AG ROBOTERSYSTEME
FACHBEREICH INFORMATIK
AN DER UNIVERSITÄT KAISERSLAUTERN

Praktikum

Praktikum Spezifikation

Gruppe A

19. Dezember 2006

Inhaltsverzeichnis

1. Praktikum Spezifikation

1.1 Praktikumsteilnehmer

- Robert Becker
- Patrick Fleischmann (Projektleiter)
- Luc Heischbourg
- Jonas Mitschang
- Thomas Pfister
- Marcel Zimmer (Schlüsselverwalter)

1.2 Aufgabenverteilung

Name	**Teilprojekt**
Robert Becker	Mapping, MasterControl
Patrick Fleischmann	Sensors, Camera
Luc Heischbourg	Forkcontrol, VehicleControlSystem
Jonas Mitschang	Mapping, MasterControl
Thomas Pfister	Forkcontrol, VehicleControlSystem
Marcel Zimmer	Sensors, Camera

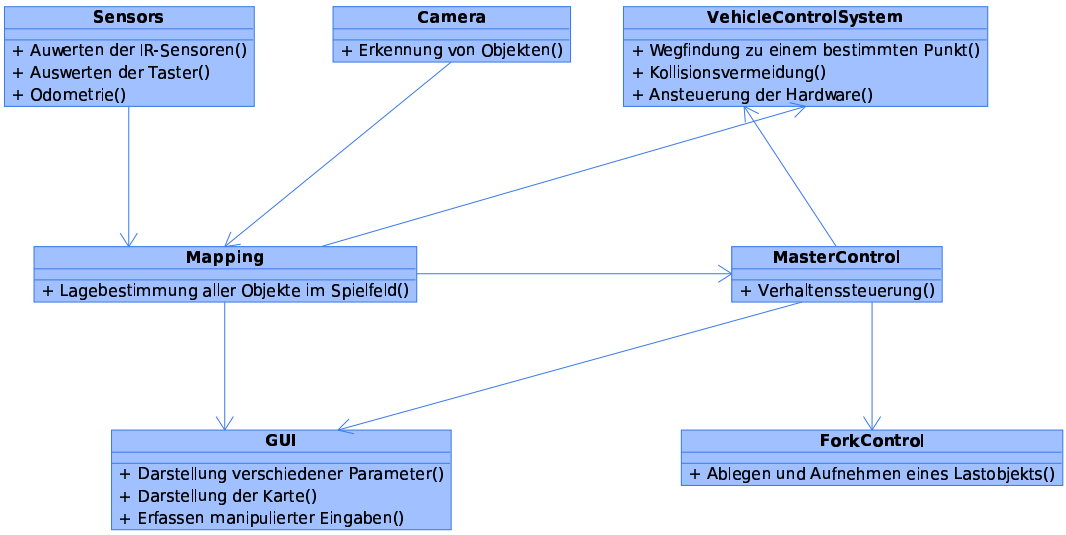

Abbildung 1.1: Übersicht über die Module des Roboters

1.2.1 Zeitplanung und Meilensteine

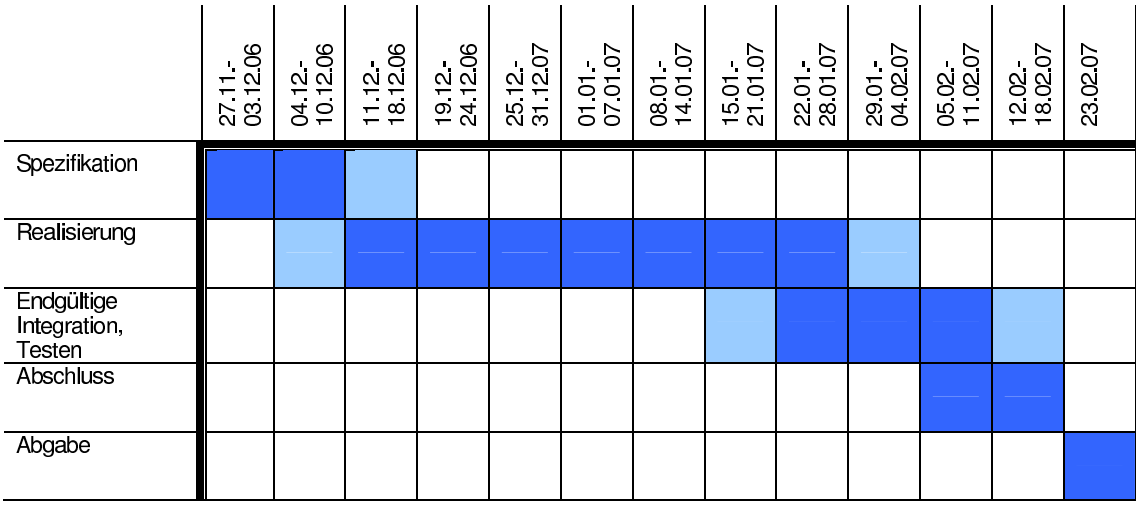

Abbildung 1.2: Zeitplan

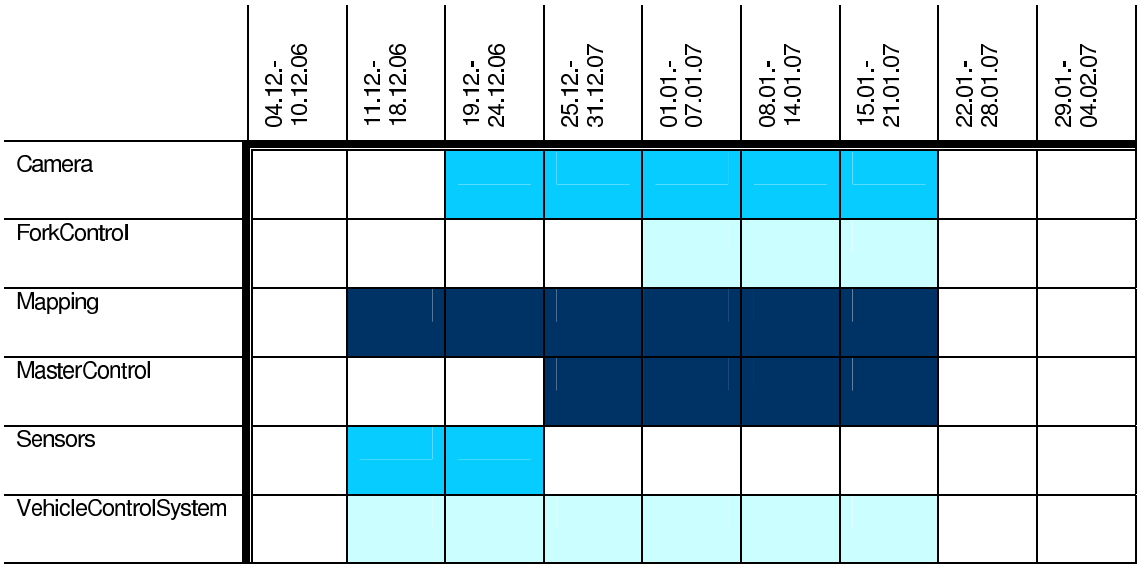

Abbildung 1.3: Zeitplan über die Meilensteine

1.3 Teilprojekte

1.3.1 Camera

Die Komponente **Camera** wertet Bilddaten einer Webcam mit Hilfe von OpenCV und CMVision aus und identifiziert damit Lastobjekte, Hindernisse, gegnerische Roboter und Basen. Zusätzlich wird Ausdehnung und Position der klassifizierten Objekte ermittelt und bereitgestellt.

1.3.1.1 Schnittstellen

Sensor Input

-

Sensor Output

-

Controller Input

-

Controller Output

-

Blackboard

`camera_map` Von der Kamera erkannte Objekte

1.3.2 Forkcontrol

Die Forkcontrol ist verantwortlich für die Bewegung der Gabel, um Lastobjekte aufzuladen, abzuladen, zu sichern und zu stapeln.

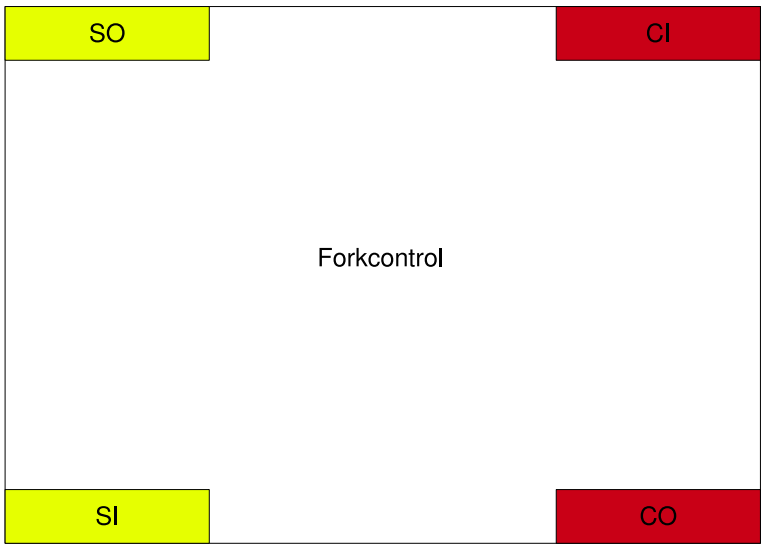

Abbildung 1.4: Aufbau des Modules Forkcontrol in MCA

1.3.2.1 Schnittstellen

Sensor Input

Sensor Output

- `fork_stuck`: Gabelbewegung blockiert

Controller Input

- `raise_fork`: Geschwindigkeit mit der die Gabel nach oben bewegt wird (negativ nach unten)

Controller Output

1.3.3 Mapping

Dieses Modul führt die von der Kamera erkannten Objekte, der aktuellen Position und Ausrichtung des Fahrzeugs sowie den Sensordaten zu einer Karte zusammen, auf der alle dem Gabelstapler bekannten Objekte verzeichnet sind.

Objekte werden eindeutig indiziert und bestimmten Klassen (Lastobjekt, Hindernis, Zielplattform und Gabelstapler) zugeordnet. Des Weiteren wird die absolute Position des Gabelstaplers ausgehend von der relativen Position (Odometrie) sowie den Kameradaten berechnet.

1.3.3.1 Module

MapSync

Synchronisiert die bekannte Karte mit den neu gewonnen Daten aus der Sensorik (Odometrie, Kamera, IAR).

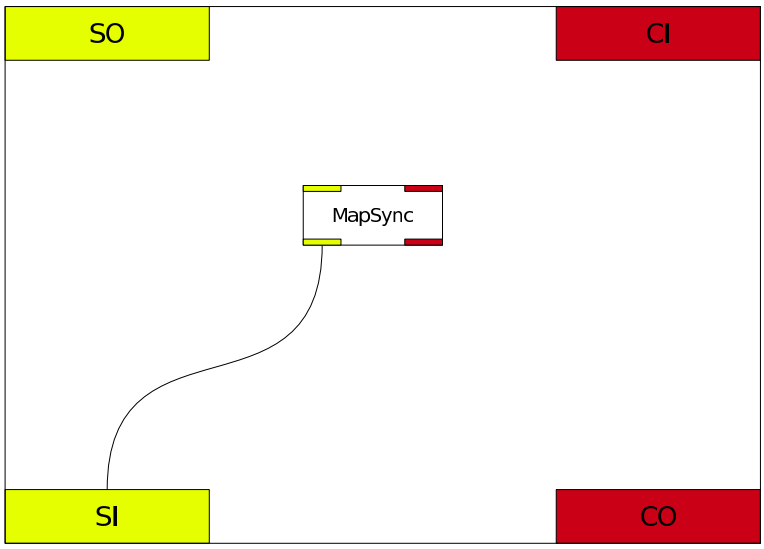

Abbildung 1.5: Aufbau des Modules Sensors in MCA

1.3.3.2 Schnittstellen

Sensor Input

- `DIST_SENSOR[0..5]`: Abstandsmesswerte der IR-Sensoren
- `POSITION_DELTA[X,Y,PHI]`: Delta Position und Drehung des Roboters

Sensor Output

-

Controller Input

-

Controller Output

-

Blackboard

- `environment_map` Karte der Umgebung mit allen bekannten Objekten im Spielfeld
- `camera_map` Von der Kamera erkannte Objekte

1.3.4 MasterControl

In diesem Modul wird das eigentliche Verhalten des Gabelstaplers realisiert. Das beinhaltet vor allem die Strategien zum Einsammeln, Stapeln sowie das Abladen der Lastobjekte.

Zu Beginn des Spiels ermittelt die Gesamtsteuerung welche Zielplattform die gegnerische bzw. die eigene ist.

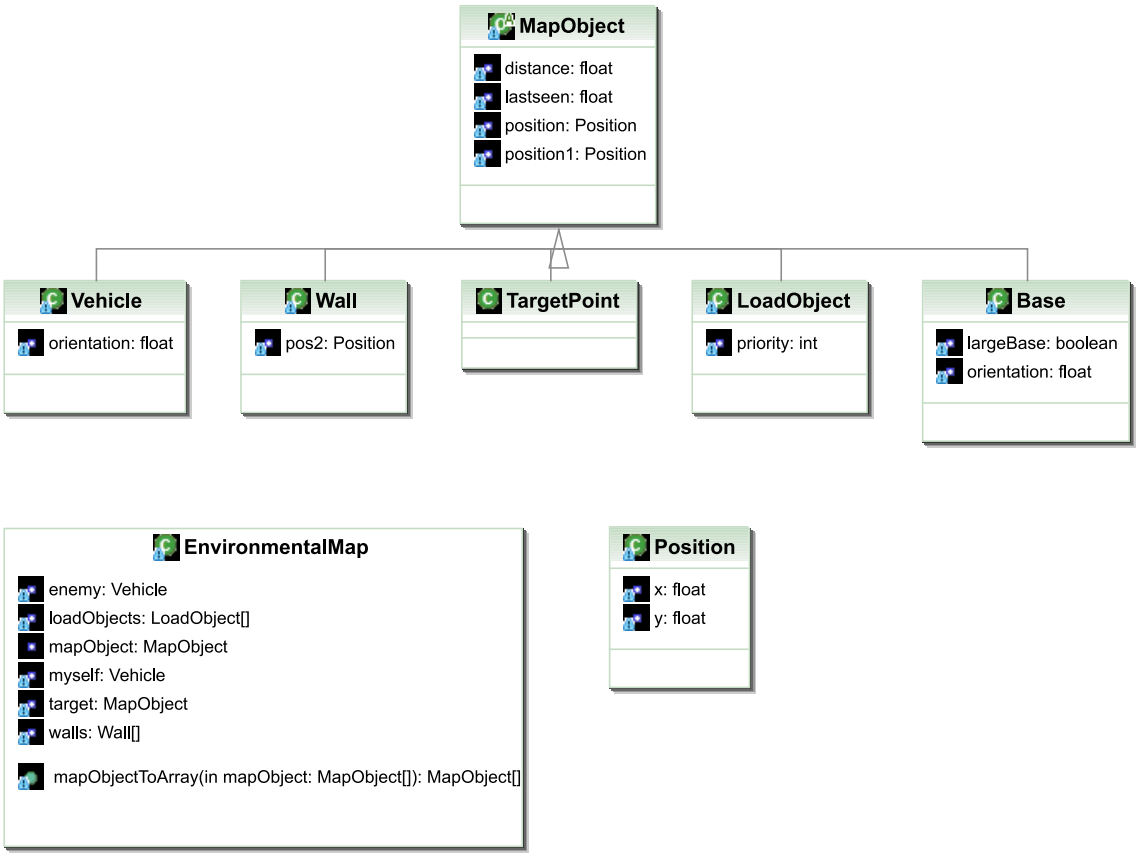

Abbildung 1.6: Klassendiagramm des Modules Mapping

1.3.4.1 Module

ChooseBehaviour

Entscheidet, was als nächstes zu tun ist.

Behave

Führt das aktuelle Verhalten durch (gotoXY, collectLoadObject etc.).

ComputeMap

Analysiert die Umgebung und bewertet u.a. die Lastobjekte.

1.3.4.2 Schnittstellen

Sensor Input

- `button_value[0..3]`: Kontakt eines Bumpers
- `action_done`: Fahrmanöver erfolgreich beendet
- `action_impossible`: Fahrmanöver nicht möglich (z.B. wenn kein passender Weg gefunden wird).

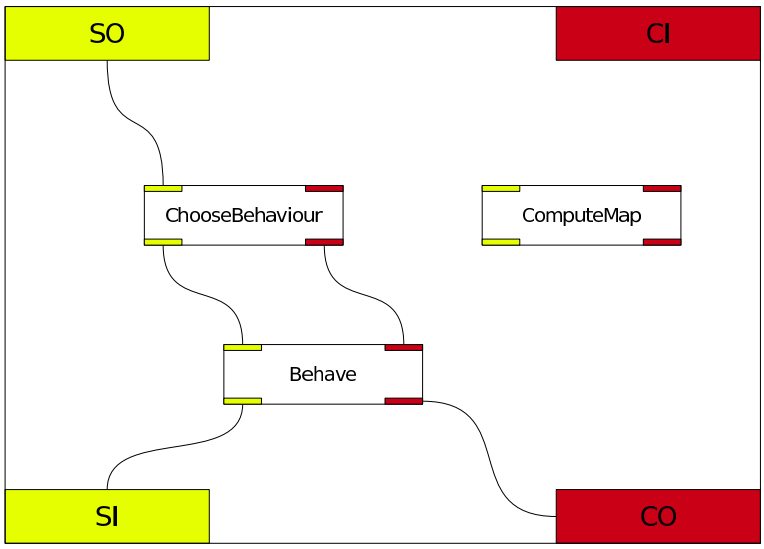

Abbildung 1.7: Aufbau des Modules MasterControl in MCA

Sensor Output

`current_behaviour`: Aktuelles Verhalten (`enum`) des Fahrzeugs zur Anzeige in der GUI.

Controller Input

-

Controller Output

- `drive_mode` bestimmt das aktuelle Fahrverhalten des Roboters.
  Mögliche Werte der Kante `drive_mode`:
  - `gProfi::DM_STOP` Stop
  - `gProfi::DM_GOTO_OBJ` Fahre das Objekt in der Kante `target_object` (Index auf Blackboard Inhalt) an, halte 5 cm davor
  - `gProfi::DM_GOTO_XY` Fahre die Koordinaten in den Kanten `target_posx` und `target_posy` an
  - `gProfi::DM_CUSTOM_DRIVE` Fahre wie in den Kanten `velocity` $[\frac{mm}{s}]$ und `angular_velocity` $[\frac{1}{s}]$ angegeben
  - `gProfi::DM_ROTATE` Richte das Fahrzeug auf einen absoluten Winkel aus
- `target_object`
- `target_posx`
- `target_poxy`
- `velocity`
- `angular_velocity`

Blackboard

- `environment_map`: Karte der Umgebung mit allen bekannten Objekten im Spielfeld

1.3.5 Sensors

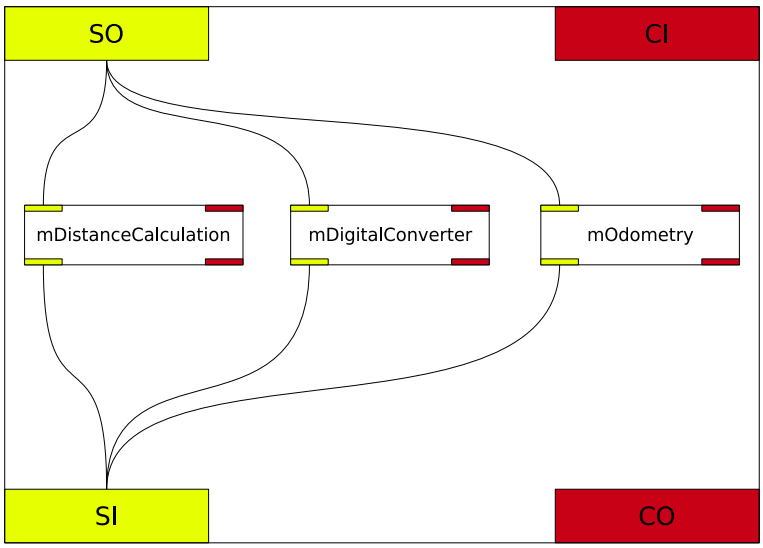

Abbildung 1.8: Aufbau des Modules Sensors in MCA

Die Komponente **Sensors** liest die aktuellen Werte der Infrarotsensoren vom DSP-Board und berechnet daraus, basierend auf der zugehörigen Kennlinie, metrische Abstände. Zudem wird das Auslösen der Taster registriert und weitergeleitet.
Das Sensorikmodul stellt weiterhin Odometriedaten bereit, die aus den Encoderticks der DC-Motoren ausgewertet werden.

1.3.5.1 Schnittstellen

Sensor Input

`ANALOG[0..5]`: Analogwerte der IR-Sensoren `DIGITAL`: Digitalwerte der Taster `ACT[0-1]_ENCODER`: Encoderwerte der Motoren `ACT[0-1]_SPEED`: Geschwindigkeit der Motoren

Sensor Output

`SO_IR_VALUE[0-5]`: Umgerechnete Analogwerte der IR-Sensoren `SO_BUTTON_VALUE[0-3]`: Dekodierte Digitalwerte der Taster `SO_POSITION_DELTA_[X,Y,PHI]`: Position des Roboters

Controller Input

-

Controller Output

-

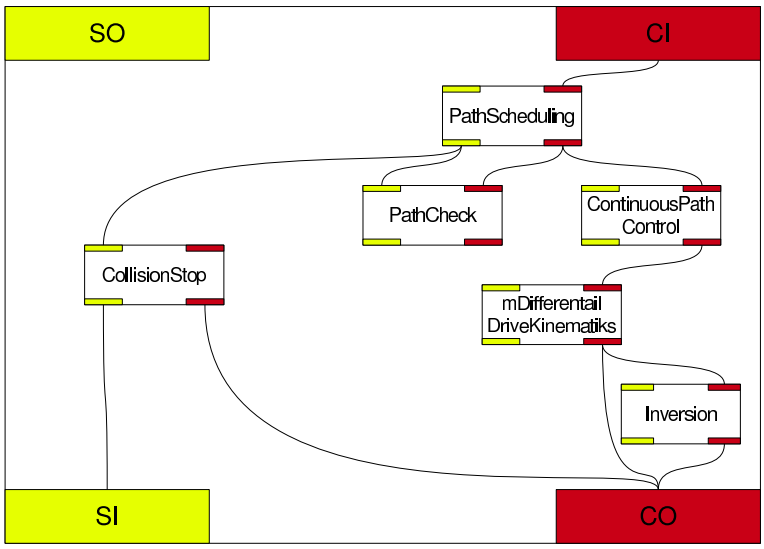

Abbildung 1.9: Aufbau des Modules VehicleControlSystem in MCA

Blackboard

-

1.3.6 VehicleControlSystem

Das VehicleControlSystem ist verantwortlich für die Planung und das Abfahren von den zu fahrenden Strecken. Weiterhin berechnet sie die benötigte Zeit zum Anfahren aller bekannter Lastobjekte. Dies wird benötigt für die Planung der weiteren Aktionen des Roboters. Der Roboter kann während des Fahrbetriebs über die Kollisionsabfrage sofort gestoppt werden, bzw kann reagieren, falls eine Kollision unmittelbar bevorsteht.

1.3.6.1 Module

PathScheduling

Die Steuerung berechnet zu einem gegebenen Zielpunkt und einer dazugehörigen Orientierung eine möglichst gute Bahn und gibt diese an die ContinuousPathControl und PathCheck weiter. Ausserdem berechnet sie die voraussichtlich benötigte Zeit die sie zu allen bekannten Lastobjekten braucht.

ContinuousPathControl

Die ContinuousPathControl ist dafür verantwortlich, dass die geplante Bahn abgefahren wird.

PathCheck

Dieses Modul überprüft ob die geplante Bahn mit möglicherweise neu erkannten Hindernissen kollidiert. Ist dies der Fall signalisiert sie dies der PathScheduling, welche eine neue Bahn berechnet.

CollisionStop

Das CollisionStop-Modul stoppt den Roboter falls eine Kollision unmittelbar bevorsteht.

1.3.6.2 Schnittstellen

Sensor Input

-

Sensor Output

- `Action_Done`: Signalisiert dass das Ziel erreicht ist
- `Action_Impossible`: Signalisiert das die Zielposition unerreichbar ist

Controller Input

- `Drive_Mode`: Bestimmt das aktuelle Fahrverhalten des Roboters (siehe 1.3.4.2)
- `Target_Object`
- `Target_PosX`
- `Target_PosY`
- `Velocity`
- `Angular_Velocity`
- `Emergency_Stop`: Stopt den Roboter sofort

Controller Output

- `Velocity_Left`: Ausgabe an DSP-Remote Part
- `Velocity_Right`: Ausgabe an DSP-Remote Part

1.4 Aufbau des Roboters

1.4.1 Kamera

Die Kamera ist verantwortlich für die Erkennung der Hindernisse und für die Abstandsberechnung. Für eine möglichst gute Übersicht über das Spielfeld zu erhalten, ist die Kamera so angebracht, dass sie nach oben auf einen ellipsoidförmigen Spiegel gerichtet ist. Dadurch erhält man eine Rundumsicht um den Roboter.

1.4.2 IR-Sensor

Die Hauptaufgabe der IR-Sensoren besteht darin die Abstände zu den Hindernissen zu bestimmen. Zu diesem Zweck sind die Sensoren wie auf Abbildung 1.10 zu sehen angebracht.

1.4.3 Taster

Taster 1 und Taster 2 sind zur Erkennung und Zuordnung der Zielplattformen. Die Aufgabe des dritten Tasters ist die Erkennung der Aufnahme und des Absetzens der Lastobjekte. Der vierte Taster signalisiert ob der Roboter nah genug an ein Lastobjekt heran gefahren ist um dieses aufzunehmen.

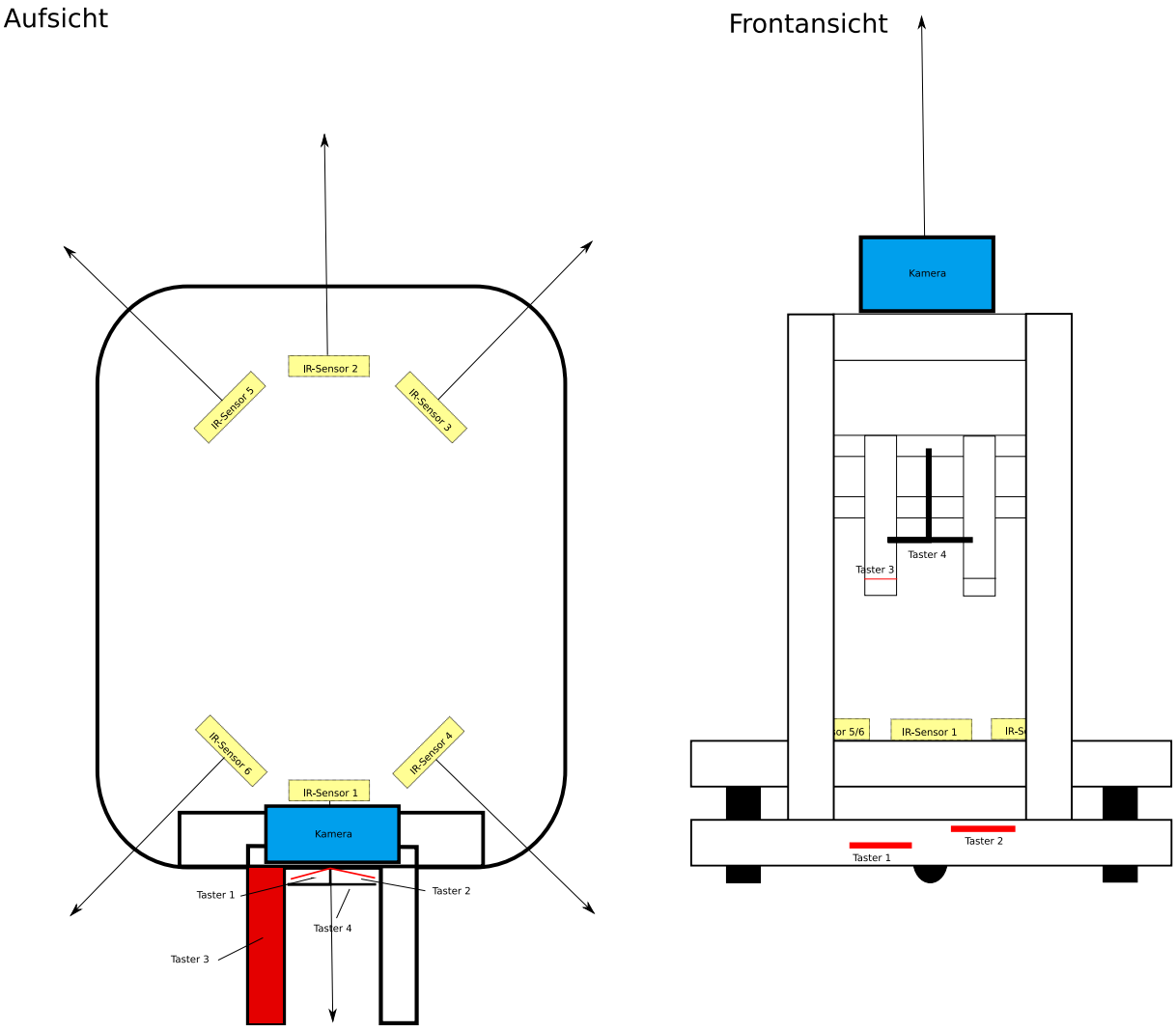

Abbildung 1.10: Aufbau des Roboters