implementiert und einige Unit Tests durchführt. Die Schnittstellen können Sie von der website zur LV runterladen.

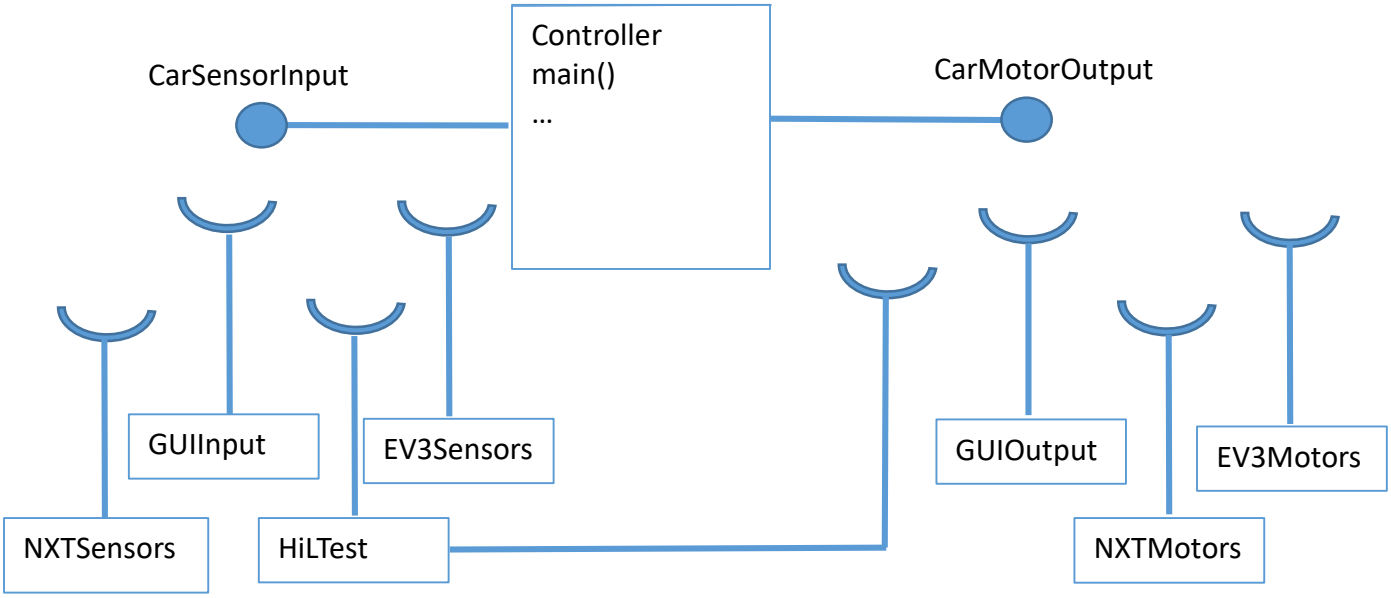


3. (bis KW46, 5 Punkte für PL) Entwickeln Sie jeweils eine GUI für die Ein- und Ausgabe, die die entsprechende Schnittstelle implementiert. In der Eingabe GUI soll man Ereignisse, wenn also ein Sensor ein Objekt erfasst beliebig asynchron erzeugen können. An der Ausgabe GUI soll dann immer ablesbar sein, wie das Fahrzeug reagiert. Implementieren Sie im Controller einige erste Regeln für kollisionsfreies Fahren, so dass es im GUI sichtbar funktioniert. Achten Sie dabei auf gute Wartbarkeit Ihres codes, später werden Sie die Regeln voraussichtlich verändern müssen.



4. (bis KW 49, 5 Punkte für PL) Implementieren Sie eine weitere Ein- und Ausgabeschnittstelle „HiLTest“ , in der bestimmte Sequenzen von Ereignissen (und Pausen) als Tests ausgeführt werden können. Analog zu Aufg. 2 soll diese Klasse auch die Ausgabeschnittstelle implementieren, um automatische Testfälle zu realisieren. Realisieren Sie darin mindestens 2 Testsequenzen (je ca. 10-20 Sekunden) für den Betrieb mit „hardware in the loop“ (HiL). Implementieren Sie dann Klassen zu beiden Schnittstellen, die die jeweiligen Harwareelemente (Sensoren und Motoren) der beiden Hardware Platformen NXT und EV3 ansteuern.
   Realisieren Sie schließlich eine Möglichkeit, ihr System zu konfigurieren. D.h. die konkrete Klasse, die zur Ein- bzw. Ausgabe verwendet werden soll wird in einem entsprechenden build-Prozess festgelegt. Vermeiden Sie dabei einerseits duplizierten Code und andererseits die Einbindung unnötiger Bibliotheken (zB das GUI ist im NXT unnötig). Behalten Sie den Überblick und stellen Sie sicher, dass Sie jederzeit jede Konfiguration in der aktuellen Version ausführen können. Ich empfehle daher den Einsatz eines build-Werkzeuges wie Apache Ant, Apache Ivy oder Gradle. Das System soll (immer) in folgenden sechs Konfigurationen funktionieren:

	Eingabe	**Ausgabe**
1	GUI	GUI
2	HiL	HiL
3	HiL	GUI
4	HiL	NXT
5	HiL	EV3
6	EV3	EV3
67	NXT	NXT



5. (bis KW51, 10 Punkte für PL) Implementieren Sie nun die Methoden chkSensorXY() so, dass die Fahrzeuge kollisionsfrei fahren. Dabei sollen sie möglichst immer in Bewegung bleiben und erst dann lenken oder anhalten, wenn ein Hindernis die Weiterfahrt verhindert. Wenn kein Hindernis im Weg ist sollten die Fahrzeuge ca. 30cm/sec schnell fahren. Durch die Asynchronität der Abfragen bzw. der Ereignisse sind hier im Controller Algorithmen zu realisieren, die geeignete Reaktionen (d.h. Steuerbefehle an die Ausgabeeinheit) abhängig von der aktuellen Situation ausführen. Zudem kann es zu Verklemmungen oder liefelocks kommen, was beides zu verhindern ist.
    a. Modellieren Sie Ihr System zunächst als Zustandsdiagramm, in dem Sie die Zustände und Zustandsübergänge aufzeichnen, die Sie im Code dann durch Attributswerte und Bedingungen implementieren.
    b. Testen Sie Ihre Implementierung am besten zunächst mit HILTests/GUIInput und dem GUIOutput. Gehen Sie dann aber zur echten Hardware über, weil diese sich evtl. etwas anders verhält als Ihre Softwareimplementierungen zur Simulation.
    c. Probieren Sie die Funktion Ihrer Steuerung aus, indem Sie Ihr Fahrzeug ohne weitere Hindernisse auf der Platte fahren lassen. Es sollte zügig fahren und nicht gegen die Begrenzungen stoßen.
    d. Probieren Sie dann den Betrieb gleichzeitig mit anderen Fahrzeugen oder anderen beweglichen Hindernissen. Es sollte zu keinen Zusammenstößen kommen. Im Zweifelsfall sind die bekannten deutschen Verkehrsregeln anzuwenden (rechts vor links, Sicherheitsabstand einhalten…).