KOOPERATIVE STEUERUNG VON MODELLVERSUCHSFAHRZEUGEN

ENTWICKLUNG EINES INTELLIGENTEN FAHRENTSCHEIDERS

Softwareentwicklungspraktikum
Sommersemester 2008

Grobentwurf zum System

E C A R

Auftraggeber
Technische Universität Braunschweig
Institut für Betriebssysteme und Rechnerverbund
Prof. Dr.-Ing. Lars Wolf
Mühlenpfordstrasse 23
38106 Braunschweig

Betreuer: Kai Homeier, Carina Flämig
Phasenverantwortlicher: Rayan Merched El Masri

Auftragnehmer

Name	E – Mail
Wira Kakar	wirak@web.de
Chen Zhiwei	czwnii@hotmail.com
Rayan Merched El Masri	rayan_masri@hotmail.com
Ekrem Enes Duman	dumanenes@hotmail.de
Günter Dusch	g.dusch@Googlemail.com

Braunschweig, 08.05.2008

Versionsübersicht

Version	Datum	Autor	Status	Kommentar
1.0	02.05.08	Alle Auftragsnehmer		
2.0	08.05.08	Alle Auftragsnehmer		Korrektur: Statechart, Aktivitätsdiagramme und Komponentendiagramm, Reduzierung auf vier Produktfunktionen, Verbesserung hinsichtlich Sprache, Grammatik und Rechtschreibung

Inhaltsverzeichnis

A b b i l d u n g s v e r z e i c h n i s

1 Einleitung

Das SEP 2008 im IBR beschäftigt sich mit der "kooperativen Steuerung von Modellversuchsfahrzeugen". Auf dem bestehenden Modellversuchsfahrzeug des IBR soll ein kooperativer, intelligenter Fahrentscheider entwickelt werden. Für das auf dem Fahrzeug installierte mini-ITX Board wird ein Softwaresystem zur Bildverarbeitung, Positionsbestimmung und für die Fahrstrategie implementiert.
In einem unbekannten Labyrinth sollen bestimmte Objekte gefunden werden (z.B. ein Ball). Zur Orientierung ist es notwendig vorhandene Baken anzupeilen. Zur Lösung dieses Problems hat sich die Gruppe ECAR folgendes überlegt.
Es existiert eine Hauptklasse, die sämtliche Informationen über das Modellversuchsfahrzeug, das gesuchte Objekt und die Baken enthält. Bevor das System startet, wird der Klasse bekannt gegeben wie das gesuchte Objekt aussieht bzw. welche Farbe und Form es hat. Wo die Baken stehen und wie das Modellversuchsfahrzeug zu reagieren hat, wenn er die Baken sieht, ist der Klasse ebenfalls bekannt. Die Hauptklasse ist dafür zuständig das System zu initialisieren, dem Modellversuchsfahrzeug bei Entscheidungsaufgaben zu helfen, den Weg, der das Modellversuchsfahrzeug zurücklegt zu speichern, die Bilder, die die Kamera aufnimmt zu analysieren und die Position des Modellversuchsfahrzeug zu bestimmen
Es sind Sensoren an dem Modellversuchsfahrzeug installiert, die den Weg auf Hindernisse abtasten, z.B. die Wände des Labyrinths, das gesuchte Objekt oder andere Gegenstände, die auf dem Weg des Modellversuchsfahrzeug liegen.
Die Kamera sorgt dafür, dass alle Gegenstände, die auf dem Weg des Modellversuchsfahrzeugs liegen, aufgenommen werden. Wie oben schon genannt, werden diese Bilder dem Prozessor gesendet, die daraufhin vom Prozessor analysiert werden.
Der Benutzer spielt eine beobachtende Rolle. Er betätigt die Starttaste und beobachtet, ob die von ihm entworfene Software funktioniert und ob es zu Störungen kommt oder nicht.

1.1 Projektdetails

Wenn das System initialisiert wird, dann ist das System bereitgestellt. Sobald die Starttaste getätigt wird, fängt das Modellversuchsfahrzeug seinen Weg abzutasten. Auf dem Weg kann ein Hindernis kommen so ändert das Modellversuchsfahrzeug seinen Weg zu ändern und tastet einen neuen Weg in eine neue Richtung. Falls auf dem Weg des Modellversuchsfahrzeugs keine Hindernisse gibt, so fährt das Fahrzeug weiter. Während des Fahrens, kann es das gesuchte Objekt identifizieren, somit wäre die Suche beendet. Es kann auch passieren, dass es alle Wege besucht und das gesuchte Objekt nicht findet, so wäre die Suche ebenfalls beendet.

Während das Modellversuchsfahrzeug seinen Weg abtastet, könnte es auch Baken identifizieren, womit die Position des Fahrzeugs bestimmt werden kann und gleichzeitig wird der Weg des Fahrzeugs gespeichert. Hiernach wird der Weg wieder abgetastet und die aufgenommenen Bilder der Kamera analysiert.

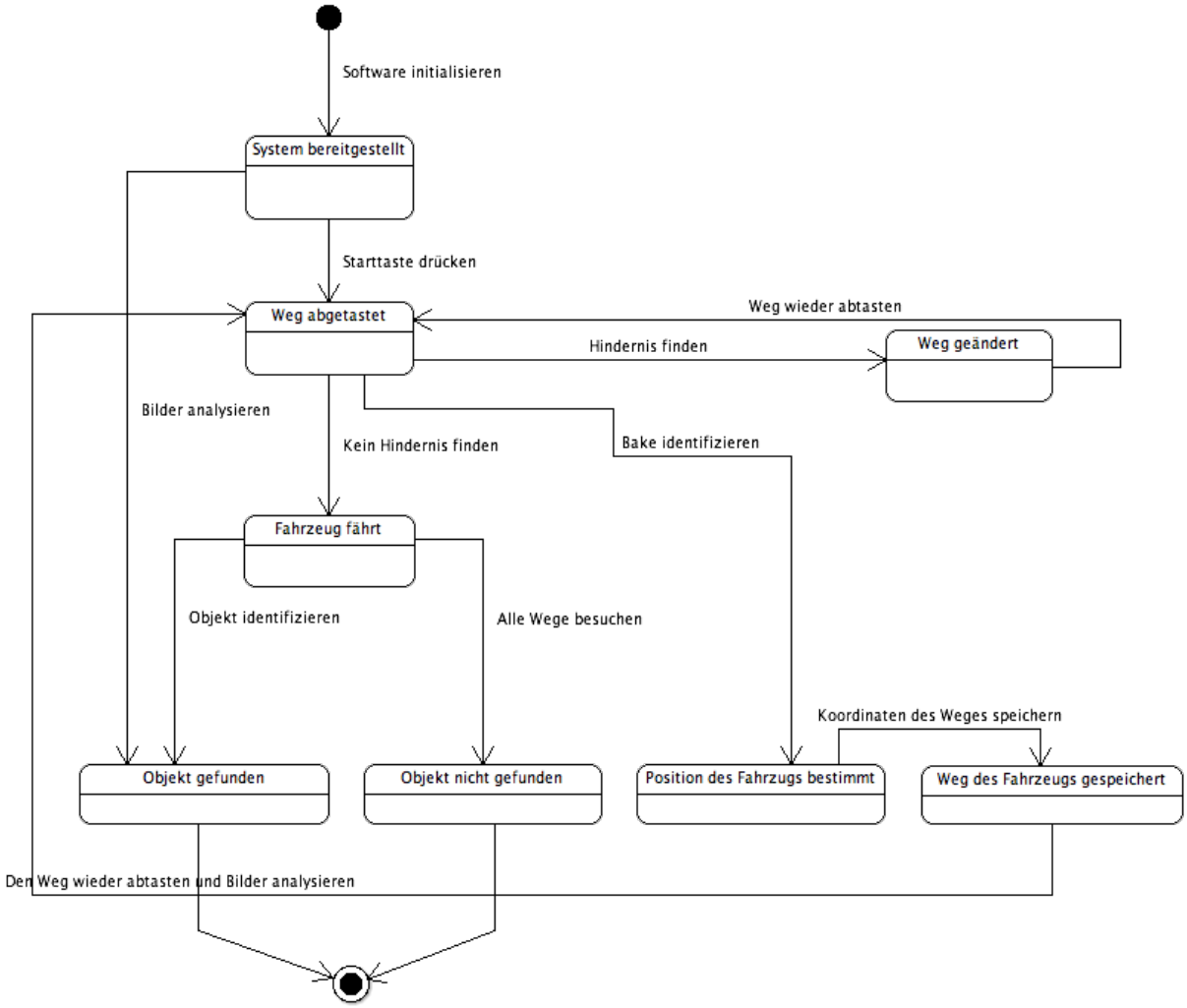

Abbildung 1: StateChart Projektdetails 1

2 Analyse der Produktfunktionen

2.1 Analyse von Funktionalität /F10/: System initialisieren

Ziel dieser Funktionalität ist die Initialisierung des Gesamtsystems. Hier wird das Fahrzeug mit Strom versorgt und das von uns entwickeltes Programm auf dem Board über eine Schnittstelle geladen.

2.1.1 Grobanalyse

Um das System zu initialisieren, soll der Benutzer den Strom einschalten und dann die C++-Dateien auf dem Board laden. Der C++-Compiler, der sich auf dem Board befindet, kompiliert die geladene C++-Quellcodes. Ist die Kompilierung erfolgreich beendet, steht das System betriebsbereit. Ist das nicht der Fall, kann das System nicht gestartet werden.

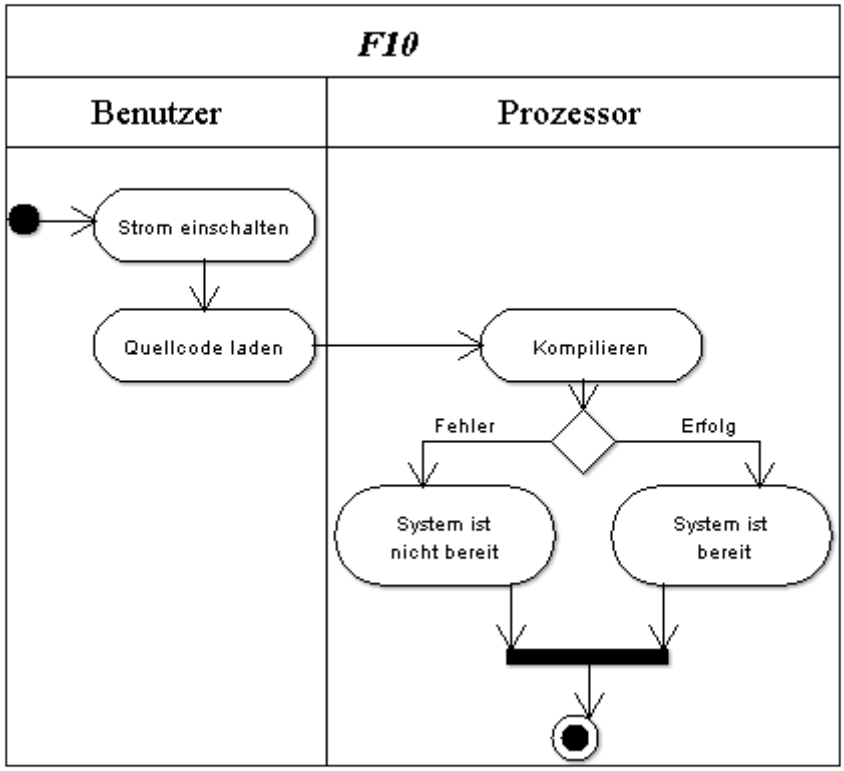

Abbildung 2: Verteilung von „System initialisieren“

2.1.2 Feinanalyse

Der Benutzer greift aktiv auf das System zu. Er schaltet den Strom ein und lädt die C++-Quellcodes auf dem Board. Dann kompiliert der C++-Compiler die Quellcodes. Nur wenn die Kompilierung mit Erfolg beendet ist, kann das System bereitgestellt werden.

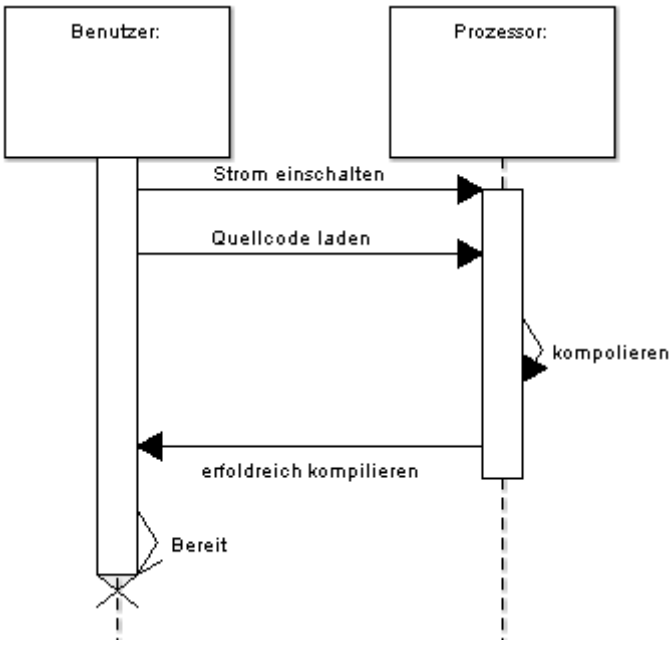

Abbildung 3: Sequenzdiagramm für „System initialisieren“

2.2 Analyse von Funktionalität /F20/: Weg aussuchen

Mit Hilfe der Sensoren wird hier durch den Prozessor eine Wegentscheidung getroffen. Ziel dieser Funktion ist das Finden eines neuen noch nicht besuchten Weges und das Vermeiden der Kollision mit den Hindernissen.

2.2.1 Grobanalyse

Der Sensor tastet den Weg ab, ob es vor dem Modellversuchsfahrzeug ein Hindernis gibt. Wenn kein Hindernis gefunden wird, fährt er geradeaus solang der Weg frei und vorher nicht abgefahren ist, ansonsten muss das Modellversuchsfahrzeug seine Richtung in einen neuen noch nicht besuchten Weg ändern.

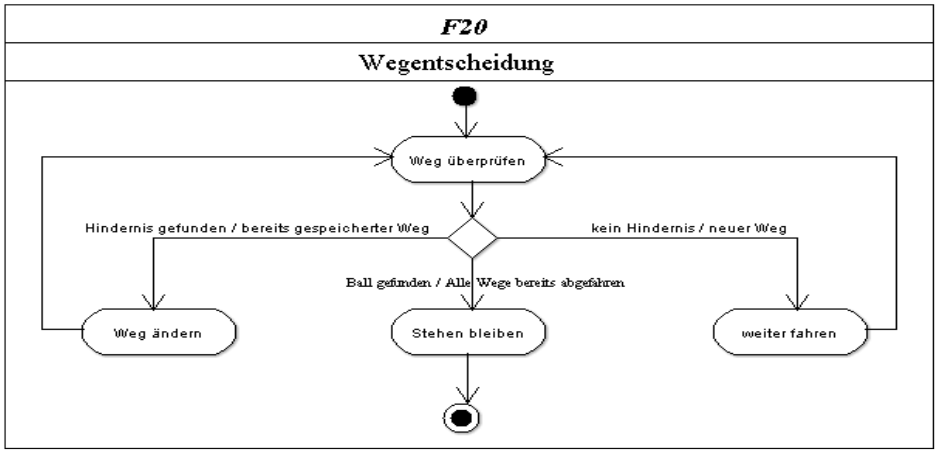

Abbildung 4: Verteilung von „Weg aussuchen“

2.2.2 Feinanalyse

Der Prozessor fragt den Sensor nach dem Abstand zwischen dem Modellversuchsfahrzeug und dem Hindernis ab. Das Modellversuchsfahrzeug darf nicht einen bestimmten Abstand zu einem Hindernis überschreiten. Wenn dieser Abstand erreicht ist, muss das Modellversuchsfahrzeug eine Entscheidung treffen, ob er einen neuen Weg wählt oder ob er stehen bleibt, wenn das Hindernis dem gesuchten Objekt entspricht oder alle Wege bereits besucht wurden.

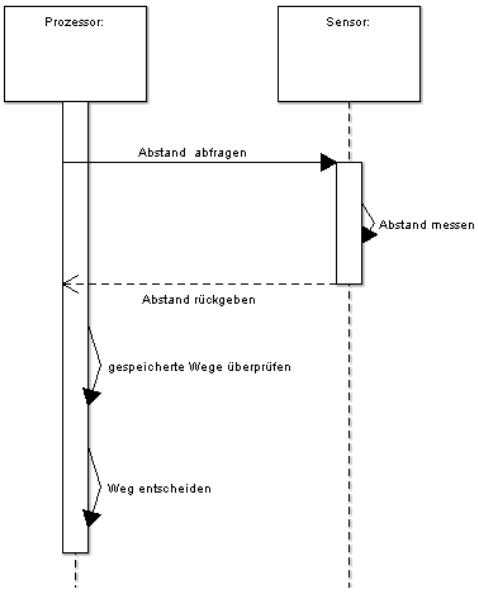

Abbildung 5: Sequenzdiagramm von „Weg aussuchen“

2.3 Analyse von Funktionalität /F30/: Position bestimmen

Eine Entscheidung über bereits besuchte Wege zu treffen, ist nur dann möglich, wenn das Fahrzeug seine aktuelle Position bestimmen kann. Um die Position zu bestimmen, benötigt das Programm Baken. Die Positionen dieser Baken im Labyrinth sind bereits bekannt und sie stehen in verschiedenen Stellen, außerdem müssen sie für die Kamera sichtlich erkennbar sein. Die Verbindung mehrerer Koordinaten erstellt einen Weg. Der Weg wird gespeichert um zu vermeiden, bereits besuchte Wege wieder zu besuchen. Damit spart das Modellversuchsfahrzeug Zeit.

2.3.1 Grobanalyse

Die Kamera nimmt die Bilder auf und liefert sie über eine Schnittstelle dem Prozessor, der diese Bilder analysiert und nach Baken sucht. Findet er die erste Bake, misst er den Winkel mit der Bake. Die Sensoren liefern dem Prozessor die gemessenen Werte. Der Prozessor wiederholt die gleiche Prozedur und sucht die zweite Bake (was vielleicht eine leichte

Richtungsänderung erfordert). Nachdem der Prozessor die beiden Baken erkennt, führt er eine Berechnung durch und bestimmt er die Position des Fahrzeugs. Nach jeder Positionsbestimmung werden die Koordinaten gespeichert. Durch die Verbindung dieser Koordinaten entsteht einen Weg.

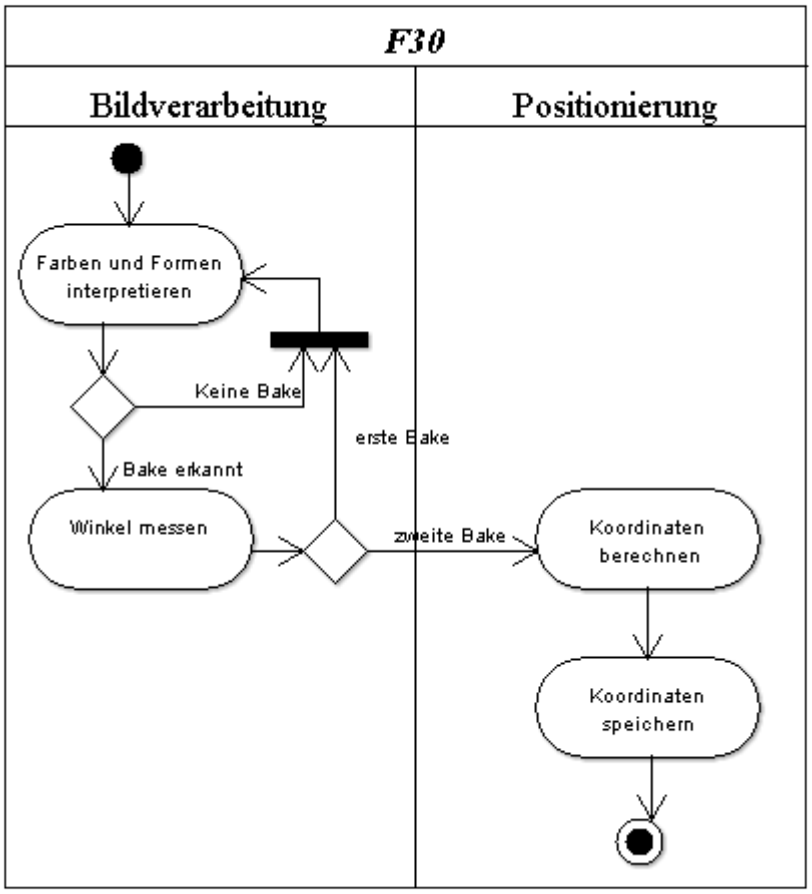

Abbildung 6: Verteilung von „Position bestimmen“

2.3.2 Feinanalyse

Drei Funktionen werden in diesem Abschnitt verdeutlicht, die die Position des Fahrzeugs bestimmen. Die Webkamera nimmt Bilder auf, der Prozessor analysiert die Farben und Formen der vorliegenden Objekte und vergleicht sie mit den Formen und Farben der Baken. Und die Daten der Baken werden in den Quellcodes vorgegeben. Werden keine Baken erkannt, geht die Suche weiter. Stimmen die Baken-Daten mit den Bildern überein, misst die Kamera den Winkel mit der Bake. Um die Position zu bestimmen sind aber zwei Baken erforderlich, daher wird die Prozedur erneut durchgeführt. Wenn die zweite Bake gefunden wird, kann der Prozessor die Position des Fahrzeugs bestimmen. Nach der Positionsbestimmung werden die Koordinaten gespeichert. Die Verbindung mehrerer Koordinaten erstellt einen Weg.

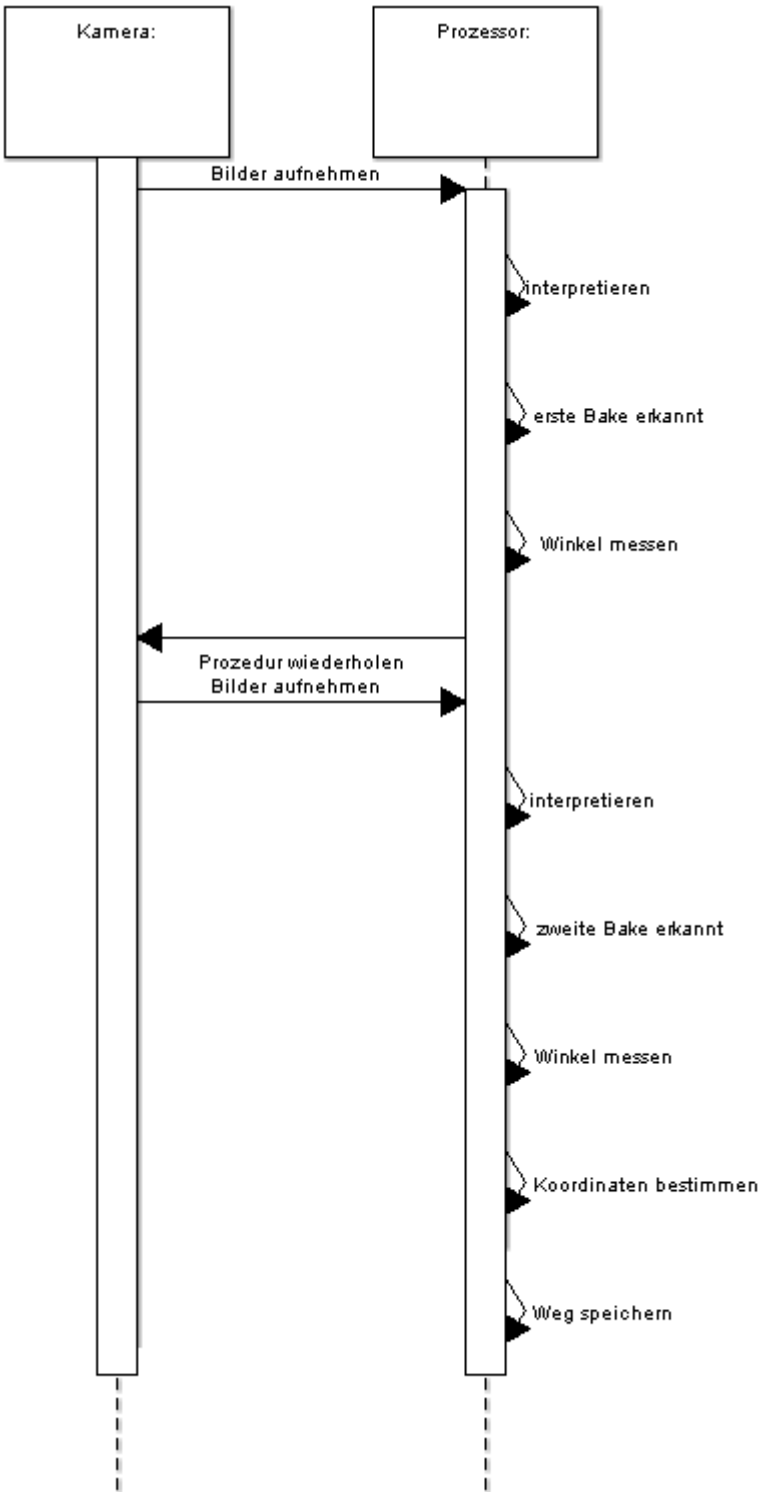

Abbildung 7: Sequenzdiagramm von „Position bestimmen“

2.4 Analyse von Funktionalität /F40/: Bilder analysieren

Die aufgenommenen Bilder werden an dieser Stelle analysiert, um die Baken bzw. das gesuchte Objekt zu identifizieren.

2.4.1 Grobanalyse

Die Kamera nimmt die Bilder auf, welche anschließend mit Hilfe der Algorithmen der OpenCV-Bibliothek analysiert werden. Falls ein Algorithmus den Ball identifiziert, wird der Suchvorgang beendet. Falls eine Bake von einem Algorithmus erkannt wird, dann wird die Position des Modellversuchsfahrzeugs berechnet und der Suchvorgang mit der neuen und präzisen Position fortgesetzt. Wenn weder der Ball noch eine Bake gefunden wird, dann setzt das System den Suchvorgang ebenfalls bis zur Identifizierung des Balls fort. Das bedeutet, dass in so einem Fall ein neues Bild verwendet wird, um die gesuchten Objekte zu identifizieren, da es durchaus vorkommen kann, dass ungünstige Lichtverhältnisse oder andere Einflussfaktoren, die Identifikation der Objekte erschweren bzw. verhindern.

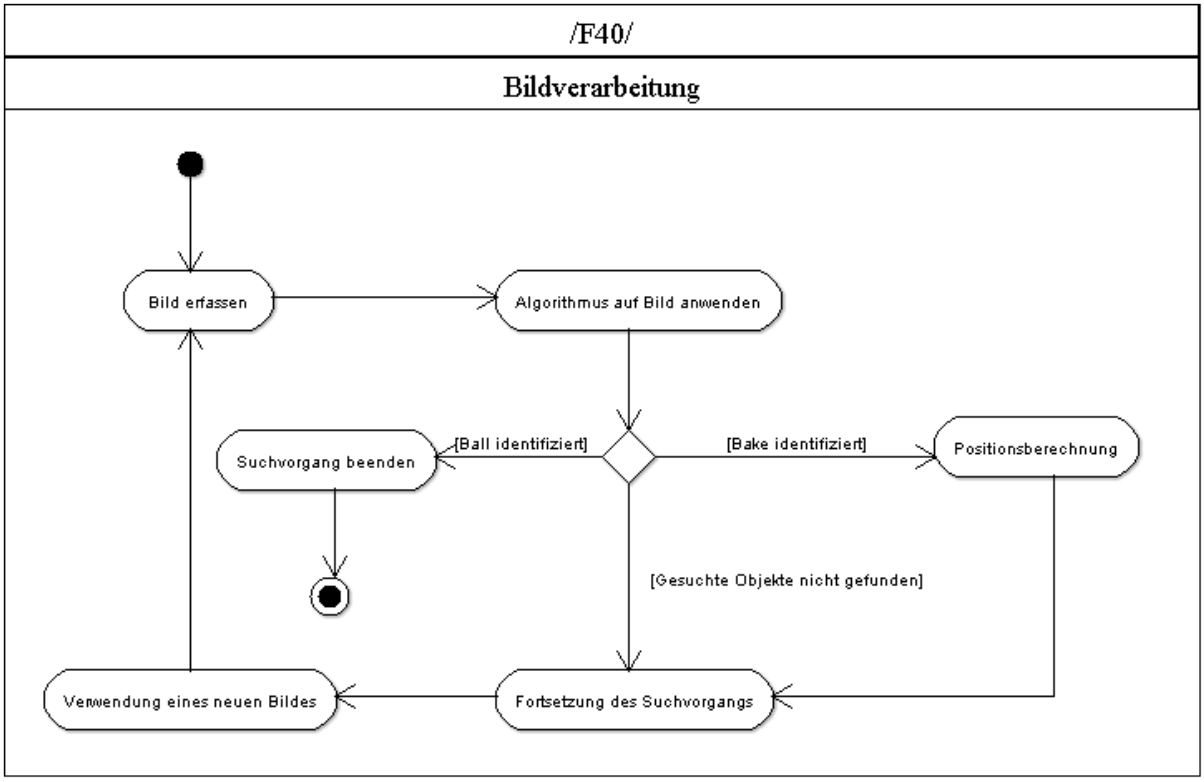

Abbildung 8: Verteilung von „Bilder analysieren“

2.4.2 Feinanalyse

Die Kamera nimmt die Bilder auf, welche anschließend mit Hilfe der entsprechenden Algorithmen bearbeitet werden, so dass der Ball oder die Baken, falls auf den Bildern vorhanden, erkannt werden.

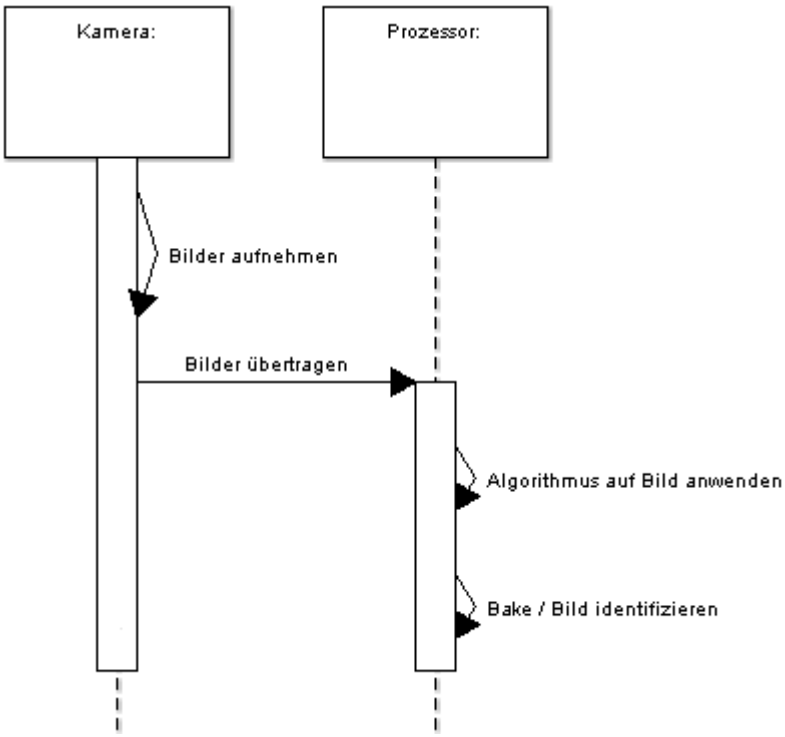

Abbildung 9: Sequenzdiagramm von „Bildern analysieren“

3 Resultierende Softwarearchitektur

3.1 Komponentenspezifikation

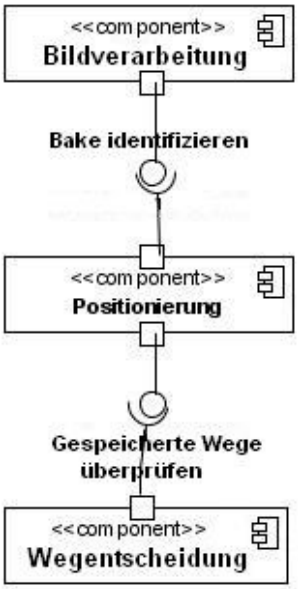

Abbildung 10: Komponentendiagramm

3.2 Schnittstellenspezifikation

Schnittstelle	Aufgabenbeschreibung	
	Operation	**Beschreibung**
/S10/: Bilder importieren	Import(): void	Die aufgenommenen Bilder werden vom Prozessor importiert, wo sie analysiert werden

Schnittstelle	Aufgabenbeschreibung	
	Operation	**Beschreibung**
/S20/: Baken identifizieren	cvMatchTemplate(): boolean	Identifiziert Baken

Schnittstelle	Aufgabenbeschreibung	
	Operation	**Beschreibung**
/S30/: Abstand messen	distance(double value): double	Misst den Abstand zwischen dem Hindernis und dem Modellversuchsfahrzeug

Schnittstelle	Aufgabenbeschreibung	
	Operation	**Beschreibung**
/S40/: Position berechnen	position(double bake1, double bake2): double	Berechnet die Position des Modellversuchsfahrzeugs

Schnittstelle	Aufgabenbeschreibung	
	Operation	**Beschreibung**
/S50/: Weg überprüfen	checkWay(): boolean	Überprüft den Weg, ob dieser Weg bereits besucht wurde oder nicht

3.3 Protokolle für die Benutzung der Komponenten

Die Funktionen von Analyse werden in mehreren Operationen verwendet, z.B. um die Baken oder den Ball zu identifizieren.

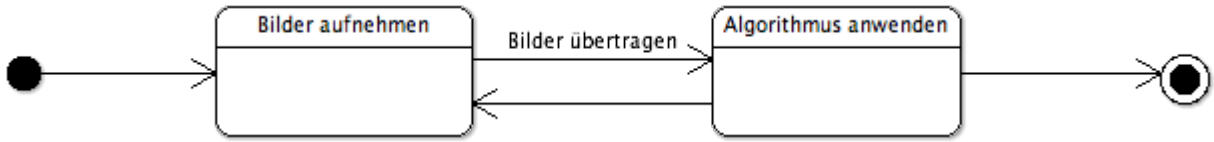

Abbildung 11: Protokollstatechart für „Bilder analysieren“

Die Position wird mehrere Male bestimmt, um den Weg zu speichern und die Position des gesuchten Objekts zu identifizieren.

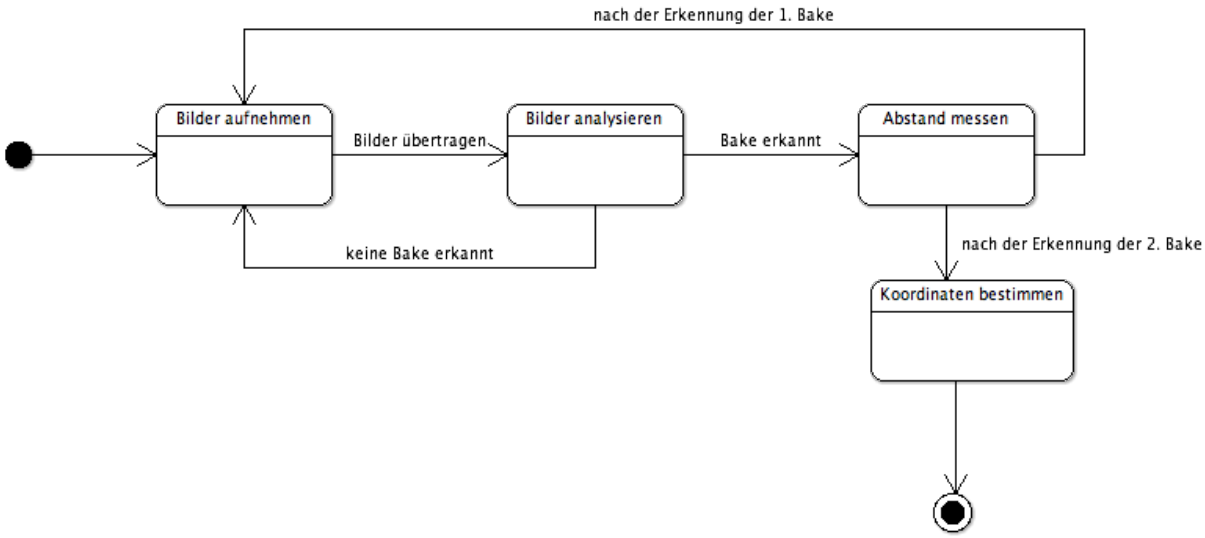

Abbildung 12: Protokollstatechart für „Position bestimmen“

4 Verteilungsentwurf

Das folgende Verteilungsdiagramm verdeutlicht, wie das System auf der vorhandenen Hardware verteilt ist und wie die Kommunikationsbeziehungen zwischen ihnen sind.
Wie in der Abbildung zu sehen ist, gibt es drei Komponenten bestehend aus der Kamera, dem Abstandssensor und dem Prozessor. Die gestrichelten Pfeile verdeutlichen die Beziehungen zwischen diesen Komponenten, z.B. das Analysieren der Bilder ist abhängig davon, wie die Kamera die Videos aufnimmt oder z.B. die Position des Modellversuchsfahrzeugs kann nur bestimmt werden, wenn die Abstandssensoren vorhanden sind und damit der Abstand gemessen werden kann.

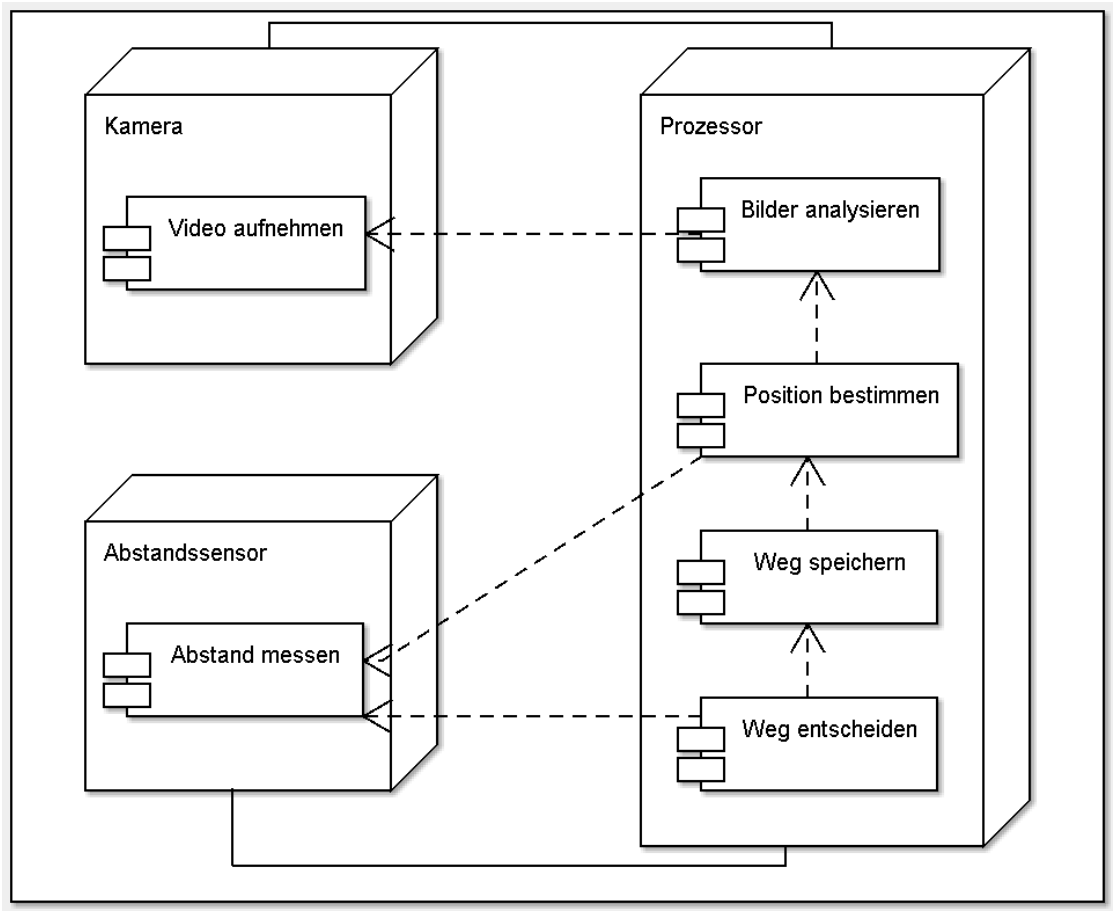

Abbildung 13: Verteilungsdiagramm