HAUPTSEMINAR AUTOMATISIERUNGS-, MESS- UND REGELUNGSTECHNIK: ALLGEMEINE ANLEITUNG UND AUFGABENSTELLUNG

"ROBOTIKPROJEKT AUTONOMES EINPARKEN"

VERSION 2.8

INHALT

1 SEMINARBESCHREIBUNG

Das Modul umfasst inhaltlich Themen und Fragestellungen der Automatisierungs-, Mess- und Regelungstechnik und die Methodik wissenschaftlicher und projektbasierter Arbeitsweise.

1.1 QUALIFIKATIONSZIELE

Nach Abschluss des Moduls sind die Studierenden in der Lage, ihre Kenntnisse, Fähigkeiten und Fertigkeiten selbständig, vorzugsweise im Team auf eine konkrete Aufgabenstellung anzuwenden, die Arbeitsschritte nachvollziehbar zu dokumentieren sowie die Ergebnisse zu präsentieren und zur Diskussion zu stellen.

1.2 LEISTUNGSPUNKTE UND NOTEN

Gemäß der Studienordnung werden für dieses Modul 4 Leistungspunkte vergeben. Die Modulnote wird aus zwei Teilleistungen, einer Projektdokumentation und einem Kolloquium, gebildet. Dabei geht die Projektdokumentation zu zwei Dritteln und das Kolloquium mit einem Drittel in die Gesamtnote ein.

1.3 ZIEL DER SEMINARGRUPPENARBEIT

In Anlehnung an heutzutage entwickelte autonome Autos und Fahrerassistenzsysteme soll im Hauptseminar die Funktionalität eines erweiterten Parkassistenten nachgebildet werden.

Ziel ist es, dass ein Fahrzeug (mobiler Roboter) dem Straßenverlauf eines vorgegebenen Parcours (s. Abschnitt 4) folgt, passende Parklücken erkennen, vermessen, bewerten kann und Parkvorgänge eigenständig durchführt. Der Bediener soll dabei über einen Tablet-Computer als Eingabegerät verschiedene Betriebsmodi des Fahrzeuges steuern und vom Roboter gesammelte Informationen abfragen können.

1.4 DETAILS ZUR GRUPPENARBEIT

Jeweils fünf Studierende arbeiten im Team an der Aufgabe. Jeder Student muss sich dabei auf ein Aufgabengebiet (Modul) spezialisieren. Es findet eine gemeinsame Betreuung durch die vier AMR-Lehrstühle statt, wobei jedes Aufgabengebiet schwerpunktmäßig von einem Lehrstuhl betreut wird (s.u.). Es werden spezielle Projektanleitungen für alle Module zur Verfügung gestellt, die die vorliegende Anleitung und Aufgabenstellung ergänzen.

Die Aufgabenbereiche/Module sind:

- Perception (Sensorik) -> betreut von MST,
- Guidance (Missionsplanung, Pfadgenerieren) -> betreut von AT und RST,
- Navigation (Posenbestimmung, Parklückendetektion) -> betreut von AT,
- Regelungstechnik (Control) (z.B. Bahnregelung) -> betreut von RST,
- Bedienschnittstelle (HMI) (z.B. Parklückenauswahl) -> betreut von PLT.

Zum gegenseitigen Austausch, innerhalb der Gruppe, mit den anderen Gruppen und mit den Betreuern soll die Lernplattform OPAL genutzt werden. Details dazu und generell zum Ablauf und der Organisation des Hauptseminars sind in Abschnitt 5 zu finden.

2 AUFGABENSTELLUNG

2.1 FUNKTIONALE ANFORDERUNG

Über eine Mensch-Maschine-Schnittstelle in Form eines 7 Zoll Android Tablets soll das Verhalten des Roboters gesteuert werden können. Auf dem Display des Tablets soll der Parcours aus der Vogelperspektive erscheinen. Über die Schnittstelle sind alle durch die Sensorik erfassten Messwerte auszugeben und auf einer Karte die aktuelle Position des Roboters sowie die gefunden Parklücken verzeichnet werden.

Die im Folgenden aufgeführten Funktionalitäten des Roboters sollen implementiert werden.

Parklückensuche (Scout Modus)

Bei der Parklückensuche soll der Roboter den Parcours entlangfahren und dabei nach Parklücken suchen. Die erkannten Parklücken sollen *vermessen* und danach *bewertet* werden, ob sie für einen Einparkvorgang geeignet sind. Jede Parklücke soll mittels einer ID eindeutig zuordenbar sein und dem Benutzer über die Mensch-Maschine Schnittstelle in einer Karte angezeigt werden. Damit der Roboter dem Parcours folgen kann, ist ein Regelalgorithmus zur Linienverfolgung zu entwickeln.

Gezieltes Einparken (Park This Modus)

Der Benutzer soll auf dem Tablet detektierte Parklücken aus der Karte für einen Einparkvorgang auswählen können. Sobald eine Parklücke ausgewählt wurde, soll der Roboter diese autonom über den Parcours anfahren und dort einparken. Um ohne Kollision und unter Einhaltung einer vorgegebenen Endposition in einer Parklücke parken zu können, muss der Einparkvorgang dabei mittels einer Bahnfolgeregelung umgesetzt werden. Das Guidance Modul berechnet die Bahn (Pfad).

Ausparken

Sobald der Roboter korrekt in einer Parklücke geparkt hat, soll er dort verbleiben und warten, bis der Benutzer über das Tablet ein Signal zum Ausparken gibt. Der Roboter muss daraufhin die Parklücke kollisionsfrei verlassen (bevorzugt auf dem gleichen Pfad wie beim Einparken) und die Führungslinie des Parcours wiederfinden. Sobald die Linie gefunden ist, soll automatisch der Scout Modus aktiviert werden.

Anhalten (Pause Modus)

Der Benutzer soll den Roboter außerdem unabhängig von der aktuell ausgeführten Aufgabe in einen Pause-Zustand versetzten können, in welchem der Roboter ruhen soll.

Hinweise

Welcher Bewegungsablauf programmiert wird um die oben genannten Aufgaben zu erfüllen ist selbständig zu erarbeiten. Rückwärtsfahren kann sinnvoll sein um eine Aufgabe effizient zu erfüllen, ist aber schwieriger zu implementierten. Falls möglich, sollen die oben genannten Funktionen möglichst schnell und mit wenig Fahrstrecke realisiert werden.

2.2 ANFORDERUNGEN AN GRUPPENARBEIT

Zur Realisierung der dargestellten Funktionalitäten sind in jedem der fünf Aufgabenbereiche modulspezifische Aufgaben zu bewerkstelligen. Eine detaillierte Beschreibung dieser Aufgaben sowie Details und Hinweise zu den Lösungswegen befindet sich in den speziellen Modulanleitungen, die ebenfalls zur Verfügung gestellt werden.

Achtung: Das Projektmanagement und das Design und der Aufbau des Fahrzeuges ist nicht Teil einer Modulaufgabenstellung und muss von der gesamten Gruppe erledigt werden. Beachten Sie dabei, dass das Design des Roboters maßgeblichen Einfluss auf die zu entwickelnden Algorithmen haben kann, sodass Sie sich zeitnah auf eine Variante einigen sollten.

Achtung: Der Zeitbedarf für die Systemintegration (das Vereinen der einzeln programmierten Module) wurde im vergangenen Jahr sehr unterschätzt! Es sollten mindestens 50% der Arbeitszeit für Systemintegration, Fehlersuche und iterierende Modulnachbesserungen eingeplant werden!

- Je besser die Zusammenarbeit und die Schnittstellen vorher abgesprochen und geplant sind, desto weniger zeitaufwendige Abspracheprobleme müssen Sie hinterher lösen.
- Oft wird beim Testen des Gesamtsystems festgestellt, dass etwas nicht funktioniert. Allein herauszufinden in welchem Modul der Fehler liegt bzw. ob es ein Abspracheproblem ist, kann sehr lange dauern. Unter Umständen ist es nach Finden des Fehlers/Abspracheproblems notwendig noch einmal erhebliche Entwicklungsarbeit und mehrere Test/Nachbesserungsiterationen in die fehlerfreie Implementierung zu stecken. Kalkulieren Sie dies bitte bei Ihrer Zeitplanung mit ein!
- Manche Softwareteile lassen sich erst testen, wenn andere bereits fehlerfrei funktionieren. Wenn die Perception erst ein paar Wochen vor Abgabe richtig funktioniert, dann bleibt kaum Zeit mehr zum Testen der Navigation und die Guidance (die beide Module benötigt) kann womöglich gar nicht mehr getestet werden.
  Planen Sie daher sorgfältig welche Funktion von welchen anderen Funktionen abhängt und setzen sie strenge Deadlines für die Fertigstellung der Funktionen auf denen andere Funktionen aufbauen!

Im eigenen Interesse sollte jede Gruppe einen Projektmanager ernennen, der im Blick behält, dass Meilensteine und Deadlines eingehalten werden. Es bietet sich an, dass diese Aufgabe anfänglich jemand übernimmt, der am Anfang ohne die Vorarbeit der anderen Gruppenmitglieder nicht so viel testen kann. Im späteren Verlauf kann die Aufgabe dann auch an andere Gruppenmitglieder übergeben werden, um den ersten Projektmanager zu entlasten. Anbei findet sich eine grobe Übersicht. Diese dient als Orientierung für die zeitliche Strukturierung der Gruppenarbeit.

	Oktober	November	Dezember	Januar	Februar	Legende
Projektmanagement						Perception
Einarbeitung						Guidance
Aufbau Roboter						Navigation
Sensorkalibrierung						Control
Sensoroptimierung						HMI
Missionsplaner						Alle
Pfadgenerator						
Lokalisierung						
Parklückendetektion						
Linienerkennung, -Verfolgung						
Kinematik						
Trajektorie						
Zielpose						
Beobachtung						
Bedienung						
Dokumentation + Verteidigung						

TABELLE 1: EMPFOHLENE ZEITLICHE STRUKTURIERUNG DER GRUPPENARBEIT

3 Ausgehändigte Hardware

3.1 Hardwarestruktur

Die ausgegebene Hardware besteht aus drei Teilen. Einem LEGO Mindstorms NXT Set inklusive verschiedener NXT-Sensoren zum Bau des mobilen Roboters, einem Tablet zur Bedienung von diesem und ein Arduino Uno Mirkocontrollerboard zum Auslesen der beigelegten Nicht-NXT-Sensoren (Perzeptionsteile). Die Struktur der Hardware und die Anbindung an einen PC ist Abbildung 1: Hardwarestruktur dargestellt.

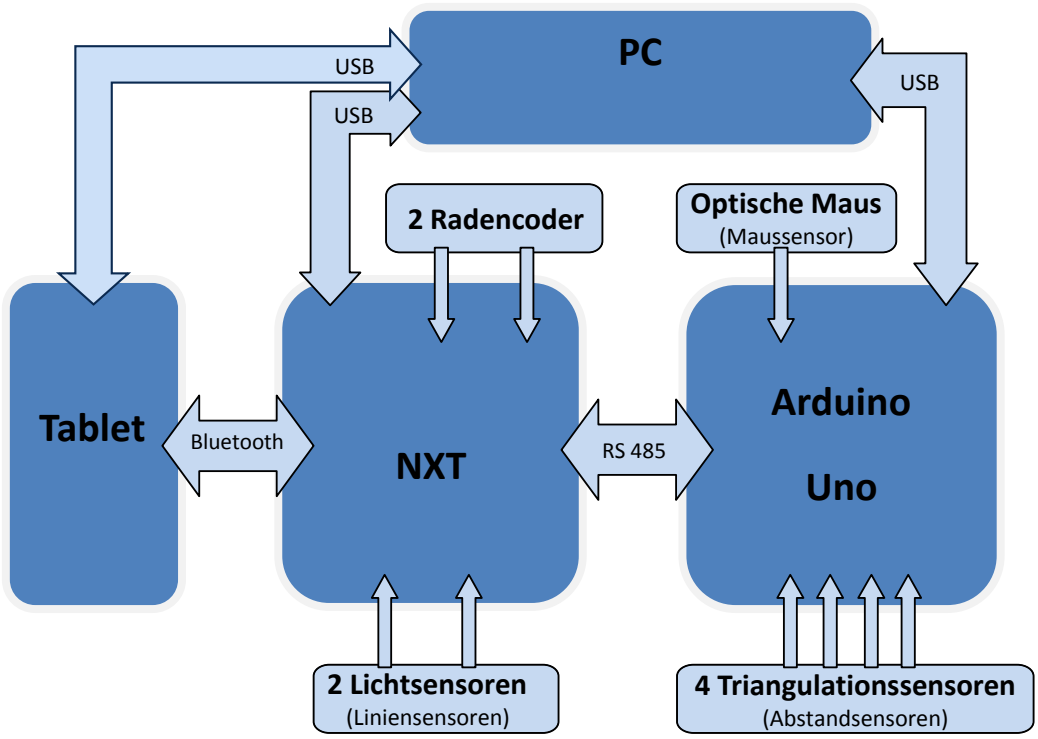

Abbildung 1: Hardwarestruktur

3.2 7 Zoll Android Tablet

Für die zu implementierende Mensch-Maschine-Schnittstelle (HMI) wird ein 7 Zoll Android Tablet inklusive Zubehör bereitgestellt (Samsung Galaxy Tab 2 7.0 P3100). Mittels Bluetooth kann mit dem LEGO Mindstorms NXT Roboter kommuniziert werden.

3.3 LEGO Mindstorms NXT Roboterset

Das LEGO Mindstorms NXT Roboterset besteht hauptsächlich aus dem "LEGO 9797: Mindstorms Education Basis Set". Im Set enthalten sind:

- 1 NXT Baustein
- 1 wiederaufladbarer Akku
- 3 NXT Servo-Motoren
- 2 NXT Berührungssensoren
- 1 NXT Lichtsensor
- 1 NXT Geräuschsensor
- 1 NXT Ultraschallsensor
- 1 USB Datenkabel
- 7 Verbindungskabel (davon 1x Arduino-Kabel, mit 3-pin Stecker)
- 3 Adapterkabel für Motoren/Sensoren des LEGO Mindstorms RCX-Systems
- Bauanleitung
- Teileliste und Sortierübersichten
- stabile Kunststoffaufbewahrungsbox
- verschiedenste LEGO Bausteine gemäß Teileliste

Dem hinzugefügt wurden noch einige weitere Teile:

- 1 LEGO 9844: NXT Lichtsensor
- 1 LEGO 8887: Transformator (10V DC)
- verschiedene zusätzliche LEGO Bausteine (auf Teileliste ergänzt und einer beigefügten Liste von zusätzlichen LEGO-Bauteilen notiert)

Auf den NXT-CPU-Baustein wurde als Firmware die Java Virtual Machine LeJOS NXJ aufgespielt. Damit kann der Baustein dafür geschriebene Java-Programme ausführen. Weitere Hinweise dazu finden Sie in den Abschnitten 7 und 9.

3.4 Optische Maus (Maussensor)

Es wird eine gewöhnliche optische Computermaus bereitgestellt. Der Korrelationssensor der Maus soll aus der Maus ausgebaut und als Sensor im Roboter eingebaut werden. Er muss dafür innerhalb des Perception-Moduls kalibriert, hinsichtlich seiner Messunsicherheit bewertet und optimiert werden und soll anschließend über einen Mikrokontroller an den NXT angeschlossen werden. Die Messdaten der Maus sollen zur lokalen Navigation genutzt werden.

3.5 Triangulationssensoren (Abstandssensoren)

Jede Gruppe erhält vier Triangulationssensoren von Typ Sharp GP2Y0A41SK0F. Mit ihnen kann man Abstände von 4 cm bis 30 cm messen und somit den Einparkvorgang unterstützen, Kollisionen vermeiden oder die Navigation verbessern. Auch diese müssen kalibriert und hinsichtlich ihrer Messunsicherheit im Modul Perception untersucht werden.

3.6 Arduino Uno Mikrocontrollerboard

Die Daten des Maussensors und der Triangulationssensoren müssen schnell und zuverlässig aufgenommen und weiterverarbeitet werden. Da der NXT nicht ausreichend passende Schnittstellen bietet und um Rechenleistung auszulagern, sollen die Sensordaten von einem externen Arduino Uno Mikrokontroller (µC) aufgenommen, weiterverarbeitet und dem NXT über eine Serielle RS485 Schnittstelle bereitgestellt werden. Zusätzlich zum Arduino (http://arduino.cc/en/Main/ArduinoBoardUno) inkl. Arduino Shield (vgl. http://arduino.cc/en/Main/ArduinoProtoShield) werden notwendige Verbindungskabel, Batterie und Hausung bereitgestellt.

4 Parcours

Der Parcours besteht aus einem bedruckten Papieruntergrund, zum Transport zerlegbaren Banden und "parkenden" Hindernissen. Neben dem Parcours, der zur Erfüllung der Seminaraufgabe verwendet werden muss, werden weitere Übungsparcours ausgegeben werden, die aber nur zum anfänglichen Testen zu Hause gedacht sind!

4.1 Papierhintergrund

Der Papieruntergrund ist in Graustufen bedruckt. Die Linie, die der Roboter entlang fahren soll, ist schwarz. Der Bereich rechts und links neben der Linie, die "Straße", ist weiß. Alle sonstigen Bereiche sind grau, auch die Bereiche in denen der Roboter einparken soll.

In Abbildung 2 ist der Parcours näher spezifiziert und bemaßt. Farbige Elemente in der Abbildung dienen nur dem Verständnis und sind nicht mit aufgedruckt. Bei dieser Abbildung handelt es sich nur um einen Ausschnitt, der wahre Parcours ist etwas breiter (s. Abbildung 3).

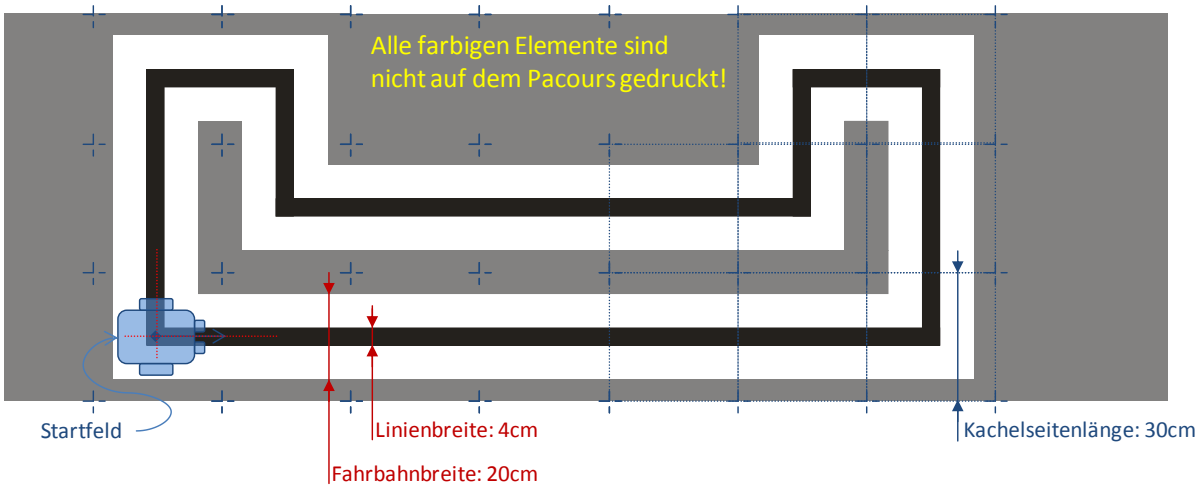

Abbildung 2: Parcoursbemaßung

Die Geometrie lässt sich in logische Einheiten unterteilen. Jede Kachel hat eine Seitenlänge von 30cm und die zu verfolgende Linie befindet sich genau mittig auf der Kachel und auf der "Straße".

4.2 Startfeld und Fahrtrichtung

Der Roboter wird zu Anfang immer mit seinem Mittelpunkt auf dem Startpunkt gestartet und mit seiner Blickachse entlang des längsten Linienabschnitts ausgerichtet. Der Mittelpunkt des Roboters ist dabei durch die geometrische Mitte der Antriebsachse definiert. Der Startpunkt befindet sich genau in der geometrischen Mitte der Startkachel.

Die Fahrtrichtung ist entgegen dem Uhrzeigersinn.

4.3 Banden

Die Banden sind in Abbildung 3 rot eingezeichnet. Sie lassen sich aus einzelnen Stücken mit Hilfe von Klettverschluss aneinander befestigen. Der nominale Abstand zu einem Kachelrand beträgt 5cm. Durch Fertigungsungenauigkeiten kann dieser Abstand allerdings variieren. Es gibt weiterhin vier Ausnahmen: An zwei Stellen befindet sich die Bande direkt auf dem Kachelrand und an weiteren zwei Stellen in Kachelmitte.

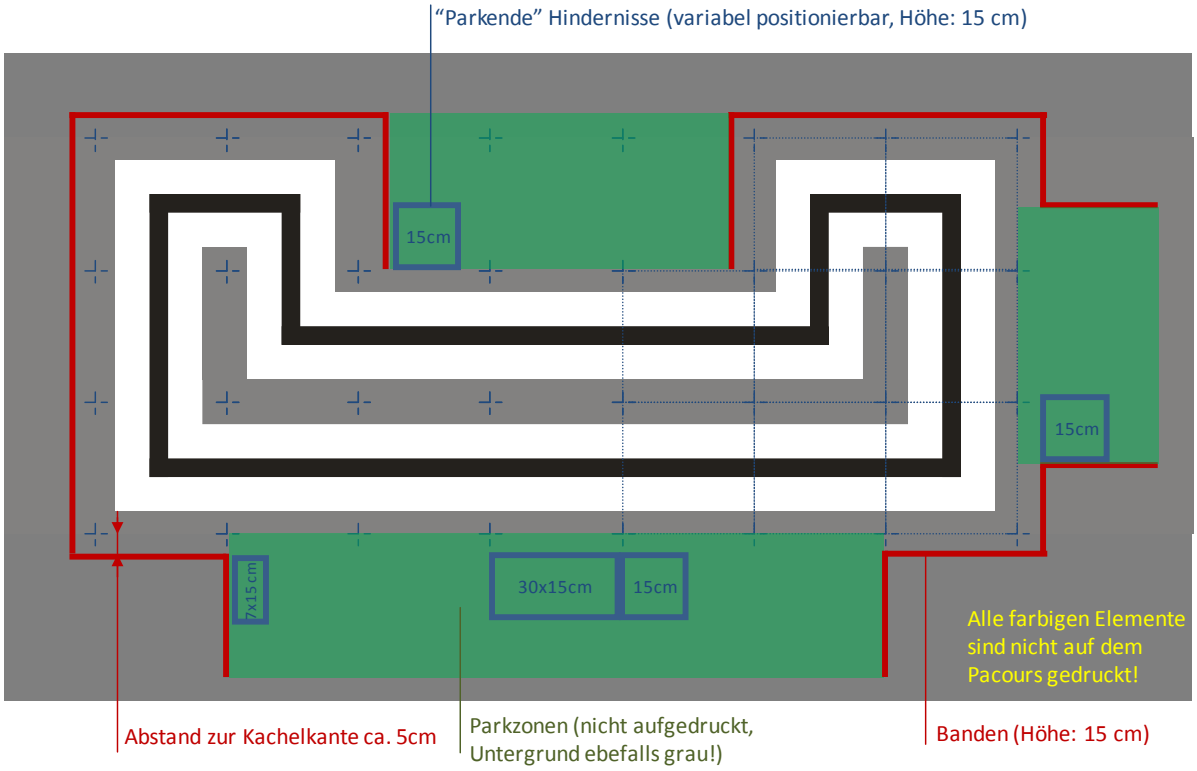

ABBILDUNG 3: BANDEN UND PARKBUCHTEN

4.4 PARKZONEN UND HINDERNISSE

Die Parkzonen, also die Bereiche in denen die "Straße" nicht von Banden begrenzt wird, sind in Abbildung 3 grün eingezeichnet. Der Untergrund ist in Wirklichkeit grau. Hier befinden sich potentielle Parklücken, aber es werden auch Hindernisse, die parkende Autos simulieren, aufgestellt.

Es gibt Hindernisse in drei Breiten, 7,5 cm, 15 cm und 30 cm. Diese können beliebig miteinander kombiniert werden. Sie sollen allerdings immer so aufgestellt werden, dass die Fläche, die zur "Straße" zeigt, parallel zum Straßenrand ausgerichtet ist und den gleichen Abstand zum Straßenrand hat wie die Bande, die die Parkzone begrenzt.

Eine Parklücke sollte mindestens 45 cm lang sein, wenn sie zum Einparken geeignet sein soll. Falls der Roboter eine kleinere Parklücke als geeignet definiert, muss er zeigen, dass er auch darin einparken kann.

Die Form des Parcours führt dazu, dass nicht alle Parklücken gleich einfach zu benutzen sind. Parklücken die z.B. hinter einer Kurve liegen, müssen vermessen werden, solange der Roboter noch nicht parallel zur Linie ausgerichtet ist und auch der Einparkvorgang kann unter Umständen nicht ohne Bewegungseinschränkungen oder Extramanöver durchgeführt werden.

Bei optimaler Lösung der Aufgabe sollte der Roboter in jede Parklücke einparken können, die größer oder gleich 45 cm lang ist und innerhalb einer Parkzone liegt.

5 NXT-Roboter

Es gibt keine verbindliche Vorgabe zum Aufbau des Roboters. Ausschlaggebend ist aber, dass der Roboter die Aufgaben erfüllen kann, ohne mit der Bande/den Hindernissen zu kollidieren!

Folgende Größe ist ein sinnvoller Vorschlag (siehe auch Abbildung 4: Spezifikation Robotergröße):

- max. Länge: 30cm
- max. Breite: 20cm
- max. Höhe: 20cm
- max. Drehradius: 15cm

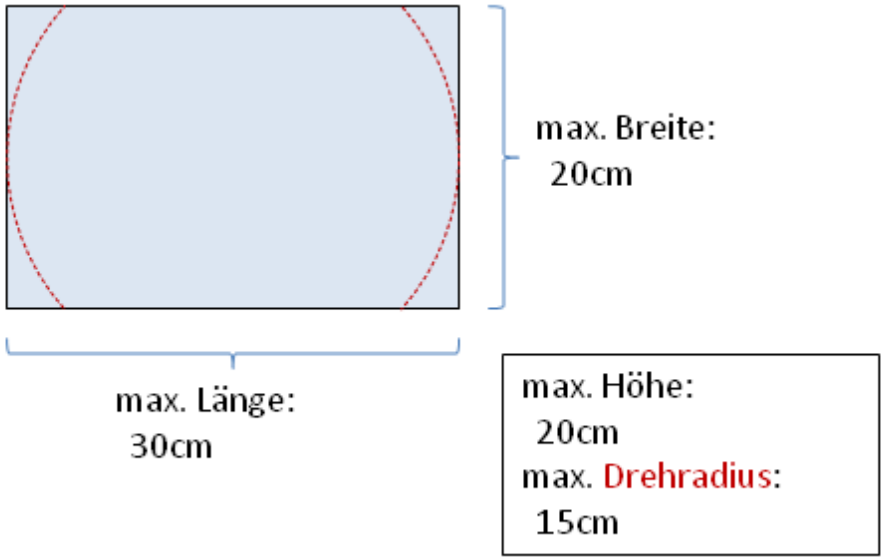

Abbildung 4: Spezifikation Robotergröße

6 Organisation

6.1 Einführungsveranstaltungen

Auf der Einführungsveranstaltung werden die einzelnen Module vorgestellt. Ziele, Ablauf und Aufgaben des Seminars werden erläutert.

Während der 1. Konsultation wird die notwendige Hardware ausgehändigt (siehe dazu Abschnitt 6.5), Schlüssel für Schließfächer ausgeliehen und die Einführungsveranstaltungen für die einzelnen Module angeboten.

6.2 Hinweise zum Ablauf des Seminars

Die gestellten Aufgaben sind selbständig zu bearbeiten. Die beteiligten Professuren bieten dazu eine wöchentliche Konsultation an. Die Termine dafür sind mit dem jeweiligen Betreuer des betreffenden Lehrstuhls abzustimmen. Des Weiteren steht ein Referenz-LEGO-Modell (ohne Perzeptionsteile) bereit, an dem sich die eigene Roboterkonstruktion orientieren sollte. Es wird im Rahmen des Seminars ein Raum zur Verfügung stehen, in dem der eigene Roboter auf einem aufgebauten Parcours getestet werden kann, jede Gruppe kann außerdem einen kleinen Übungsparcours für Zuhause erhalten.
Falls gewünscht, kann bei den Konsultationen beim Betreuer des AT-Lehrstuhls ein Schlüssel für ein Schließfach ausgeliehen werden, in dem der Roboter eingeschlossen werden kann.

6.3 Austauschplattform und Kommunikation

6.3.1 Allgemeines

Zum Austausch zwischen Betreuern und Gruppen soll die Lernplattform OPAL sowie eine Mailingliste verwendet werden.

Als Austauschplattform für dieses Hauptseminar dient ein OPAL-Kurs. Die URL zur Seite des OPAL-Kurses lautet: https://bildungsportal.sachsen.de/opal/url/RepositoryEntry/7404093447

Jeder Teilnehmer der noch keinen OPAL-Account (ZIH-Account) besitzt, muss sich zunächst registrieren.

6.3.2 Hinweise zur Struktur und Nutzung der Austauschplattform

6.3.3 Auf der Hauptseite des OPAL-Kurses finden sich verschiedene thematische Abschnitte. Im ersten Abschnitt werden wichtige Links, wichtige Anleitungen, Quellcode und weitere Informationen zum Seminar von den Betreuern veröffentlicht. Allgemeine Kommunikation und Mailingliste

Erfahrungsgemäß etablieren die Gruppen für die interne Kommunkation eigene Kanäle. Empfehlenswert ist der Austausch von E-Mailadressen und Telefonnummern. Für Kommunikationsbedürfnisse, die auch für Mitglieder aus anderen Gruppen relevant sind, wird eine Mailingliste (hsmar-info@groups.tu-dresden.de) zur Verfügung gestellt. Bitte nutzten sie diesen Informationskanal verantwortungsvoll und themenbezogen. Schreiben Sie jeweils einen aussagekräftigen Betreff.

Gruppenübergreifender Austausch von nützlichen Informationen, Hinweisen oder Anleitungen ist durchaus gerne gesehen. Beachten Sie aber bitte, dass das Seminar sich um eine für jeden Studenten *eigenständige Prüfungsleistung* handelt und deshalb die Abgabe **nicht selbstständig erstellten kompletten Programmen oder ganzen Dokumentationsabschnitten zum NICHTBESTEHEN des Seminars führt** (wird geprüft!).

6.4 Pflichtpräsentationen, Dokumentation und entwickelte Software

6.4.1 Zwischenpräsentation

Jede Gruppe hat **eine Zwischenpräsentation** zu halten. *Die Termine für die Zwischenpräsentation* sind auf der Austauschplattform zu finden. Bitte beachten Sie folgende, allgemeine Richtlinien zur Gestaltung der Zwischenpräsentation:

- Dauer 10 Minuten, d.h. 2min pro Person
- Titelfolie mit Namen der Lehrveranstaltung, Gruppennummer und Namen der Gruppenmitglieder,
- Vorstellung des aufgebauten Roboters (inkl. Foto(s) auf einer Folie),
- Vorstellung des gesamten Zeitplans (Format ähnlich Tabelle 1),
- Kurzer Bericht jedes Studenten über sein Modul, inklusive aktuellem Stand, nächsten Zielen und eventuellen Problemen.

Für die Zwischenpräsentation ist eine Wiederholung der allgemeinen Aufgabenstellung nicht erforderlich, nutzen Sie Ihre Zeit effizient zur Darstellung Ihres aktuellen Standes! Bitte beachten Sie, dass das NXT-Beispielprogramm auf dem Roboter erprobt sein soll (eine Vorführung ist während der Zwischenpräsentation nicht erforderlich!). **Alle** Gruppen müssen während der Dauer der (gesamten) Veranstaltung anwesend sein! Beachten Sie bitte auch *Spezielle Hinweise zur Zwischenpräsentation* (siehe unten).

6.4.2 Verteidigungen

Des Weiteren hat jede Gruppe **zwei Verteidigungen** abzulegen. Diese bilden zusammen die Basis für die abschließende **Benotung**.

6.4.2.1 1. Verteidigung

In der **1. Verteidigung** ist die **grundsätzliche Funktionsweise des Roboters zu präsentieren**. Der Roboter soll dabei auf dem Parcours im **reduzierten Scout Modus** erfolgreich nach Parklücken suchen. Für die einzelnen Module müssen dafür folgende Teilaufgaben abgeschlossen sein:

- Perception: Bereitstellung aller Sensordaten außer des Maussensors,
- Control: Line-Control, Velocity-Control,
- HMI: Zeichnen des Parcours, des Pfades und der Parklücken,
- Navigation: funktionell vollständig und hinreichend genau und robust,
- Guidance: Entwurf der Zustandsmaschine.

<u>Teil 1: Vorführung des Roboters (5min pro Gruppe)</u>

Jede Gruppe hat genau 5min Zeit die Funktionalität des Roboters vorzuführen. Finden Sie sich daher als Gruppe mindestens 10min vorher ein, um den Roboter startklar zu machen. Auf dem Roboter muss das Programm der Gruppe bereits installiert sein. Auch müssen die beiden vorzuführenden Demoprogramme für Control installiert sein. Während des Vorführungszeitraums ist keine erneute Übertragung von Programmcode auf den Roboter möglich.

Ablauf:

- Betreten des Raumes
- Kalibrierung und Positionierung des Roboters auf dem Parcours
- Aktivierung von Scout mittels des HMIs
- Demonstration: Es befinden sich 3 valide Parklücken auf dem Parcours. Ziel ist, dass der Roboter 3 Runden am Stück auf der Spur bleibt, sich lokalisieren kann, alle Parklücken detektiert und pausieren kann. Jede Gruppe hat maximal 3 Versuche oder 5 Minuten Zeit für die Demonstration.
- Demoprogramme Control
    - Start der Demoprogramme jeweils über Robotermenü oder HMI

- Vorführung der Funktionalität

Teil 2: Vorstellung der eigenen Arbeit

Jedes Gruppenmitglied bereitet eine 2min Präsentation vor. Die Folien müssen zuvor über die Austauschplattform an die Betreuer gesendet werden. Die Deadline für das Zusenden können Sie rechtzeitig der Austauschplattform entnehmen. Die Verteidigungen der einzelnen Gruppenmitglieder erfolgt parallel. Das betrifft die Module (Control, Navigation, Perception, HMI). Für Guidance wird ein separater Termin vereinbart.

Ablauf:

- Präsentation des Vortrages (2min)
  - Beachten Sie hierzu die Inhalte, die in den Modulanleitungen gefordert werden
- Diskussion mit dem Betreuer (8min)

Bei der Funktion des Roboters werden die Zuverlässigkeit und Schnelligkeit, mit denen er die geforderten Aufgaben erfüllt, bewertet. *Der Termin wird über die Austauschplattform organisiert*.

Beachten Sie: Die angegebenen Zeitabläufe der Verteidigungen sind unbedingt einzuhalten, nach Überschreitung der angegebenen Zeiten wird der jeweilige Teil abgebrochen!

6.4.2.2 2. Verteidigung

Den **Abschluss des Seminars** bildet diee **2. Verteidigung, bei der das Gesamtprojekt vorgestellt wird**. Für die einzelnen Module müssen dafür alle Teilaufgaben abgeschlossen sein.

Teil 1: Vorführung des Roboters (5min pro Gruppe)

Jede Gruppe hat genau 5min Zeit die Funktionalität des Roboters vorzuführen. Finden Sie sich daher als Gruppe mindestens 10min vorher ein, um den Roboter startklar zu machen. Auf dem Roboter muss das Programm der Gruppe bereits installiert sein. Auch müssen die beiden vorzuführenden Demoprogramme für Control installiert sein. Während des Vorführungszeitraums ist keine erneute Übertragung von Programmcode auf den Roboter möglich.

Ablauf:

- Betreten des Raumes
- Kalibrierung und Positionierung des Roboters auf dem Parcours
- Aktivierung von Scout mittels des HMIs
- Demonstration: Zusätzlich zu den Anforderungen der 1. Verteidigung soll der Roboter die Modi Park this und beherrschen. Es befinden sich 3 valide und eine zu kleine Parklücke auf dem Parcours. Nach der ersten Runde wählt der Betreuer die Parklücken nacheinander im Park this Modus an. Die dritte Runde fährt der Roboter frei. Jede Gruppe hat maximal 3 Versuche oder 5 Minuten Zeit für die Demonstration.
- Demoprogramme Control
  - Start der Demoprogramme jeweils über Robotermenü oder HMI
  - Vorführung der Funktionalität

Während der Vorführung wird der Roboter an verschiedenen Stellen durch die Betreuer in den Modus Pause und anschließend wieder in den Modus Scout versetzt.

Teil 2: Vorstellung der eigenen Arbeit

Jedes Gruppenmitglied bereitet eine 2min Präsentation vor. Die Folien müssen zuvor über die Austauschplattform an die Betreuer gesendet werden. Die Deadline für das Zusenden können Sie rechtzeitig der Austauschplattform entnehmen. Die Verteidigungen der einzelnen Gruppenmitglieder erfolgt parallel. Das betrifft die Module (Control, Navigation, Perception, HMI). Für Guidance wird ein separater Termin vereinbart.

Ablauf:

- Präsentation des Vortrages (2min)
    - Beachten Sie hierzu die Inhalte, die in den Modulanleitungen gefordert werden
- Diskussion mit dem Betreuer (8min)

Bei der Funktion des Roboters werden die Zuverlässigkeit und Schnelligkeit, mit denen er die geforderten Aufgaben erfüllt, bewertet. *Der Termin wird über die Austauschplattform organisiert*.

Beachten Sie: Die angegebenen Zeitabläufe der Verteidigungen sind unbedingt einzuhalten, nach Überschreitung der angegebenen Zeiten wird der jeweilige Teil abgebrochen!

6.4.3 Dokumentation

Von jedem Studenten ist **2 Wochen vor der 2. Verteidigung** eine schriftliche **Dokumentation** seiner Arbeit (im PDF-Format) und der relevanten Quellcode einzureichen. In der Dokumentation sollen die Lösungen und Lösungswege zu den gestellten Aufgaben nachvollziehbar und ansprechend dargestellt werden. Zu jedem Modul wird in der Modulanleitung ein Beispiel-Inhaltsverzeichnis bereitgestellt, an dem sich der Inhalt der angefertigten Dokumentation orientieren muss. Der Umfang der Dokumentation soll *15 Seiten pro Student* nicht überschreiten. Die Angabe der Seitenanzahl bezieht sich auf technische Inhalte, d.h. Text, Bilder und Tabellen. Deckblatt, Inhaltsverzeichnis, Abbildungsverzeichnis (optional), Quellen, Anhänge, usw. werden nicht mitgerechnet. Bis 23:59 Uhr muss die Dokumentation in OPAL hochgeladen sein. Benennen Sie die Datei bitte nach dem Schema: "Dokumentation_Gruppe_*Gruppennr*_*Modul*_*Nachname*.pdf". Dokumentationen in ausgedruckter Form sind nicht erforderlich! Des Weiteren muss das **LeJOS NXT-Projekt** mit Quellcode für den NXT-Baustein als ein einziges kompilierbares und lauffähiges Eclipse-Projekt *zum gleichen Termin* als einzelne Zip-Datei hochgeladen werden. Die ganze Gruppe darf nur *ein* Projekt hochladen dieses Projekt darf auch nur relevante Modulimplementierungen enthalten. Wenn aus irgendeinem Grund vorgegebenen Struktur abgewichen werden muss (z.B. weil es für ein Modul unterschiedliche relevante Implementierungsvarianten gibt), dann ist unbedingt eine ausführliche Erklärung beizulegen. Das **Android-Projekt** muss unter *gleichen* Bedingungen *hochgeladen werden*. Bitte verwenden Sie das Namensschema: „Quellcode_NXT_Gruppe_*Gruppennr*.zip“. bzw. „Quellcode_Android_Gruppe_*Gruppennr*.zip“. Alle Eclipse-Projekte müssen eine vollständige und fertig kompilierte **Javadoc-Dokumentation** enthalten. Wenn Sie ein Programm zur Versionsverwaltung (z.B. git) verwenden, achten Sie darauf dass ihre zip-Dateien nicht die Versionierungsdaten nicht enthalten (.git-Verzeichnis), weil sonst unnötig große Dateien entstehen.

Die Seminarsprache ist Deutsch. Die Präsentationen und Dokumentationen müssen in der Seminarsprache gehalten/verfasst werden.

Anforderungen und Hinweise zu den Präsentationen (Zwischenpräsentation, 1./2. Verteidigung):

- Präsentationsfolien jeder Gruppe müssen spätestens **vier** Stunden vor Beginn der Veranstaltung als eine Quelldatei (z.B. pptx-Datei) und eine pdf-Datei (siehe mehr dazu unten) in OPAL hochgeladen werden. Namensschema: „Folien_Gruppe_*Gruppennr.xyz*“. Sicherheitshalber bringen Sie beide Dateien auch auf einem USB-Speicherstick mit.

- Wenn Sie Power-Point verwenden müssen die pptx-Dateien im Office-2010-kompatiblen Format gespeichert werden.
- Auf jedem Fall ist eine pdf-Version der Präsentationsfolien zu erstellen und mit abzugeben.
- Bei der Erstellung von Folien achten Sie bitte auf korrekte Darstellung von Folienelementen, vor allem von Grafiken und Schriften. Insbesondere beim pdf-Export können hier Probleme auftreten.
- Es ist empfehlenswert ein Notebook (mit VGA-Anschluss) mit Präsentationsfolien (pro Gruppe) mitzubringen.
- Hinweise zur Identifikation:
    - Präsentationsfolien allgemein: Bitte geben Sie Ihre Gruppennummer und alle Teilnehmer/innen auf der Titelfolie an.
    - Präsentationsfolien für Module müssen die Gruppennummer, den Modulnamen und den Namen vom jeweiligen Modulverantwortlichen enthalten.
    - Jede von Ihnen erstellte bzw. bearbeitete Text-Datei (vor allem Quellcodedateien) soll Ihre Gruppennummer und den(die) Namen von Autor(en) enthalten.
    - Im Verzeichnis des gesamten LeJOS NXT-Projekts oder des Android-Projektes legen Sie bitte eine Textdatei mit Ihrer Gruppennummer, Namen und Matrikelnummern aller Gruppenmitglieder an
    - Ihre Gruppennummer muss in Dateinamen für Präsentationsfolien und Zip-Archive der Softwareprojekte vermerkt werden. Siehe Namensschemata oben.
-

Spezielle Hinweise zur 1. Verteidigung:

- Es müssen sowohl die Präsentationsfolien **als auch die entwickelte Software** hochgeladen werden. Siehe Hinweise zur Benennung oben.
- Für die entwickelte Software (LeJOS NXT-Projekt und Android-Projekt) gelten die oben genannten Hinweise.
- Die Abgabefrist „spätestens **vier** Stunden vor Beginn der Veranstaltung“ gilt sowohl für die Präsentationsfolien als auch für die entwickelte Software.

Spezielle Hinweise zur 2. Verteidigung:

6.5 Die Dokumentation und die Software muss zur angegebenen Frist hochgeladen sein. Ca. 9 Tage vor der 2. Verteidigung.Bestimmungen für Ausgabe und Rückgabe der Hardware

Achtung! Kontrollieren Sie beim Erhalt der Hardware diese auf ihre Vollständigkeit. Alle Studenten sind für die Ihnen ausgehändigte Hardware selbst verantwortlich. Sie haben ihren Erhalt zu unterzeichnen und bei Verlust oder grob fahrlässiger bzw. mutwilliger Beschädigung die Hardware zu ersetzen.

Speziell für NXT-Robotersets. Falls zu Beginn des Seminars Teile fehlen machen sie bitte KEINE eigenen Eintragungen auf der Liste! Schreiben Sie Ihren Modulbetreuer kurz, dass etwas fehlt, notieren Sie sich die Teile auf einem extra Zettel und bringen Sie Set und Zettel zur nächsten Konsultation mit! Dort füllen Sie mit dem Betreuer zusammen eine Fehlliste aus. Wenn Teile überzählig sind, bringen Sie die bitte einfach so mit zur Konsultation.

Die ausgehändigte Hardware ist **spätestens 1 Woche nach Verteidigung des Seminars vollständig zurückzugeben**. Die LEGO-Teile sind entsprechend der Sortierübersichten in die Sortierfächer einzusortieren

und auf Vollständigkeit zu überprüfen. Eventuelle Klebereste sind vorher *vorsichtig* zu entfernen. Bitte beschreiben Sie die Teilelisten und die Sortierübersichten nicht!

Ort, Zeit sowie weitere spezielle Hinweise für die Rückgabe werden über die Austauschplattform rechtzeitig bekanntgegeben.

7 Notwendige Software und deren Installation

Bevor der gestellte Java-Quellcode kompiliert werden kann, ist eine Reihe von Vorbereitungen zu treffen. Vor allem muss die unten angegebene Software auf einem PC-Rechner installiert und ggf. konfiguriert werden.

WICHTIG: Bitte lesen und beachten Sie die folgenden Hinweise sorgfältig! Folgen Sie den Tutorials **Schritt für Schritt** und beachten Sie alle dort enthaltenen Hinweise.

7.1 Android Studio und SDK (nur HMI-Modul)

Hinweis: Dieser Teil der Softwareinstallation muss nur für die Bearbeitung des HMI-Moduls durchgeführt werden.

Hinweis: Google hat den Support für Eclipse ADT beendet. Daher ist es erforderlich in Android Studio zu arbeiten. Dies hat wiederum zur Folge, dass in zwei unterschiedlichen Entwicklungsumgebungen gearbeitet werden muss (Android Studio für die Appentwicklung und Eclipse für Anpassungen an der Robotersoftware). Die Pakete „parkingRobot“ in Android Studio und Eclipse müssen dementsprechend stets synchronisiert werden, da dort die Kommunikationsschnittstelle beschrieben ist. Dafür empfehlen wir die Verwendung von Versionskontrolle, wie in Abschnitt 7.3 beschrieben.

Zuerst muss Android Studio installiert werden. Die zugehörigen Installationsdateien werden unter https://developer.android.com/studio/index.html bereitgestellt. Folgen Sie anschließend den Installationsanweisungen, die unter https://developer.android.com/studio/install.html aufgeführt sind. Beachten Sie, dass Sie die Installationsdatei unter Windows als Administrator ausführen. Auf der Website mit den Installationshinweisen finden Sie auch weitere nützliche Hinweise in Bezug auf Android Studio. Machen Sie sich mit Hilfe dieser Hinweise mit der Software vertraut. Falls noch kein SDK auf dem Rechner installiert ist, dann wird dies automatisch während der Installation von Android Studio mitinstalliert, wenn der entsprechende Haken bei der Installation gesetzt ist. Nach dem ersten Start von Android Studio und der initialen Einrichtung gelangen Sie zu einem Fenster, an dem Sie ein neues Projekt anlegen können. Wählen Sie hier im unteren Bereich *Configure* und dann *SDK Manager* aus. Stellen Sie anschließend sicher, dass sie alle notwendigen Komponenten für die API 19 und API 25 installiert haben – falls diese nicht installiert sind so wählen Sie die notwendigen Komponenten aus und klicken Sie *Apply*. Des Weiteren müssen im Reiter *SDK Tools* die *Android SDK Build-Tools* in der Version *26.0.1*, das *ConstraintLayout for Android* und den *Solver for ConstrainedLayout* installiert werden.

Nach dem die Anweisungen erfolgreich durchgeführt wurden, fahren Sie bitte mit dem nächsten Abschnitt fort.

7.2 LeJOS, USB-Treiber, Eclipse und NXT-Beispielprogramm (alle Module)

Hinweis: Falls Sie ein 64Bit-System verwenden, beachten Sie die entsprechenden Hinweise in der Anleitung, dass sie NICHT alle 64Bit-Versionen installieren dürfen!

Hinweis: Für alle Nutzer von Windows einer anderen Version als Windows 7 empfiehlt sich die Verwendung einer Virtuellen Maschine, in welcher ein Windows 7 aufgesetzt wird. Das LeJOS-Projekt ist für andere Windowsversionen nicht fehlerfrei portierbar. Sowohl Windows 7 als auch der notwendige Lizenzschlüssel stehen jedem Studenten über das MSDNAA-Projekt kostenfrei zur Verfügung.

Zum kompilieren des LeJOS NXT-Projektes (z.B. NXT-Beispielprogramm) und für die Übertragung zum NXT-Baustein müssen, wie unter http://lejos.sourceforge.net/nxt/nxj/tutorial/Preliminaries/GettingStartedWindows.htm ausführlich beschrieben, die LeJOS NXJ Software (die letzte Version 0.9.1beta-3) im ersten Schritt und der USB-Treiber im zweiten Schritt installiert werden. Linux-Nutzer beachten bitte die Hinweise auf http://lejos.sourceforge.net/nxt/nxj/tutorial/Preliminaries/GettingStartedLinux.htm .

ACHTUNG: Die LeJOS-Firmware ist auf den NXT-Bausteinen bereits installiert! Bitte flashen Sie diese NICHT erneut, sondern überspringen diesen Schritt in den Tutorials.
Um ein Programm auf den NXT-Baustein zu übertragen nutzen Sie bitte ausschließlich die "Run as" -> "2 LeJOS NXT Program" - Funktion von Eclipse (siehe folgendes Tutorial).

Im dritten Schritt wird Eclipse mit LeJOS Eclipse-Plugins installiert. Hinweise für die Installation von Eclipse und des LeJOS Eclipse-Plugins finden sich unter http://lejos.sourceforge.net/nxt/nxj/tutorial/Preliminaries/UsingEclipse.htm.

7.2.1 Importieren des NXT-Beispielprogramms

Nachdem diese Vorbereitungen abgeschlossen sind, müssen Sie das vorbereitete NXT-Beispielprogramm als LeJOS NXT-Projekt in Eclipse importieren. Der Quellcode und die Javadoc-Dokumentation sind in einer Zip-Datei auf der Austauschplattform zu finden (in *NXT_v*.zip*, jeweils in Unterordnern *src*\ und *doc*\). Bitte erstellen Sie ein neues LeJOS NXT-Projekt mit dem Namen *NXT* und importieren oder kopieren Sie dann den Quellcode und die Javadoc-Dokumentation aus der Zip-Datei. Am einfachsten ist es, wenn Sie beide Unterordner *src*\ und *doc*\ in den Ordner des LeJOS NXT-Projektes (*workspace**NXT*) kopieren, z.B. mit dem Windiows-Explorer. Nach dem Kopieren müssen Sie das LeJOS NXT-Projekt in Eclipse aktualisieren (im *Package Explorer* auf dem Projektnamen rechtsklicken und dann *Refresh F5*), die kopierten Dateien werden dann dem Projekt automatisch hinzugefügt.

Weitere Informationen dazu in Abschnitt 9.1.

7.2.2 Installation von begleitenden Programmen

In diesem letzten Schritt werden begleitende Programme installiert. Momentan handelt sich um ein MATLAB-Skript zur Datendarstellung bei Verwendung der Monitor-Schnittstelle des NXT-Beispielprogramms (mehr dazu siehe in Abschnitt 10.4). Zur Installation kopieren Sie den Unterordner *view*\ aus der Zip-Datei (*NXT_v*.zip*) in den Ordner des LeJOS NXT-Projektes.

7.3 Verwendung von Versionskontrolle

Wenn Sie für die Zusammenarbeit in Ihrer Gruppe eine Versionsverwaltung (z.B. git) für den Quellcode verwenden möchten, ist es ggf. empfehlenswert, diese ebenfalls in Eclipse einrichten.Eine Möglichkeit dafür ist das Plugin EGit, siehe https://www.eclipse.org/egit/.

Bevor Sie anfangen erstmals Versionskontrolle zu benutzen, ist es ratsam, ein Tutorial dazu durchzuarbeiten, man kann sich dabei auch sehr verzetteln. Weitere Informationen finden Sie z.B. hier: https://tu-dresden.de/ing/elektrotechnik/rst/studium/diplom-master-studienarbeit/hinweise-fuer-die-durchfuehrung-von-diplom-master-und-studienarbeiten#section-2

8 Systemkonzept

8.1 Systemstruktur

Zur Modellierung der Aufgabe wurde die Systemstruktur in Abbildung 5 entworfen. Das System wurde dafür in verschiedene funktionale Einheiten gegliedert. Diese Einheiten werden als Module bezeichnet und können wiederum Untermodule besitzen. Die Module tauschen Informationen über Schnittstellen miteinander aus, die in der Implementierung durch *get-Methoden* repräsentiert werden.

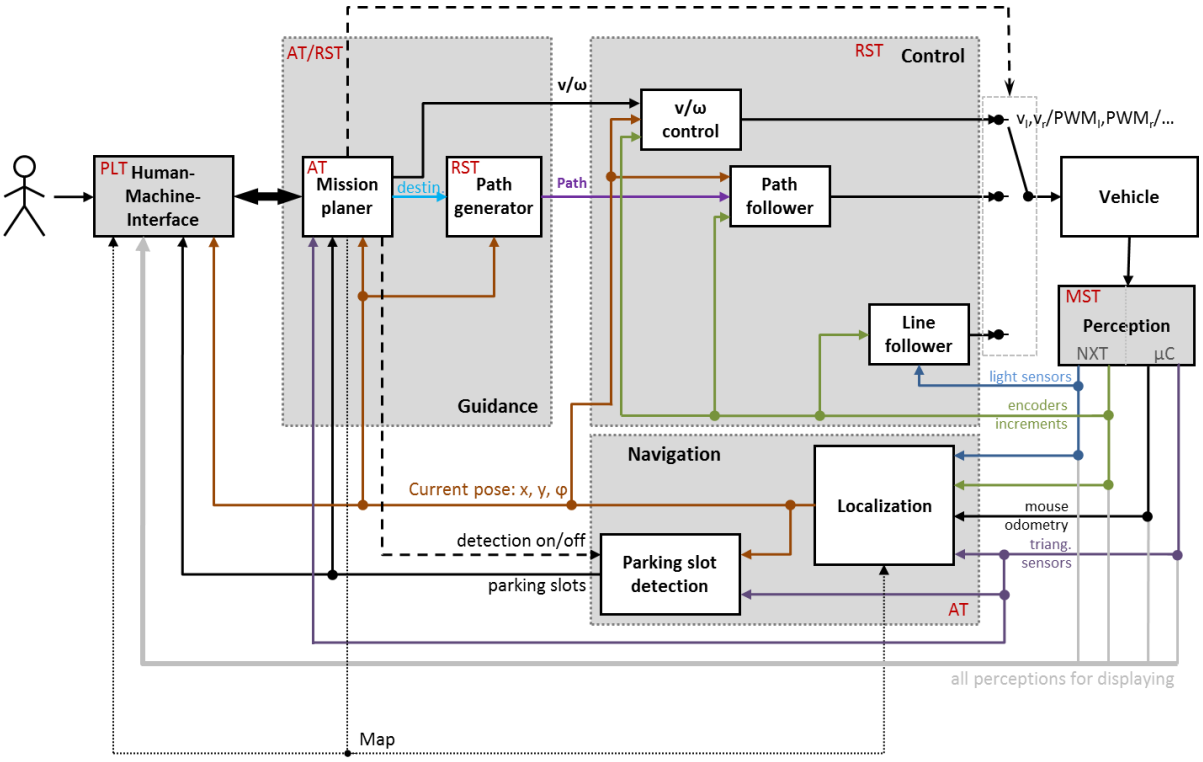

Abbildung 5: Systemstruktur

8.2 Schnittstellenstruktur

Die einzelnen Module tauschen Informationen über die Schnittstellen miteinander aus. Im Code werden diese Schnittstellen als Java-Interfaces implementiert. In der Grafik sind alle notwendigen Schnittstellen, welche die jeweiligen Module über get-Methoden zur Verfügung stellen müssen, an den Pfeilen angetragen. Die Module am Ursprung der Pfeile stellen die get-Methoden zur Verfügung. Die Klasse GuidanceAT.java hat eine Sonderrolle: Sie implementiert die main-Methode des Programms und stellt daher keine get-Methoden bereit. Pfeile die im Guidance-Modul ihren Ursprung haben stellen set-Methoden bei den Zielen dar.

Die Kommunikation zwischen dem Tablet und dem NXT basiert auf einer Bluetooth-Verbindung. In Abbildung 5 ist diese durch den Doppelpfeil zwischen dem HMI und Mission Planer Module gekennzeichnet.

Die Implementierung enthält auch ein HMI-Modul auf der Seite des NXT. Da es die Schnittstelle nur transparent durchreicht, ist es in der Grafik nicht mit verzeichnet.

Genauere Informationen zu den Schnittstellen finden sich in den Modul-Anleitungen und in der Javadoc-Dokumentation zum Quellcode.

8.3 THREADSTRUKTUR

Sowohl auf Roboterseite als auch auf dem Tablet ist der Programmablauf mit Threads parallelisiert. Auf dem NXT werden die Threads in den Konstruktoren der Modulklassen gestartet welche in der main-Methode des Guidance-Moduls Instanziiert werden.

Threads dienen zum Parallelisieren wiederkehrender Aufgaben. Threads bieten allerdings keine „echte“ Parallelität. Man spricht in diesem Fall von Nebenläufigkeit. Das Betriebssystem schaltet mithilfe eines Schedulers zwischen den Prozessen um.

„Echte“ Parallelität ist lediglich bei Mehrkernprozessoren möglich. Im Projekt verwendet aber der NXT einen Einkernprozessor, sodass auf Threads zurückgegriffen werden muss, damit die einzelnen Module ihre Aufgaben quasi-parallel (nebenläufig) ausführen können. Auf Seite des NXT erzeugt also jedes Modul mindestens einen Thread.

Fehler im Threading sind schwer zu erkennen und zu debuggen. Der Zugriff mehrerer Threads auf eine Ressource verursacht Probleme. Um den korrekten Ablauf bei gemeinsamem Zugriff auf eine Ressource zu gewährleisten, sind Synchronisierungsmaßnahmen erforderlich. Das Schlüsselwort *synchronized* signalisiert in Java die Verwendung eines sogenannten Monitors. Dieser stellt mittels einer Warteschlange sicher, dass immer nur ein Thread Zugriff auf eine solche Zugangsmethode hat.

Eine Einführung zum Thema findet sich z.B. auf den Webseiten http://openbook.rheinwerk-verlag.de/javainsel/, weiter zum Kapitel *12 Einführung in die nebenläufige Programmierung* und http://www.dpunkt.de/java/Programmieren_mit_Java/Multithreading/2.html

Hinweis: Auf der genutzten Plattform kann immer nur ein einziger Thread laufen und bleibt dann so lange aktiv bis er beendet wird oder sich selber in den Schlafzustand versetzt. Danach kommt der nächste wartende Thread mit der höchsten Priorität an die Reihe. Fordern verschiedene gleich priorisierte Threads Rechenzeit an, kommt derjenige an die Reihe, der zuerst angefragt hat. Im bereitgestellten Beispielcode werden von den Modulen bereits Threads genutzt, die gleich priorisiert und mit festen Sleep-Zeiten versehen sind. Diese Zeiten und Prioritäten wurden auf Werte eingestellt, die für die Beispielimplementierung sinnvoll sind, müssen aber gegebenenfalls noch an die Bedürfnisse der tatsächlichen Modulaufgaben angepasst werden.

9 HINWEISE ZUR IMPLEMENTIERUNG

9.1 HIERARCHIE DES LEJOS NXT-PROJEKTES

Die bereitgestellten Interfaces liegen im Package *parkingRobot* und müssen dort unverändert verbleiben. In diesem Paket soll, wie im Folgenden erklärt, ein Unterpaket mit dem Namen hsamrX erstellt werde, wobei X der Gruppennummer entspricht.

Am einfachsten ist es, ausgehend vom mitgelieferten NXT-Beispielprogramm zu arbeiten. Dieses ist als Unterpacket hsamr0 in der Quellcode-Zip-Datei *NXT_v*.zip* enthalten.

- Wie in Abschnitt 7.2.1 erklärt, erstellen Sie ein neues LeJOS NXT-Projekt mit dem Namen *NXT* und importieren oder kopieren Sie die in der Zip-Datei enthaltenen Dateien in das Projekt.
- Benennen Sie im Package-Explorer das hsamr0-Unterpaket mit der Taste F2 um. Dadurch sollten alle Zugehörigkeiten korrekt mit umgeändert werden. Wenn das nicht der Fall ist, muss manuell in der ersten Zeile jeder Klassendatei folgendes eingefügt werden:

```
package parkingRobot;
```

bzw.

```
package parkingRobot.HMSRx;
```

Die Datei- und Klassennamen dürfen nicht umbenannt werden. Dadurch sind Module zu Testzwecken leicht austauschbar.

Nach dem Umbenennen können sie das Verzeichnis hsamr0 erneut als Unterpaket in Ihr Projekt importieren um jederzeit auch das NXT-Beispielprogramm auf den Roboter aufspielen zu können.

9.2 HERANGEHENSWEISE BEI DER ENTWICKLUNG DER SOFTWARE

Es ist empfehlenswert den bereitgestellten Beispielcode schrittweise zu erweitern und dabei regelmäßig zu testen. Die Fehlersuche kann andernfalls aufwändig werden, da mit fremdem, wenig bekanntem Code gearbeitet wird.

9.3 PROGRAMMIERRICHTLINIEN

Bei der Implementierung sollten Sie sich an die Java-Code Richtlinien halten. Dies macht es bedeutend einfacher, Code zu verstehen und zu verbessern.

- Der erste Buchstabe von Klassennamen sollte immer großgeschrieben werden, der erste Buchstabe von Methoden und Objekten immer klein. Beispiel: DasIstEinKlassenname, dasIsteinMethodenname
- Bezeichner von Klassen, Methoden und Variablen sollten selbsterklärend sein und ihre Funktion bereits beim Lesen klar werden lassen.
- Alle Buchstaben von Konstanten (static final) sollten großgeschrieben werden, um direkt klar zu machen, dass sich deren Werte zur Laufzeit nicht ändern.
- Methoden und Klassen sollten immer, wenn sie nicht von anderen Modulen benötigt werden, als private deklariert werden! Variablen sollten nach Möglichkeit stets als private deklariert sein. Wenn sie übergeben werden sollen, empfehlen sich Methoden zu schreiben, die die Variable empfangen und übergeben (so genannte setter und getter). Beispiel: getVaribale(), setVariable(Typ var)
- Schließende Klammern } stehen auf der gleichen Einrückungsstufe, wie die öffnende Klammer {

- Der Quelltext sollte vollständig kommentiert werden, damit das Nachvollziehen für andere leichter wird. Außerdem hilft ausführliches Kommentieren dabei, sein programmiertes noch einmal zu reflektieren und möglicherweise Fehler zu entdecken.
- Alle als public deklarierten Methoden und Klassen sollten mit einem Javadoc-Kommentar versehen werden. Mittels Javadoc kann eine Klassendokumentation in HTML generiert werden. Eingeleitet wird ein Javadoc-Kommentar mit /** und geschlossen mit */
  Um fehlende Kommentare anzuzeigen, bietet Eclipse entsprechende Einstellungen an: unter Window > Preferences unter Java > Compiler > Javadoc (siehe Abbildung 6: Javadoc Warning Einstellungen).

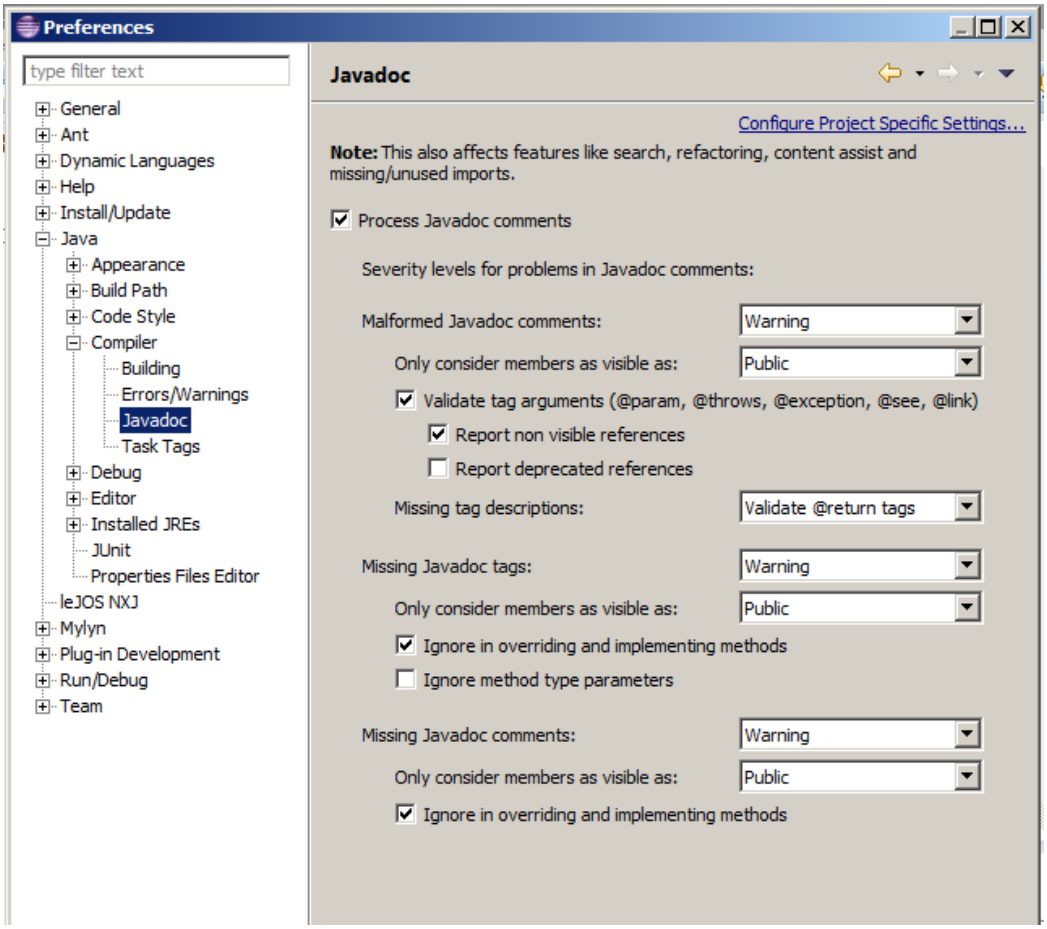

ABBILDUNG 6: JAVADOC WARNING EINSTELLUNGEN

9.4 ERSTELLEN VON JAVADOC-DOKUMENTATION

Die Javadoc-Dokumentation des NXT-Beispielprogramms wird über das *\doc* Verzeichnis in der Zip-Datei bereitgestellt. Sie können diese Hinweise nutzen, um sie zu aktualisieren oder die von Ihnen erweiterte bzw. vorbereitete Javadoc-Dokumentation in anderen Eclipse-Projekten zu erstellen.

Nachdem Sie alle erforderliche Software installierten und das LeJOS NXT-Projekt des NXT-Beispielprogramms erstellten, sollen Sie noch folgende vorbereitende Schritte in Eclipse durchführen:

- Wenn erforderlich, Eclipse starten, workspace bestätigen.
- Die Warnungseinstellungen wie in Abschnitt 9.3 beschrieben kontrollieren und ggf. vornehmen.
- Die LeJOS NXJ-Bibliothek dem *Java Build Path* im Projekt hinzufügen:

- rechter Mausklick am Projektnamen im *Package Explorer* links -> *Properties* -> *Java Build Path* -> *Libraries* -> *Add External JARs,*
- zum Ordner mit der Java-Archive der Bibliothek *classes.jar* gehen, z.B. *c:\Program Files (x86)\leJOS NXJ\lib\nxt\,*
- die Java-Archive *classes.jar* doppelklicken,
- *OK.*
    - Das Fenster *Generate Javadoc* konfigurieren:
- *In der Menüleiste der Eclipse: Project -> Generate Javadoc...*, im Fenster *Generate Javadoc* unter *Javadoc command* die Datei *javadoc.exe* komplett mit ihrem Pfad angeben. Typischer Ort für diese Datei ist beim Java Development Kit (JDK), z.B. *C:\Program Files (x86)\Java\jdk?.?.?_??\bin\javadoc.exe.*
- In demselben Fenster *Generate Javadoc* unter *Use standard doclet*, *Destination* den kompletten Zielpfad für die Javadoc-Dateien (das Verzeichnis *\doc*) angeben (wenn erforderlich).
- *Cancel.*

Ab jetzt können Sie die Javadoc jede Zeit erstellen:

- Vergewissern Sie sich, dass die Markierung im *Package Explorer* links auf dem LeJOS NXT-Project steht.
- Gehen Sie dann über *Project -> Generate Javadoc...* wieder zum Fenster *Generate Javadoc*.
- Dort stellen Sie sicher, dass das LeJOS NXT-Projekt angekreuzt ist, und klicken dann die *Finish*-Taste.
- Beim ersten Mal müssen die Aufforderung im Dialogfenster mit *Yes, to all* bestätigen.
- Das Erstellen der Dokumentation wird im Eclipse-View „Console“ protokolliert.

10 Beigefügtes NXT-Beispielprogramm (für NXT)

10.1 Allgemeines

Nachdem die LeJOS NXJ Software, USB-Treiber, Eclipse mit den notwendigen Plugins installiert sind und das NXT-Beispielprogramm importiert ist, kann das NXT-Beispielprogramm kompiliert, auf den NXT geflasht und dort ausgeführt werden. Das Programm bietet nur eine grundlegende Funktionalität. Beim Lösen der Aufgabenstellungen soll auf dieser aufgebaut werde, so dass stets ein lauffähiges Programm zum Testen der Implementierungen zur Verfügung steht.

Das NXT-Beispielprogramm setzt folgende Anschlüsse des NXT-Bausteines voraus:

- der (in Fahrtrichtung) rechte Lichtsensor – Port 1,
- der linke Lichtsensor – Port 2,
- der rechte Motor – Port A,
- der linke Motor – Port B.

Es ist bereits ein sehr einfacher Algorithmus zur Linienverfolgung, eine Navigation basierend auf den Radencodern, die vollständige Bereitstellung der Sensordaten und die Bluetooth-Kommunikation mit dem Tablet implementiert. Viele weitere Methoden sind allerdings auch erst als Pseudocode angelegt und müssen gemäß den Aufgabenstellungen selber implementiert werden.

Bei Kommunikation mit dem Android-Tablet kann der NXT über dieses bedient werden und Mess- und Navigationsdaten werden darauf angezeigt. Dafür ist es notwendig, auf dem Tablet das Android-Beispielprogramm (Android-Beispielprojekt) zu installieren, was ebenfalls zur Verfügung gestellt wurde.

Im Folgenden wird der Programmablauf grob beschrieben. Detailliertere Informationen zum den implementierten Funktionen finden Sie in den Modulanleitungen, der Javadoc-Dokumentation des Quellcodes und im Quellcode selber.

10.2 Programmablauf

Bei Start des NXT-Beispielprogramms auf dem NXT wird gleich ein Unterprogramm aufgerufen, welches ermöglicht die Sensoren für weißen und schwarzen Untergrund zu kalibrieren. Dies ist notwendig, um die Linienverfolgung korrekt auszuführen.

Danach kann ausgewählt werden, ob sich der Roboter mit Bluetooth verbinden, oder über die vorhandenen Knöpfe am NXT-Baustein bedient werden soll. Soll Bluetooth benutzt werden, muss mit *Enter* bestätigt werden (oranger Knopf), andernfalls mit *Escape* (grauer Knopf).

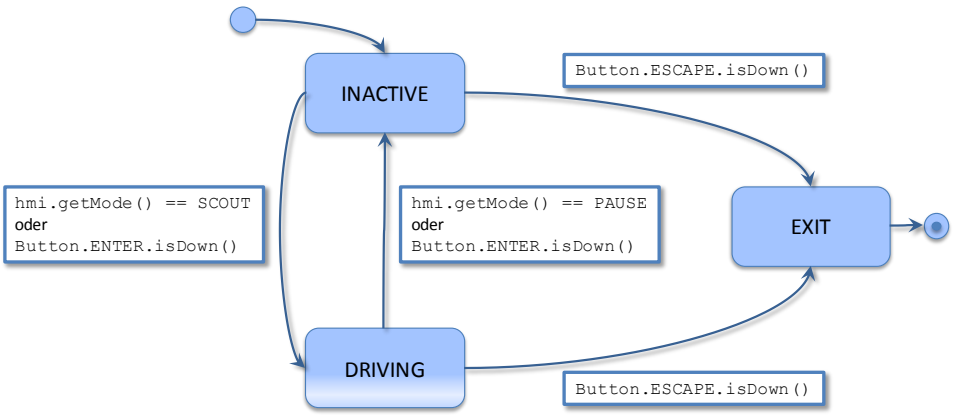

Abbildung 7: Zustandsmaschine des NXT-Beispielprogramms (Zustandsübergangsdiagram)

Danach wird die Hauptroutine abgearbeitet (Abbildung 7). In der Standardeinstellung beginnt der NXT im Wartemodus (INACTIVE). Die implementierte Linienverfolgung kann mit einem Druck auf *Enter* oder durch eine Schaltfläche auf dem Tablet gestartet werden, der NXT geht dann in den Zustand DRIVING. Der Roboter zeigt ständig die ermittelten Navigationsdaten auf dem Display an. Ein erneuter Druck auf *Enter* (oder die Schaltfläche auf dem Tablet) unterbricht die Linienverfolgung und lässt den Roboter erneut in den Wartemodus gehen. Ein Druck auf *Escape* beendet das Programm (EXIT).

10.3 Mitgliedsklasse EncoderSensor

Die Klasse `EncoderSensor` ist eine Mitgliedsklasse des Interfaces `IPerception`. Ihre Funktion ist wichtig, (auch für den Bearbeiter des Control und des Navigation-Moduls) aber nicht ganz einfach zu verstehen, daher soll hier eine kurze Erklärung das Verständnis unterstützen.

Bei jeder Abfrage des Encodersensor-Objekts gibt dieses die gemessene Winkelsumme seit dem letzten Aufruf und die seitdem vergangene Zeit zurück. Nach der Abfrage wird die Winkelsumme wieder auf null und die Referenzzeit auf die momentane Zeit gesetzt. Wegen dieses Rücksetzens kann man pro Abfragezeitpunkt die Abfragemethode `getEncoderMeasurement()` nur einmal (!) aufrufen. Damit man bei dieser einen Abfrage dennoch beide gewünschten Werte (Winkelsumme und Zeitdifferenz) übergeben bekommt, gibt die Methode `getEncoderMeasurement()` ein `IPerception.AngleDifferenceMeasurement` Objekt zurück. Speichert man dieses, kann man dann beliebig oft mit den Methoden `getAngleSum()` und `getDeltaT()` die gespeicherte Winkelsumme und die zugehörige Zeitdifferenz daraus abfragen.

Unterschiedliche Encodersensor-Objekte beeinflussen sich aber gegenseitig nicht! Das Navigationsmodul muss daher nicht mit dem Controlmodul abgestimmt werden: Jedes Modul erhält ein eigenes Encodersensor-Objekt und kann daher problemlos öfter/seltener und zu anderen Zeitpunkten als das jeweilige andere Modul die Encodermesswerte auslesen.

10.4 Monitor-Schnittstelle für Programmierer

10.4.1 Überblick

Um das Verhalten des Roboters besser nachvollziehen zu können, wurde eine Monitor-Schnittstelle implementiert. Diese Schnittstelle realisiert zum einen die Funktion des Datenlogging und zum anderen die der Datendarstellung. Das Datenlogging wird von Seiten des NXT über ein zusätzliches Modul mit eigenem Thread realisiert. Dieses Modul greift beim Senden der Daten auf die leJOS-Klasse NXT Data Logger zurück. Auf dem über USB oder Bluetooth verbundenen PC werden die gesendeten Daten vom leJOS-Programm NXT Charting Logger entgegen genommen. Die Daten werden vom NXT Charting Logger in einer Textdatei auf dem PC gespeichert und können zusätzlich mithilfe eines MATLAB-Skriptes dargestellt werden. Im Folgenden wird die Verwendung der Monitor-Schnittstelle erläutert.

10.4.2 Einrichtung

Um die Monitor-Schnittstelle verwenden zu können ist ein nach **Kapitel** 7 eingerichtetes System nötig. Das leJOS-Programm NXT Charting Logger (*nxjchartinglogger.bat*) befindet sich dann standardmäßig unter *C:\Program Files\leJOS NXJ\bin*. Um gespeicherte Logdateien ordnungsgemäß darstellen zu können, wird außerdem ein Unix- bzw. Linux-basierter Texteditor (beispielsweise notepad++) benötigt. Zur erweiterten Darstellung der Logdateien wird MATLAB benötigt. Hierbei besteht die Möglichkeit auf die vom IfA bereitgestellte Studentenversion zurückzugreifen. Es kann auch die MATLAB-Version auf den Rechnern im ET-Pool verwendet werden.

Zum Herstellen einer Verbindung zwischen NXT und PC über USB ist lediglich das Anschließen des USB-Kabels nötig. Beim erstmaligen Verbinden muss gegebenenfalls kurz abgewartet werden.

Das Herstellen einer Verbindung zwischen NXT und PC über Bluetooth ist etwas aufwändiger. Zunächst ist PC-seitig entweder das interne Bluetooth-Modul zu aktivieren oder ein externes Bluetooth-Modul anzuschließen. Das Aktivieren eines internen Moduls erfolgt in der Regel über das Aufrufen eines herstellerspezifischen Menüs zur Einstellung von Drahtlosverbindungen, welches meistens über eine Taste oder Tastenkombination aufgerufen werden kann. Ein externes Modul ist nach Anschluss und einmaliger Treiberinstallation sofort aktiviert. Falls für die Arbeit am Hauptseminar eine virtuelle Maschine verwendet wird, muss das Bluetooth-Modul nach Aktivierung noch in die virtuelle Maschine durchgereicht werden. Bei der Verwendung eines externen Bluetooth-Moduls ist dies in der Regel problemlos möglich. Bei der Verwendung des internen Bluetooth-Moduls kann es je nach System vorkommen, dass ein Durchreichen nicht möglich ist. Beim erstmaligen Herstellen einer Verbindung zwischen einem PC und dem NXT ist ein Pairing der beiden Geräte notwendig. Das Pairing erfolgt über einen Rechtsklick auf das Bluetooth-Symbol im Infobereich in der Taskleiste am PC-Desktop und anschließendem Klick auf *Gerät hinzufügen*. Anschließend muss der richtige NXT ausgewählt werden. Der Name des NXT wird im Hauptmenü oben in der Mitte des Displays angezeigt.

Beachte: Das externe Bluetooth-Modul kann bei Bedarf in Form eines Bluetooth-USB-Dongles ausgeliehen werden. Dafür kontaktieren Sie bitte den Betreuer vom Institut für Automatisierungstechnik.

10.4.3 VERWENDUNG

Um später selbst die Monitor-Schnittstelle sinnvoll einsetzen zu können, ist es insbesondere für die Bearbeiter der Module Control, Guidance und Navigation von Vorteil, sich mit der Verwendung der Schnittstelle vertraut zu machen. Das NXT-Beispielprogramm enthält hierzu im Control-Modul ein Beispiel, welches im Folgenden Schritt für Schritt erläutert wird. Die entsprechenden API-Funktionen sind in den anderen Modulen vorbereitet aber auskommentiert.

Datenlogging

Stellen Sie zunächst sicher, dass Sie alle in Kapitel 7.2.1 beziehungsweise 9.1 beschriebenen Schritte ausgeführt haben. Öffnen Sie die Datei *Monitor.java* im Unterordner *parkingRobot.hsamrX* Ihres Eclipse-Projektes. Die Datenlogging-Funktion kann über die Konstante *MODE* vom Typ *monitorMode* (Enumerationstyp) zu Beginn des Quelltextes der Klasse *Monitor* (diese Stelle ist im Quelltext durch den Hinweis *USER INPUTS* markiert) aktiviert (*monitorMode.ONLINE_LOGGING*) oder deaktiviert (*monitorMode.LOGGING_OFF*) werden. Nehmen Sie bitte außerhalb des gekennzeichneten Bereiches keine Änderungen vor – das gilt auch für die Dateien *MonitorThread.java* und *IMonitor.java*. Aktivieren Sie nun die Datenlogging-Funktion.

Die Datenlogging-Funktion generiert zwei verschiedene Logdatentypen: Nutzerlogdaten und Präambel des NXT Charting Logger. Die Nutzerlogdaten sind Tabellenzeile und Zeichenzeile. Mit einer *Tabellenzeile* können vor allem Werte von Variablen gesendet werden. Mithilfe einer *Zeichenzeile* kann eine beliebige Zeichenkette (siehe Feld „Inhalt“) gesendet werden. Die Nutzerlogdaten werden automatisch mit einem Zeitstempel versehen.

Ein Nutzerlogeintrag enthält entweder eine Tabellenzeile oder eine Zeichenzeile. Ein Logeintrag besteht aus einer Präambel und einer beliebigen Anzahl von Nutzerlogeinträgen.

Format einer Präambel (in der Logdatei):

- Zeitstempel des NXT Data Logger“Tabulator““0““\n“Zeitstempel des NXT Data Logger“Tabulator““\n“

Format Tabellenzeile (in der Logdatei):

- Zeitstempel;Modulnummer;Nutzerlogdatentyp;Wert1;Wert2;Wert3;… WertN“\n“

Format Zeichenzeile (in der Logdatei):

- Zeitstempel;Modulnummer;Nutzerlogdatentyp;Inhalt“\n“

Beachte:

- Der Zeitstempel des NXT Data Logger in der Präambel ist eine ascii-kodierte ganze Zahl und wird automatisch beim Sendevorgang generiert. Die Messeinheit des Zeitstempels ist Millisekunde.
- “\n“ ist Zeilenumbruch (“x0A“)
- Der Zeitstempel ist eine ascii-kodierte ganze Zahl und wird automatisch beim Aufruf der unten erläuterten Funktionen *writeControlVar* und *writeControlComment* generiert. Die Messeinheit des Zeitstempels ist Millisekunde.
- Nutzerlogdatentyp der Tabellenzeile ist immer “0“.
- Nutzerlogdatentyp der Zeichenzeile ist immer “1“.
- Modulnummern ergeben sich entsprechend der alphabetischen Reihenfolge der Modulnamen: “0“ (Control), “1“ (Guidance), “2“ (HMI), “3“ (Navigation), “4“ (Perception).
- WertX ist die in einen String konvertierte zu übermittelnde Zahl (siehe *writeControlVar*).
- Inhalt darf keine Backslashs “\“ beinhalten.

Nachfolgend wird die Verwendung der API-Funktionen zum Senden der Daten näher erläutert (API steht für application programming interface).

Öffnen Sie nun die Datei *ControlRST.java* (Unterordner *parkingRobot.hsamrX*). Die beispielhaften Methodenaufrufe im Quelltext des Control-Moduls sind immer mit dem Kommentar *MONITOR (example)* versehen. Die entsprechenden Stellen lassen sich über die Suchfunktion von Eclipse leicht finden.

Das Übertragen der Zeilen an den Rechner erfolgt gepuffert (Für die Tabellenzeile ist die Puffergröße gleich eine Zeile.). Der Monitor-Thread läuft mit einer niedrigeren Priorität als die anderen Module. Bei Ausführung des Monitor-Threads werden zuvor alle gepufferten Daten gesendet. Bei jedem Sendevorgang wird den zu sendenden Daten automatisch eine Präambel vorangestellt.

Um eine Tabellenzeile zu puffern, müssen vorher im Konstruktor der Klasse des jeweiligen Moduls die Spalten der Tabelle durch Vergabe eines Namens definiert werden. An der ersten Stelle im Quelltext, die Sie über die Suchfunktion erreichen, befindet sich eine solche Definition der Tabellenspalten in Form zweier Aufrufe der Monitor-Methode *addControlVar(String name)*. Es werden zwei Spalten mit den Namen *RightSensor* und *LeftSensor* definiert. Springen Sie zur nächsten Stelle im Quelltext. Sie befinden sich nun in der Methode, welche die Linienverfolgung realisiert. Beim Aufruf dieser Methode sollen die Werte des rechten und linken Lichtsensors (hier jeweils 0, 1 oder 2) in eine Tabellenzeile geschrieben werden. Die Methode *writeControlVar(String name, String var)* ermöglicht das Schreiben eines Variablenwertes in die entsprechende Spalte der Tabellenzeile. Als Argumente müssen der Name der gewünschten Spalte sowie der zugehörige Variablenwert übergeben werden. Dabei muss sichergestellt werden, dass der übergebene Name exakt mit dem bei der Definition der Spalte verwendeten Namen übereinstimmt. Außerdem ist zu beachten, dass der Variablenwert in einen String konvertiert werden muss. Dies lässt sich, wie im Quelltext zu sehen, leicht durch eine Stringaddition erreichen.

Hinweis: In jedem Modul darf pro Aufruf der jeweiligen Aktualisierungsfunktion(en) insgesamt nur eine Tabellenzeile (bestehend aus einer oder mehreren Spalten) gepuffert werden. Sollen mehrere Tabellenzeilen pro Modulaufruf gepuffert werden, so wird die zuvor gepufferte Tabellenzeile überschrieben.

Um eine Zeichenzeile zu puffern, genügt es die Monitor-Methode *writeControlComment(String str)* aufzurufen. Als Argument wird dabei ein frei wählbarer String übergeben. Betrachten Sie nun erneut den Quelltext des Control-Moduls. Sie werden feststellen, dass sich unterhalb der zuletzt beschriebenen Stelle im Quelltext sechs *if*-Blöcke befinden. Falls sich einer der beiden Lichtsensoren ganz oder fast auf der Linie befindet, ist eine der

Bedingungen erfüllt und der Roboter dreht sich entsprechend entweder nach links oder rechts. In jedem der Blöcke wird eine der Drehrichtung entsprechende Zeichenzeile (*turn right* oder *turn left*) in den Puffer geschrieben.

Schalten Sie nun Ihren NXT ein. Nach dem Laden des Betriebssystems erscheint das Hauptmenü. Der Name Ihres NXT wird oben in der Mitte des Displays angezeigt. Notieren Sie sich den Namen Ihres NXT (standardmäßig *HSAMRX*). Öffnen Sie das leJOS-Programm NXT Charting Logger auf dem PC und tragen Sie den Namen Ihres NXT in das dafür vorgesehene Textfeld (*NXT Name/Address:*) ein. Den Speicherort sowie den Namen der Logdatei sollten Sie sinnvoll anpassen (*Logfile:*). Laden Sie anschließend mit Eclipse das NXT-Beispielprogramm auf den NXT.

Nach erfolgreichem Laden startet das Programm automatisch und Sie werden dazu aufgefordert eine Verbindungsart (USB oder Bluetooth) auszuwählen. Nach erfolgter Auswahl versucht der NXT 30 Sekunden lang sich mit dem NXT Charting Logger zu verbinden. Klicken Sie im NXT Charting Logger auf den Button *Connect* und warten Sie ab. War der Verbindungsversuch erfolgreich, so wird dies im Display des NXT angezeigt und es ertönt ein akustisches Signal.

Falls der Verbindungsaufbau scheitert, tritt eine Exception auf. Halten Sie in diesem Fall die Tasten *Enter* und *Escape* auf Ihrem NXT gleichzeitig gedrückt, um einen Neustart zu erzwingen. Starten Sie anschließend das NXT-Beispielprogramm erneut.

Nachdem die Verbindung mit dem NXT Charting Logger erfolgreich aufgebaut wurde, werden Sie dazu aufgefordert die Lichtsensoren zu kalibrieren. Nach erfolgter Kalibrierung kann sich der NXT mit dem Tablet verbinden. Für das NXT-Beispielprogramm ist dies jedoch zunächst nicht notwendig. Drücken Sie daher *Escape* auf dem NXT.

Hinweis: Zwei gleichzeitige Bluetooth Verbindungen mit dem NXT sind nicht möglich. Um die Datenlogging-Funktion bei gleichzeitiger Verbindung mit dem Tablet dennoch nutzen zu können, muss die Verbindung zum PC über USB hergestellt werden.

Stellen Sie nun den NXT auf einen Parcours. Nach Betätigung der *Enter*-Taste fährt der Roboter los (dazu muss einer der beiden Lichtsensoren auf der Linie sein). Der NXT Charting Logger empfängt nun fortlaufend die vom NXT gesendeten Logeinträge. Diese erscheinen in einem Textausgabefeld, wenn der Reiter *Data Log* ausgewählt ist. Das Datenlogging kann nur dann ordnungsgemäß beendet werden, wenn das auf dem NXT laufende Programm durch Drücken von *Escape* beendet wird. Verwenden Sie **nicht** den Button *Disconnect* im NXT Charting Logger, wenn das Programm auf dem NXT noch läuft. Nun können Sie das NXT-Beispielprogramm beenden.

Die Abbildung 8 und die Abbildung 9 zeigen beispielhafte Nutzerlogeinträge mit einer Tabellenzeile beziehungsweise einer Zeichenzeile im NXT Charting Logger sowie den korrespondierenden Quelltext im Control-Modul. Um die empfangenen Nutzerlogeinträge später möglichst einfach auswerten zu können, wird bei deren Erstellung das festgelegte Format verwendet.

```
// MONITOR (example)
monitor.addControlVar("RightSensor");
monitor.addControlVar("LeftSensor");
```

```
private void exec_LINECTRL_ALGO(){
    leftMotor.forward();
    rightMotor.forward();
    int lowPower = 1;
    int highPower = 30;

    // MONITOR (example)
    monitor.writeControlVar("LeftSensor", "" + this.lineSensorLeft);
    monitor.writeControlVar("RightSensor", "" + this.lineSensorRight);
```

Dreimal aufgerufen

Data Log | Status | Chart

```
18065    0
18065
8658;0;0;0;2
8658;0;1;turn left
18170    0
18170
8763;0;0;0;1
8763;0;1;turn left
18273    0
18273
8868;0;0;0;0
18378    0
```

Abbildung 8: Tabellenzeile –Implementierung und Ausgabe

```
else if(this.lineSensorLeft == 2 && (this.lineSensorRight == 0)){

    // when left sensor is on the line, turn left
    leftMotor.setPower(lowPower);
    rightMotor.setPower(highPower);

    // MONITOR (example)
    monitor.writeControlComment("turn left");

}
```

```
else if(this.lineSensorLeft == 1 && this.lineSensorRight == 0) {

    // when left sensor is on the line, turn left
    leftMotor.setPower(lowPower);
    rightMotor.setPower(highPower);

    // MONITOR (example)
    monitor.writeControlComment("turn left");

}
```

Data Log | Status | Chart

```
18065    0
18065
8658;0;0;0;2
8658;0;1;turn left
18170    0
18170
8763;0;0;0;1
8763;0;1;turn left
18273    0
18273
8868;0;0;0;0
18378    0
```

Abbildung 9: Zeichenzeile –Implementierung und Ausgabe

Sicherlich wird Ihnen aufgefallen sein, dass im Textausgabefeld des NXT Charting Logger, zusätzlich zu den oben abgebildeten beispielhaften Ausgaben von Tabellenzeilen und Zeichenzeilen, auch noch zwei weitere Zeilen (Abbildung 10) mit gewisser Regelmäßigkeit ausgegeben werden. Diese beiden Zeilen sind die Präambel und können ignoriert werden.

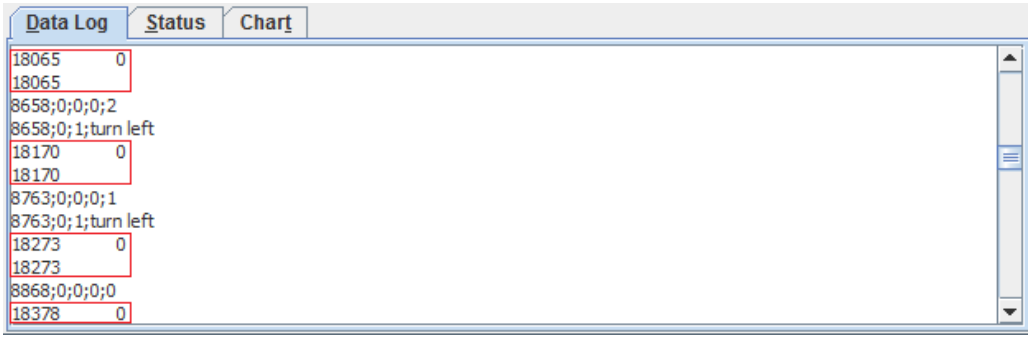

ABBILDUNG 10: PRÄAMBEL – DIESE ZEILEN KÖNNEN IGNORIERT WERDEN.

Öffnen Sie nun die vom NXT Charting Logger abgespeicherte Textdatei. Bei Verwendung eines Unix- bzw. Linux-basierter Texteditors sollte die Darstellung der Logeinträge genauso aussehen wie im NXT Charting Logger. Falls kein solcher Texteditor verwendet wird, können die Zeilenumbrüche nicht ordnungsgemäß dargestellt werden.

Datendarstellung

Für eine bequemere Auswertung der Logging-Daten empfiehlt sich die Nutzung des bereitgestellten MATLAB-Skriptes zur Datendarstellung. Öffnen Sie hierzu MATLAB und stellen Sie das Arbeitsverzeichnis (*Current Directory*) auf den Ordner view\ ein. Das Skript und die Logdatei sollten sich beide im eingestellten Arbeitsverzeichnis befinden, daher sollten Sie den Ordner view\ als Speicherort für die Logdatei im NXT Charting Logger wählen (siehe oben).

Das MATLAB-Skript zur Datendarstellung ist in Abschnitte untergliedert. Die Nutzerschnittstelle befindet sich im Abschnitt *User inputs*. Passen Sie an der entsprechenden Stelle den Dateinamen an, um die soeben mit dem NXT-Beispielprogramm erstellte Logdatei auswerten zu können. Die Datendarstellung erfolgt sowohl modulspezifisch als auch spezifisch in Bezug auf den Nutzerlogdatentyp (Tabellenzeile oder Zeichenzeile). Um die vom Control-Modul geschriebenen Tabellenzeilen darzustellen, muss an der entsprechenden Stelle folglich *[0 0]* stehen.

Führen Sie nun das MATLAB-Skript aus. Aus den einzelnen Tabellenzeilen generiert MATLAB eine Tabelle in Gestalt einer Matrix. Die erste Spalte beinhaltet die Zeitstempel. Die weiteren Spalten repräsentieren die vom Nutzer im Quelltext des NXT-Programms definierten Spalten, welche somit jeweils den Variablenwert zum zugehörigen Zeitstempel enthalten. Die entsprechende Matrix speichert das Skript in der Variable *output*. Außerdem visualisiert das Skript jede vom Nutzer definierte Spalte in Form eines Plots (Variablenwerte über der Zeit).

Ändern Sie den Abschnitt *User Inputs* nun so, dass die Zeichenzeilen des Control-Moduls dargestellt werden. Die Darstellung der entsprechenden Zeichenzeilen erfolgt dann analog zur Darstellung in der Textdatei in der MATLAB-Konsole.

10.4.4 FEHLERBEHANDLUNG

In bestimmten Fällen kann es bei der Verwendung der Datenlogging-Funktion der Monitor-Schnittstelle zu Fehlern kommen. Diese Fehler können sowohl das Format der Logeinträge im NXT Charting Logger als auch das Verhalten des Roboters betreffen. **Lesen Sie den folgenden Abschnitt daher besonders aufmerksam durch.**

Verbindungsfehler

- Beim physikalischen Trennen der Verbindung zwischen NXT und PC bei laufendem Datenlogging kann es zu nichtdeterministischem Fahrverhalten des Roboters kommen. **Vorsicht: Die Bewegung des Roboters ist immer zu kontrollieren, um ein Herunterfallen und Ähnliches zu vermeiden!**
- Eine gleichzeitige Verbindung des NXT mit einem PC **und** einem Tablet über Bluetooth ist nicht möglich. Um die Datenlogging-Funktion nutzen zu können, muss der PC über USB mit dem NXT verbunden werden.
- Das Beenden einer Verbindung zwischen NXT und PC muss immer durch das Beenden des auf dem NXT laufenden Programms erfolgen. Bei Verwendung des Buttons *Disconnect* im NXT Charting Logger zum Beenden einer Verbindung kann es zu Fehlern kommen.

Fehler beim Verwenden von Zeichenzeilen

- Bei Verwendung des Zeichens Backslash (\) im Feld „Inhalt“ der Zeichenzeile kann es zu Fehlern bei der Darstellung der Zeile auf PC kommen (Steuerzeichen). Die Datendarstellung mithilfe des MATLAB-Skriptes funktioniert dann gegebenenfalls nicht.

Fehler beim Verwenden von Tabellenzeilen

- Beim Schreiben von Variablenwerten muss sichergestellt werden, dass der übergebene Name exakt mit dem bei der Definition der Spalte verwendeten Namen übereinstimmt.
    - In jedem Modul darf pro Aufruf der jeweiligen Aktualisierungsfunktion(en) insgesamt nur eine Tabellenzeile (bestehend aus einer oder mehreren Spalten) geschrieben werden. Werden mehrere Tabellenzeilen pro Aufruf geschrieben, so wird nur die zuletzt geschriebene Zeile berücksichtigt.
    - Bei besonders intensiver Nutzung der Datenlogging-Funktion kann es dazu kommen, dass nicht alle geschriebenen Tabellenzeilen gesendet werden. Die betreffenden Zeilen fehlen dann in der Logdatei. Die Nutzung der Datenlogging-Funktion sollte beim Auftreten dieses Fehlers auf das Nötigste reduziert werden.

Auftreten eines *Out-Of-Memory-Errors*

- Wenn zu viele und/oder zu lange Zeilen gesendet werden sollen, kann der Speicher des NXT überlaufen. Insbesondere Schleifen im Quelltext sind dahingehend zu prüfen.

Sonstige Fehler

- Falls keine der beschriebenen Vorgehensweisen den Fehler behebt, kann es sinnvoll sein zu überprüfen, ob der Fehler auch bei ausgeschalteter Datenlogging-Funktion auftritt.

Neustart-Empfehlung

- Grundsätzlich sollte das gesamte System (NXT, NXT Charting Logger, Tablet, usw.) nach der Behebung eines der oben beschriebenen Fehler neugestartet werden.