TECHNISCHE UNIVERSITÄT DRESDEN

IfA LS-AT

Fakultät Elektrotechnik und Informationstechnik Institut für Automatisierungstechnik

Dokumentation

zum Thema

Hauptseminar Automatisierungs-, Mess- und Regelungstechnik

vorgelegt von Max Kirchner
im Studiengang Elektrotechnik, Jg. 2017
geboren am 03.05.1998 in Räckelwitz

Betreuer: Dr.-Ing. Carsten Knoll
Dipl.-Ing. Elias Scharf
Dipl.-Ing. Julian Rahm
Dr.-Ing. Andrey Morozov
Dipl.-Ing. Mustafa Saraoglu
Verantwortlicher Hochschullehrer: Prof. Dr. techn. Klaus Janschek
Tag der Einreichung: 12.01.2020

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Analyse der Aufgabenstellung

1.1 Allgemeine Funktionsbeschreibung und Ziele

Das Ziel des Moduls Guidance (deutsch Führung) ist die anderen Module miteinander zu verbinden und sie zu steuern. Damit dies möglich ist, wurde mit den Modulverantwortlichen alle Funktionalitäten besprochen.

Die Hauptfunktion des Roboters ist das Abfahren eines Parcours. Dabei wird nach passenden Parklücken gesucht. Dieser Zustand heißt Scout-Modus. In eine der gefundenen Lücken soll anschließend eingeparkt werden. Mit Hilfe einer Android-App werden Steuersignale übermittelt. Wenn zum Beispiel das Ausparksignal übertragen wird, folgt der Roboter einem Polynom und wechselt anschließend in den Scout-Modus. Gefordert ist zusätzlich, dass von jeder Aktion das Fahrzeug in einen Pause-Modus gewechselt werden kann.

1.2 Geplantes Vorgehen

Um den Missionsplaner und Pfadgenerator zeit effektiv zu implementieren, wurde als erster Schritt eine theoretische Vorbetrachtung getätigt. Dabei wurde ein Zustandsdiagramm mit entsprechenden Aktionstabellen und die Berechnungsvorschrift für das Pfadpolynom entworfen.
Für den Entwurf der Zustandsmaschine wurde die online Software Lucidchart verwendet. Mit dieser ist es möglich Diagramme zu zeichnen.

Anschließend wurden die Modelle implementiert. Beim Programmieren sollte regelmäßig der aktuelle Stand in ein „GitHub Repository“ geladen werden, damit die anderen Module ihre Funktionen testen können.

1.3 Schnittstellen und Zusammenarbeit zu/mit anderen Modulen/Modulverantwortlichen

Das Projekt wurde in der objektorientierten Programmiersprache Java umgesetzt. Dies ermöglicht eine klare Modultrennung. Damit unsere Gruppe eine Versionsverwaltung besitzt, entschieden wir uns für ein „GitHub Repository“.

Mit Hilfe von Setter-Methoden ist es möglich festzulegen, welche Funktionen in den entsprechenden Modulen ausgeführt werden sollen. Im Gegenzug können mit Getter-Methoden aktuelle Eigenschaften des Roboters abgefragt werden. Mit diesen Informationen werden die jeweiligen Zustandsübergänge überprüft.

Unsere Gruppe hat sich regelmäßig getroffen. Dabei wurden Etappenziele gesetzt und wichtige Änderung besprochen. Zum Beispiel sollte bis Ende 2019 alles implementiert sein, damit in den letzten zwei Wochen ein ausführlicher Funktionstest durchgeführt werden kann und alle auftretenden Fehler beseitigt werden können.

Beim Programmieren war es auch oft notwendig zusammen zu arbeiten, weil man allein nicht alles überschauen konnte. Vor allem die Demoprogramme zwei und drei wurden zusammen mit der Control implementiert. Ich programmierte alle notwendigen Methodenaufrufe und Zustandswechsel. Der Controlverantwortliche konnte seine Regler testen und anpassen, sodass alle Manöver abfahrbar waren.

Durch die Implementierung der Demoprogramme war es möglich einen guten Überblick über die Programmteile der anderen Module zu bekommen. Es ist somit möglich mein Programmentwurf stückweise zu testen. Auftretende Fehler in allen Modulen konnten entdeckt werden. Vor allem jene, die erst beim Zusammenspiel mit allen Modulen auftraten. Durch die Zusammenarbeit wurden diese schnell behoben.

2 Entwurf Missionsplaner

2.1 Geforderte Funktionen

Die geforderten Funktionen können in fünf Kategorien eingeteilt werden.

i Scout-Modus mit Parklückensuche

ii Einparken

iii Parken

iv Ausparken

v Inaktiv sein

Diese Kategorien definieren die Hauptzustände.

Was das Fahrzeug in jener Kategorie machen soll, ist in den Unterzuständen definiert.

Scout-Modus bedeutet, dass das Fahrzeug einen Parcours abfährt. Dabei wird einer Linie gefolgt. Wenn eine Ecke detektiert wird, muss die Geschwindigkeit reduziert werden. Das Ziel dabei ist, dass die Regelung genauer arbeiten kann und die Kurve abgefahren wird. Während dem Abfahren sucht der Roboter nach Parklücken.

In dem Zustand „Park_This“ wird als erstes die Startpose angefahren. Danach wird ein Polynom abgefahren. Wenn die Zielpose in der Lücke erreicht ist, wechselt der Automat automatisch in den Zustand „Park“. In diesem wird gewartet, bis über die App das Ausparksignal übermittelt wird.

Beim Ausparken folgt der Roboter erneut einem Polynom zurück auf die Linie. Dabei muss sichergestellt werden, dass nach vorne genügen Freiraum vorhanden ist, sonst gibt es einen Unfall mit der Bande.

Vor allem für das Ein - und Ausparken müssen neben den Hauptzuständen die jeweiligen Unterzustände abgespeichert werden, damit beim Fortsetzen die richtigen Aktionen ausgeführt werden. Bei den Abfahrtspolynomen muss beachtet werden, dass kein neues Polynom berechnet wird.

2.2 Hauptzustandsmaschine

Für den Entwurf der folgenden Zustandsmaschinen wurde sich an den geforderten Funktionen orientiert. Dabei wurde beachtet, dass die Aufteilung so getätigt wird, dass die Zustandsübergange möglichst einfach sind.

Zustand	Eingangsaktion	Nominalaktion	Ausgangsaktion
Driving	Auswahl des Controlmodus	siehe Unterzustandsmaschine	Parklückensuche ausschalten
	Parklückensuche einschalten (Ausnahme bei der Anfahrt)		
Inactive	Setzen des Controlmodus „INACTIVE“	siehe Unterzustandsmaschine	
	Abspeichern des Vorzustandes		
Exit		Abschalten des Systems	
Park	Auswahl Controlmodus „INACTIVE“	Ausgabe Sensordaten (Abstand nach vorn und hinten, Position)	
Park This	Auswahl des richtigen Subzustandes	siehe Unterzustandsmaschine	
	Parkplatzinformation abfragen		
	Variable „anfahrt“und „correct“auf falsch setzen		
Park Out	Auswahl des richtigen Subzustands	siehe Unterzustandsmaschine	
	Variable „back“		

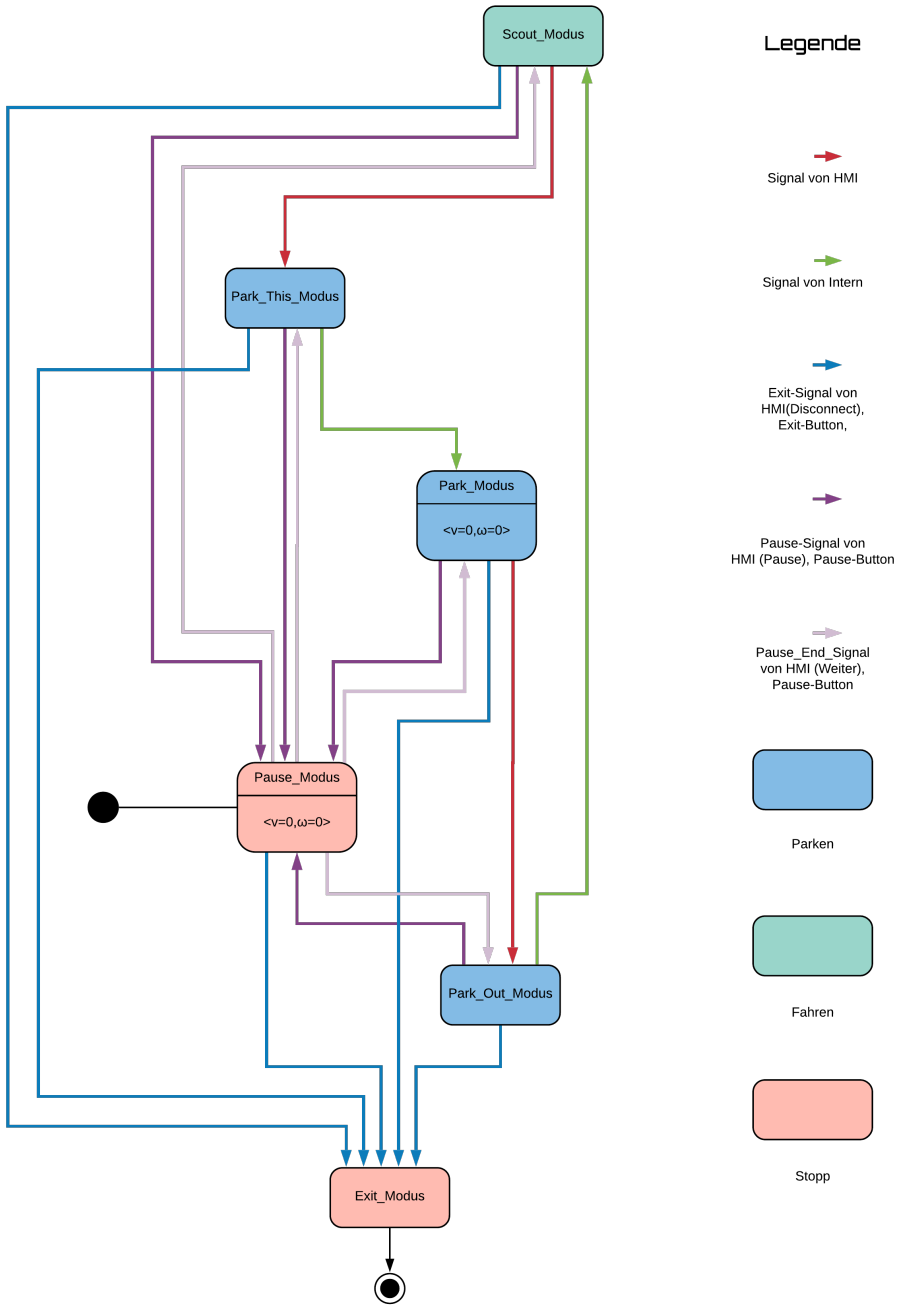

Abbildung 2.1: Entwurf Hauptzustandsautomat

2.3 Unterzustandsmaschine

2.3.1 Scout-Modus

Zustand	Eingangsaktion	Nominalaktion	Ausgangsaktion
Slow	Controlmodusauswahl „SLOW“		
Fast	Controlmodusauswahl „FAST“		

2.3.2 Pause

Es gibt keine Unterzustandsmaschine.

2.3.3 Exit

Es gibt keine Unterzustandsmaschine.

2.3.4 Park

Es gibt keine Unterzustandsmaschine.

2.3.5 Park This

Zustand	Eingangsaktion	Nominalaktion	Ausgangsaktion
To Slot	Zielwinkelberechnung	Zustand „DRIVING“bis Pose erreicht ist	
	Anfahrort festlegen		
Reached Slot	Start- und Endpose vom Polynom festlegen		
	Geschwindigkeit festlegen		
	Setzen des Controlmodus „PARK_CTRL“	Überprüfung ob Polynom abgefahren wurde	
In Slot		Variable „back“auf wahr setzen	

2.3.6 Park Out

Zustand	Eingangsaktion	Nominalaktion	Ausgangsaktion
Backwarts	Berechnung der Rückfahrdistanz	Überprüfung ob das Zurückfahren erfolgt ist	
	Setzen des Controlmodus „SETPOSE“		
ParkOut	Start- und Endpose vom Polynom festlegen	Überprüfung ob Polynom abgefahren wurde	
	Geschwindigkeit festlegen		
	Setzen des Controlmodus „PARK_CTRL“		

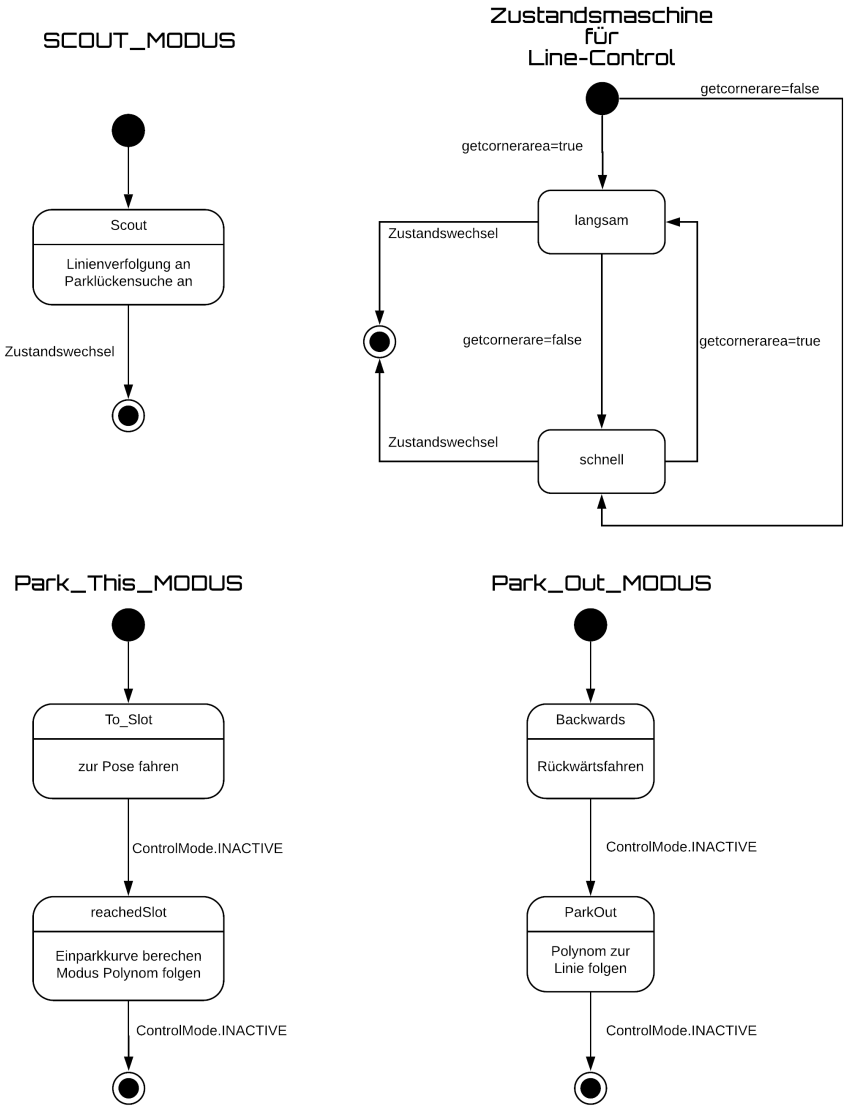

Abbildung 2.2: Entwurf Unterzustandsmaschine

3 Entwurf Pfadgenerator

3.1 Geforderte Funktionen

Damit der Roboter einparken kann, muss er einem Polynom folgen. Dabei ist zu beachten, dass für jede Parklücke individuell ein Polynom entwickelt werden muss.

In dem Modul Guidance wird die Start- und Endpose festgelegt. Anschließend muss der entsprechende Controlmodus gestartet werden. Die Berechnung des Polynoms wird danach im Control-Modul aufgerufen.

3.2 Mathematische Beschreibung

Eine einfache und gute Einparkkurve entspricht einer Funktion dritten Grades:

$$f = y(x) = a * x^3 + c * x^2 + b * x + d \tag{3.1}$$

Die Funktion soll durch den Koordinatenursprung gehen ($d = 0$) und punktsymmetrisch sein (nur ungerade Exponenten $\Rightarrow c = 0$).
Das Problem vereinfacht sich auf folgendes Polynom:

$$f = y(x) = a * x^3 + b * x \tag{3.2}$$

Es werden zwei Bedingungen benötigt, damit die Koeffizienten berechnet werden können.

$$y'_{ende} = f'(x_{ende}) = 3 * a * x^2_{ende} + b = 0 \tag{3.3}$$

$$y_{ende} = a * x^3_{ende} + b * x_{ende} \tag{3.4}$$

Durch das Umstellen folgt:

$$a = -\frac{y_{ende}}{2 * x_{ende}^3} \tag{3.5}$$

und

$$b = -3 * a * x_{ende}^3 \tag{3.6}$$

Damit die Brechung stets auf das selbe Problem zurückzuführen ist, gibt es einen globales und lokales Koordinatensystem. Wie in der Abbildung 3.1 zu sehen, ist der Mittelpunkt zwischen Start- und Zielpose im globalen Koordinatensystem identisch mit dem Ursprung des lokalem Systems. Dieses ist je nach Richtungswinkel des Roboters entsprechend gedreht.

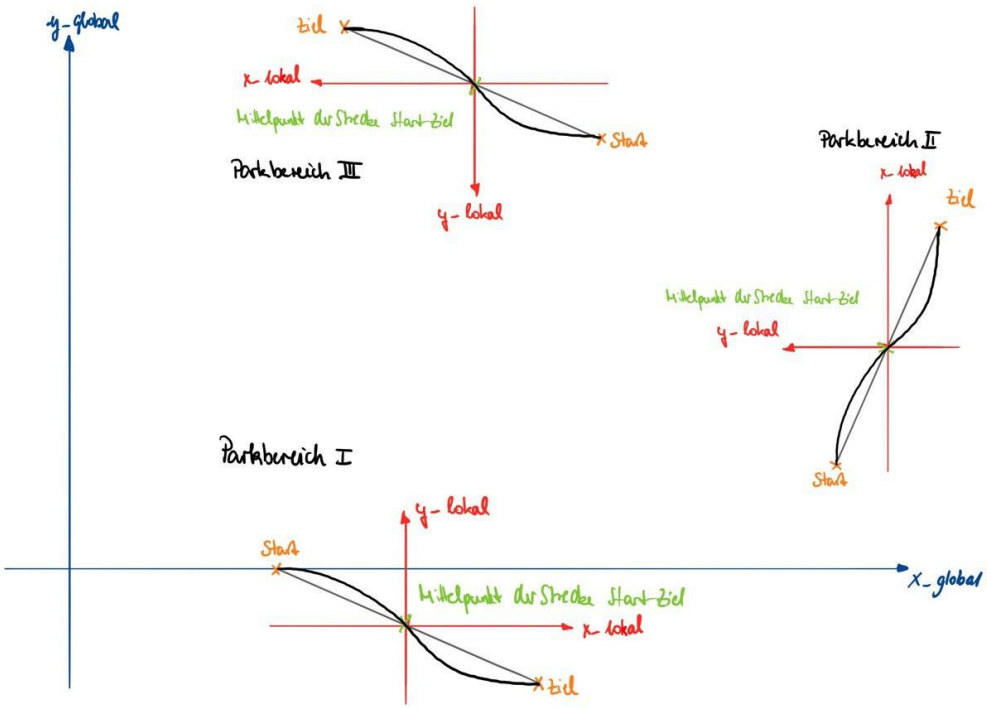

Abbildung 3.1: Skizze für die Parklückenberechung

4 Implementierung

4.1 Missionsplaner

Ausgangspunkt für die Implementierung ist das gegebene Beispielprogramm. In diesem wurde die Hauptzustandsmaschine mit drei Grundzuständen (DRIVE, INACTIVE und EXIT) vorgegeben. Mit Hilfe von Switch-Case-Abfragen wird nur der Code des entsprechenden Zustandes ausgeführt. Nach der Abfrage wird überprüft, ob in einem neuen Zustand gewechselt werden muss.

Die Grundversion verfügt noch über keine Unterzustandsmaschinen. Es fehlen auch die wesentlichen Zustände für das Ein- und Ausparken. Der getätigte Entwurf des Missionsplaners wurde mit weiteren Switch-Case-Abfragen implementiert.

Es stellte sich heraus, dass kleine Änderungen am Entwurf noch korrigiert werden musste. Durch die Überarbeitung des Modells wurde der Roboter effizienter gestaltet, sodass vor allem das Ein- und Ausparken beschleunigt werden konnte. Zum Beispiel musste man keine Routine schreiben, welche die Linie wiederfindet, da das Abfahren der Trajektorie sehr genau funktioniert.

4.2 Pfadgenerator

Zusammen mit dem Controlmodul wurde entschieden, dass im Guidancemodul die Start- und Zielpose, die Geschwindigkeit übergeben wird und zum Schluss der richtige Controlmodus aktiviert wird. Alle Berechnungen werden im Controlmodul getätigt.

Damit das Polynom richtig berechnet wird und anschließend dieses abgefahren werden kann, muss zwischen dem lokalem und globalem Koordinatensystem transformiert werden. Wenn das Fahrzeug im Parkbereich eins einparken muss, wie in Abbildung 3.1 zusehen ist, muss nichts geändert werden. Im Parkbereich zwei müssen die x- und y-Achse vertauscht werden. Zusätzlich ist zu beachten,

dass die y-Achse gespiegelt werden muss. Im Parkbereich drei müssen beide Achsen gespiegelt werden. Das Tauschen kann mit Hilfe einer Zwischenspeichervariable einfach realisiert werden. Falls eine Achse gespiegelt werden muss, wird sie mit einer -1 multipliziert.

5 Anmerkungen und Verbesserungsmöglichkeiten

Im späteren Berufsleben wäre es möglich Zustandsmaschinen anstatt mit einer Programmiersprache mit Hilfe von Fachsprachen nach IEC 1131 zu entwerfen. Jedoch kann mit Programmiersprachen die selben und viele weiter Probleme gelöst werden.

Der Roboter fährt auf dem Parcours schon sehr genau und stabil. In seltenen Fällen können noch Fehler auftreten, welche noch nicht mit Hilfe einer Exception ausgewertet werden. Es könnten noch weiter Fehlerabfangroutinen hinzugefügt werden.

Das HMI-Modul versuchte außerdem das Problem der graphischen Parklückenüberschreitung in der App zu lösen. Dabei wurde festgestellt, dass beim Ausparken die Parklücke falsch vermessen wird. Der Roboter vermisst nach dem Ausparken nicht die komplette Parklücke von Anfang an, sondern von seiner aktuellen Pose. Um dies zu umgehen, müsste implementiert werden, dass er erst nach der aktuellen Parklücke anfängt mit messen.

Da unser Roboter ein Modell ist, könnte nun versucht werden, diese Technik des Einparkens in Parkhäusern und Parkplätzen für ein echtes Fahrzeug zu realisieren. Bekannte Automobilfirmen haben dieses Problem schon gelöst. Dies bedeutet, dass moderne Autos in der Lage sind einen freien Parkplatz zu detektieren und anschließend können sie in diesen einparken.

Selbstständigkeitserklärung

Hiermit versichere ich, Max Kirchner, geboren am 03.05.1998 in Räckelwitz, dass ich die vorliegende Dokumentation zum Thema

Hauptseminar Automatisierungs-, Mess- und Regelungstechnik

ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Bei der Auswahl und Auswertung des Materials sowie bei der Herstellung des Manuskripts habe ich Unterstützungsleistungen von folgenden Personen erhalten:

Dr.-Ing. Carsten Knoll, Dipl.-Ing. Elias Scharf, Dipl.-Ing. Julian Rahm, Dr.-Ing. Andrey Morozov, Dipl.-Ing. Mustafa Saraoglu

Weitere Personen waren an der geistigen Herstellung der vorliegenden Dokumentation nicht beteiligt. Mir ist bekannt, dass die Nichteinhaltung dieser Erklärung zum nachträglichen Entzug des Diplomabschlusses (Masterabschlusses) führen kann.

Dresden, den 12.01.2020 .....

Unterschrift