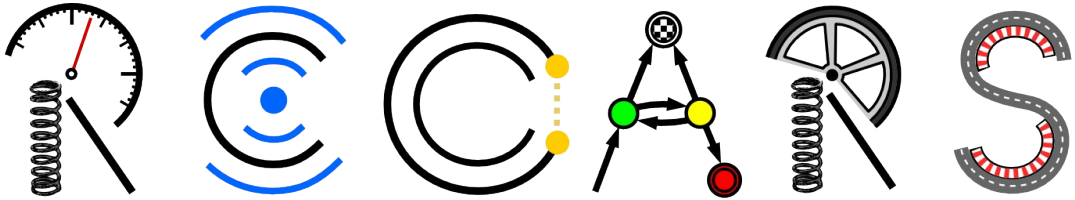


Realtime Controlled Cooperative Autonomous Racing System

Nikolai Bräuer
Michael Bukowski
Anatolij Fandrich
Tom Reske

Zweites Review

Carl von Ossietzky Universität Oldenburg
Department für Informatik

Agenda

- Vorstellung
- Einleitung
- Projektorganisation
- Systemarchitektur und Schnittstellen
- Sicherheitskonzept
- Hardware-Evaluation
- Software-Entwicklung
- Testen
- Angepasste Anforderungen
- Aktueller Stand
- Ausblick
- Live-Demo

Agenda

- Vorstellung
- Einleitung
- Projektorganisation
- Systemarchitektur und Schnittstellen
- Sicherheitskonzept
- Hardware-Evaluation
- Software-Entwicklung
- Testen
- Angepasste Anforderungen
- Aktueller Stand
- Ausblick
- Live-Demo

Realtime Controlled Cooperative Autonomous Racing System

Vorstellung

Vorstellung

Name: Nikolai Bräuer (B.Sc.)

Position: Testmanagement
Dokumentenmanagement

Studiengang: Informatik (M.Sc.)

Name: Michael Bukowski (B.Sc.)

Position: Hard- und Softwarebeauftragter
Öffentlichkeitsarbeit

Studiengang: Informatik (M.Sc.)

Vorstellung

Name: Anatolij Fandrich (B.Sc.)

Position: Qualitätsmanagement

Studiengang: Eingebettete Systeme und Mikrorobotik (M.Sc.)

Name: Tom Reske (B.Eng.)

Position: Projektgruppenleiter

Studiengang: Eingebettete Systeme und Mikrorobotik (M.Sc.)

Agenda

- Vorstellung
- Einleitung
- Projektorganisation
- Systemarchitektur und Schnittstellen
- Sicherheitskonzept
- Hardware-Evaluation
- Software-Entwicklung
- Testen
- Angepasste Anforderungen
- Aktueller Stand
- Ausblick
- Live-Demo

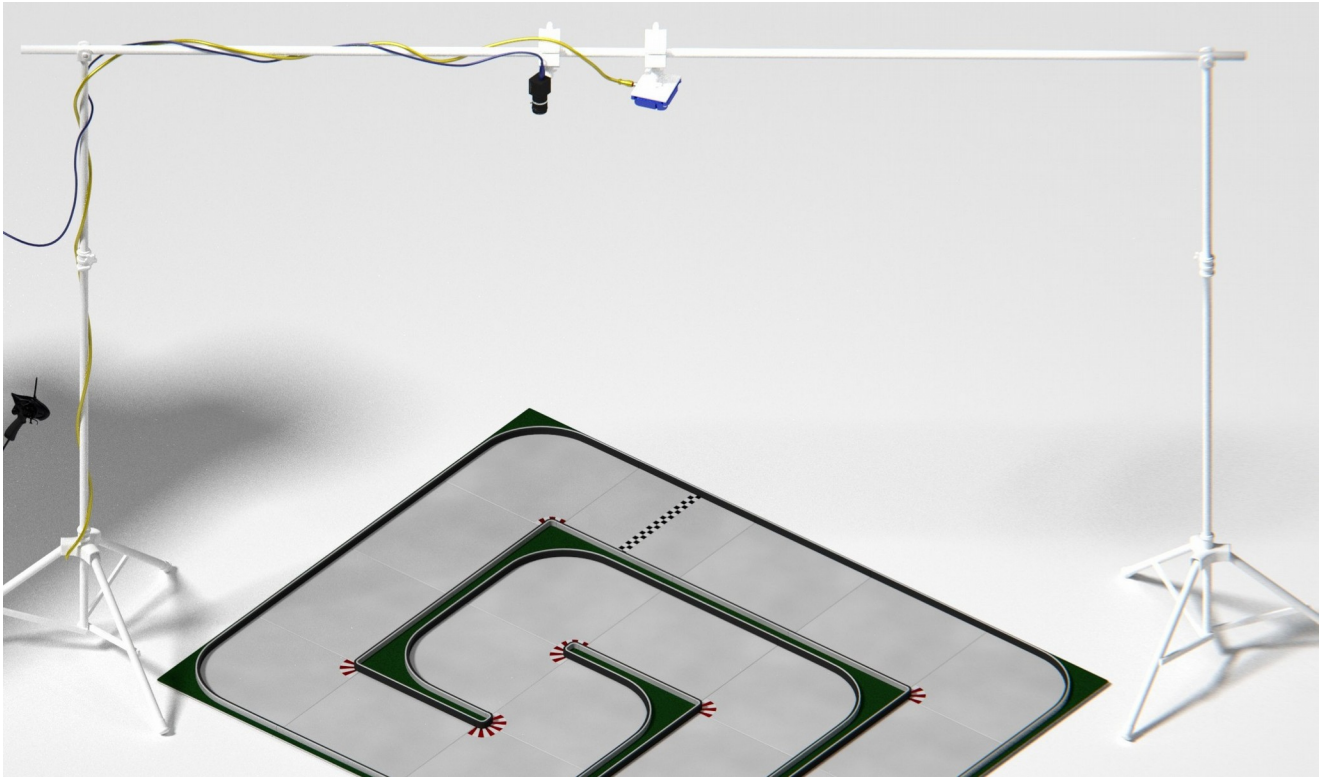
Einleitung
Was ist *"autonomous racing"*?

- **autonomous**
  - selbstständig, unabhängig, eigenständig, selbst-bestimmend, sich selbst regierend

Video: https://www.youtube.com/watch?v=RDNJcFi6pWs

Realtime Controlled Cooperative Autonomous Racing System

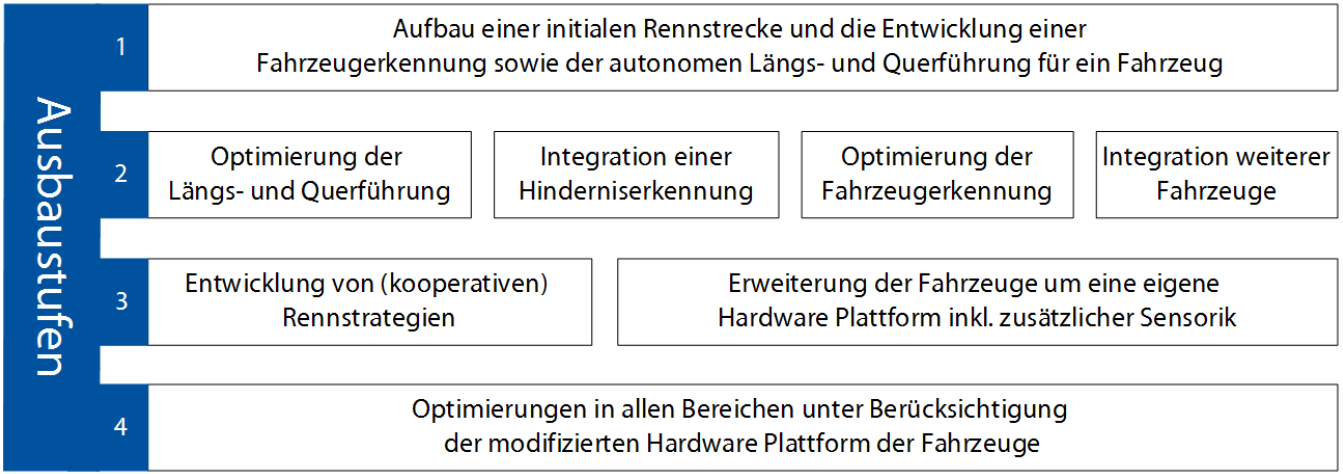
Einleitung
Was ist *"autonomous racing"*?



Quelle: CARS – Camera-based Autonomous Racing System, Abbildung 1

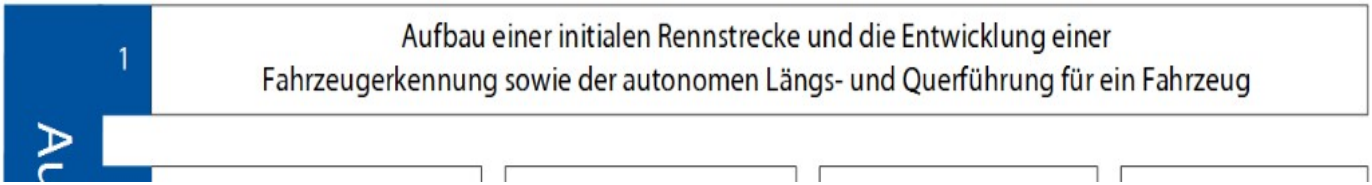
Einleitung
Long Term Vision

- Stetige **Weiterentwicklung** durch studentische Projektgruppen sowie Abschlussarbeiten ermöglichen
- Auf Grundlage der **State of the Art** und den **Erfahrungen** der beteiligten Universitäten / Forschungseinrichtungen ergibt sich **ein möglicher, mehrstufiger Prozess**



Einleitung
Aufgabenstellung der Projektgruppe

- Realisierung erster Ausbaustufe der Long Term Vision

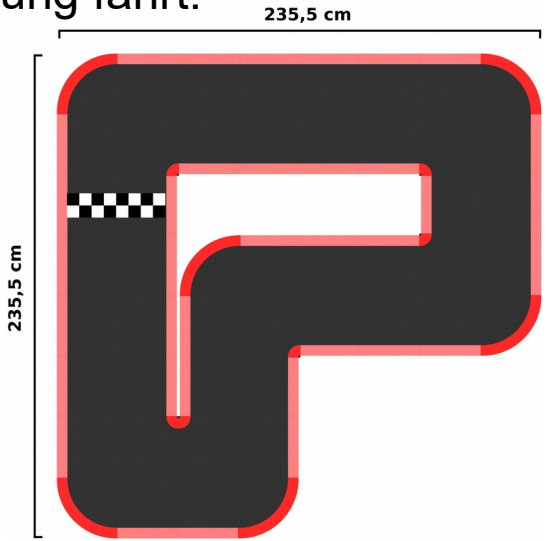


  - Aufbau einer **initialen Rennstrecke**
  - Entwicklung einer **Fahrzeugerkennung** und **autonomen Längs- und Querführung** eines Fahrzeuges
  - **Erweiterbarkeit** ermöglichen
  - Nutzung von **Kyosho dNaNo Fahrzeugen** und des Werkzeuges **SCADE**

Realtime Controlled Cooperative Autonomous Racing System

Einleitung
Szenario: Unfallfreie Fahrt

Das Szenario unfallfreie Fahrt sieht vor, dass **ein einzelnes Fahrzeug** auf einer **geschlossenen Rennstrecke autonom** mit einer **Mindestdurchschnittsgeschwindigkeit von 1,5 m/s** mindestens **fünf Runden ohne eine Kollision** mit der Fahrbahnbegrenzung fährt.

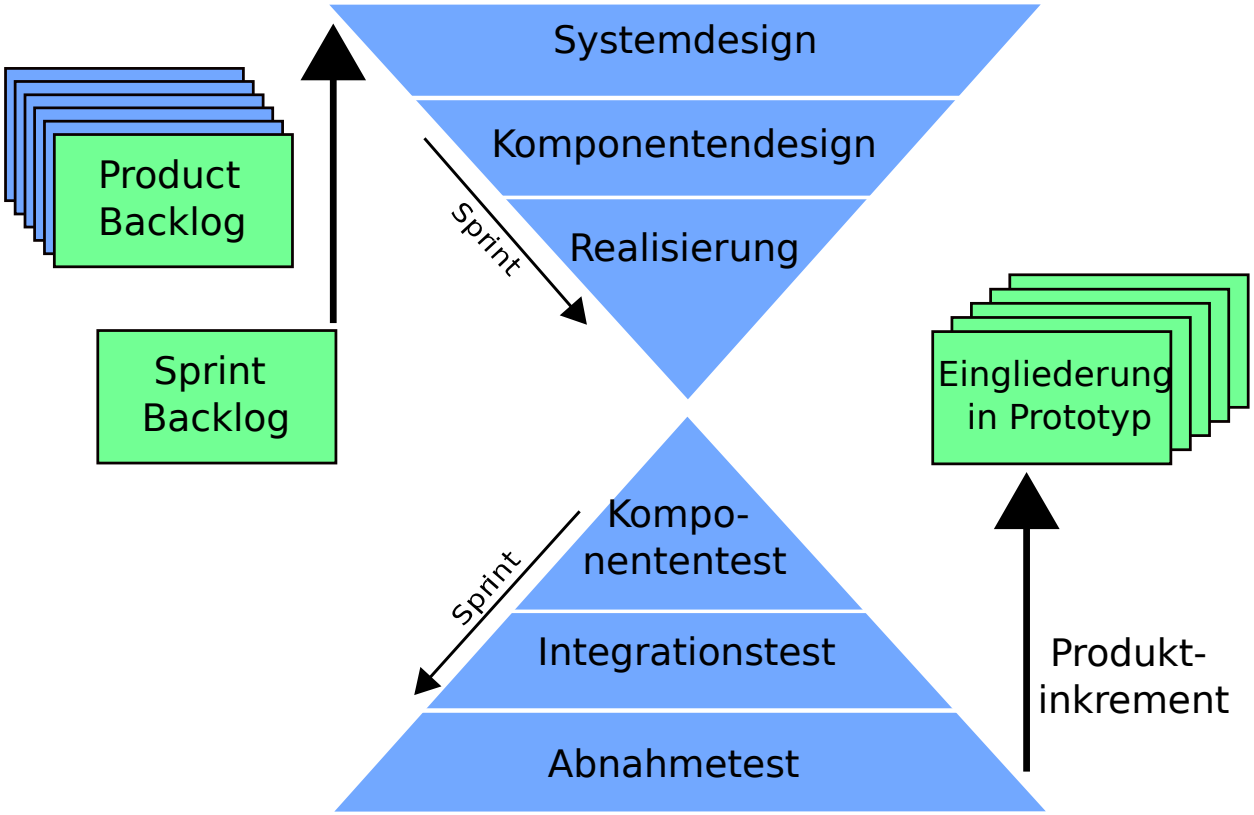


Realtime Controlled Cooperative Autonomous Racing System

Agenda

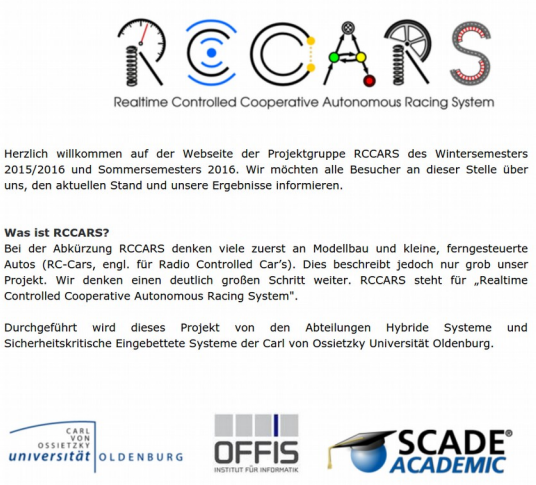
- Vorstellung
- Einleitung
- Projektorganisation
- Systemarchitektur und Schnittstellen
- Sicherheitskonzept
- Hardware-Evaluation
- Software-Entwicklung
- Testen
- Angepasste Anforderungen
- Aktueller Stand
- Ausblick
- Live-Demo

Projektorganisation
Vorgehensmodell ScrVm



Projektorganisation

- Wöchentliche ScrVm Treffen
  - Moderation
  - Protokolle
- Monatliche Tätigkeitsberichte
- Weitere flexible Treffen
- Öffentlichkeitsarbeit
  - Blog
  - Wöchentliche Aktualisierungen
  - Veranstaltungen



Projektorganisation
Re-Akkreditierung & Schülerinformationstag

- Schülerinformationstag (Januar 2016)
  - Lichtschranke
  - Schüler 5.000 Runden
  - Hardware-Evaluation
- Re-Akkreditierung der Informatikstudiengänge (April 2016)
  - Vorstellung Exponat und Projektfortschritt
  - Positionsbestimmung auf Fernseher
  - Reges Interesse der Zuschauer
  - Positive Resonanz

RCCARS
Realtime Controlled Cooperative Autonomous Racing System

Agenda

- Vorstellung
- Einleitung
- Projektorganisation
- Systemarchitektur und Schnittstellen
- Sicherheitskonzept
- Hardware-Evaluation
- Software-Entwicklung
- Testen
- Angepasste Anforderungen
- Aktueller Stand
- Ausblick
- Live-Demo

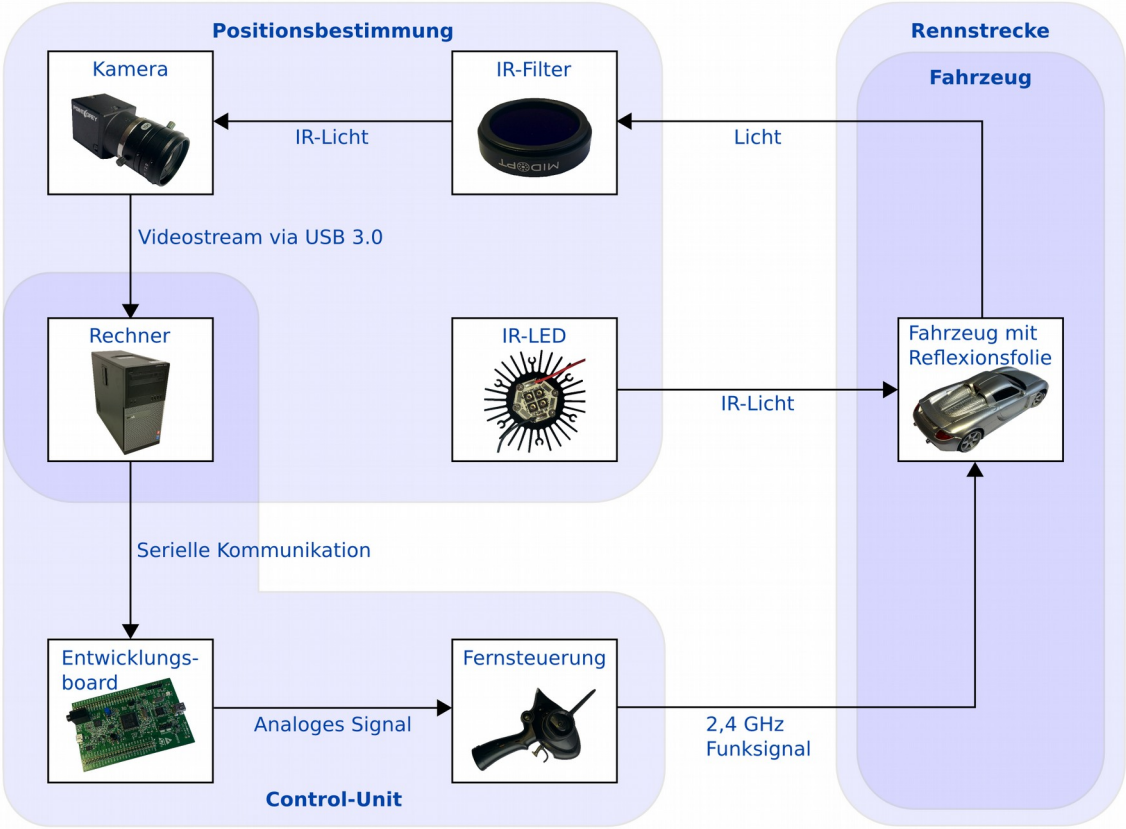
Systemarchitektur und Schnittstellen

- Systemarchitektur (kurze Wiederholung)
- Schnittstellen
  - Subsystem Fahrzeug
  - Subsystem Positionsbestimmung
  - Subsystem Control-Unit
  - Subsystem Rennstrecke
  - Systemsteuerungssoftware
  - Kommunikationsschnittstelle

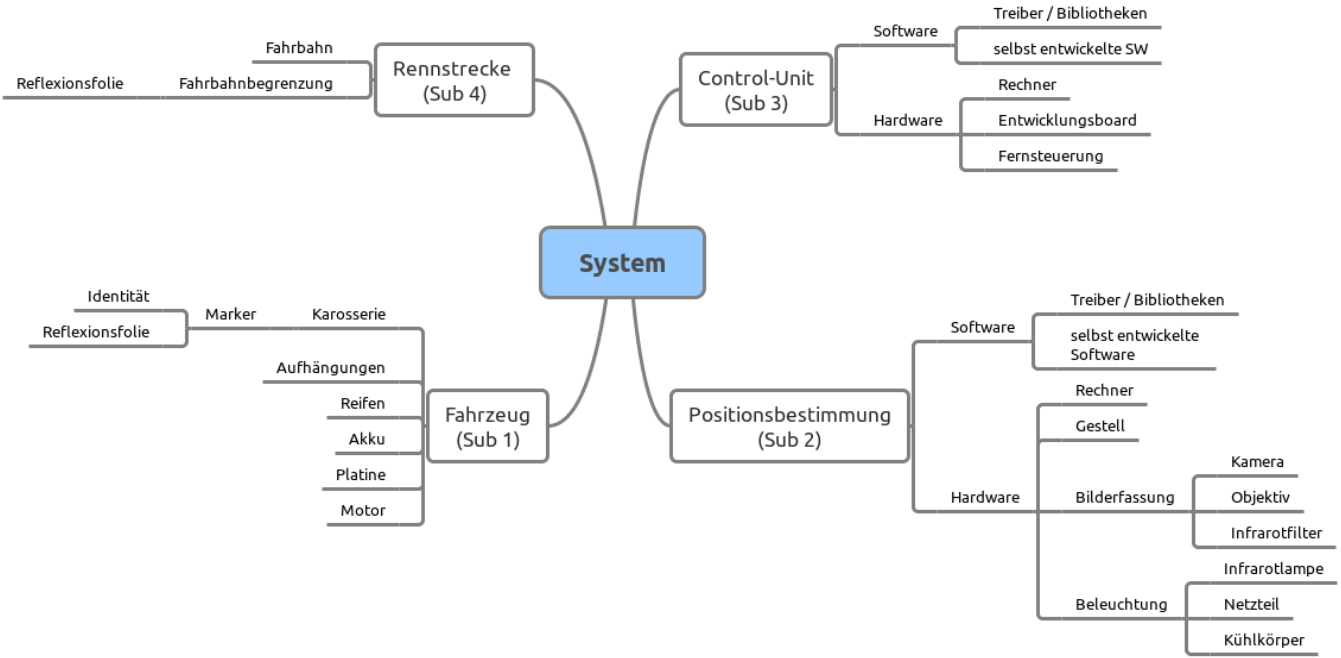
Systemarchitektur und Schnittstellen

- Systemarchitektur (kurze Wiederholung)
- Schnittstellen
  - Subsystem Fahrzeug
  - Subsystem Positionsbestimmung
  - Subsystem Control-Unit
  - Subsystem Rennstrecke
  - Systemsteuerungssoftware
  - Kommunikationsschnittstelle

Systemarchitektur
Signalfluss

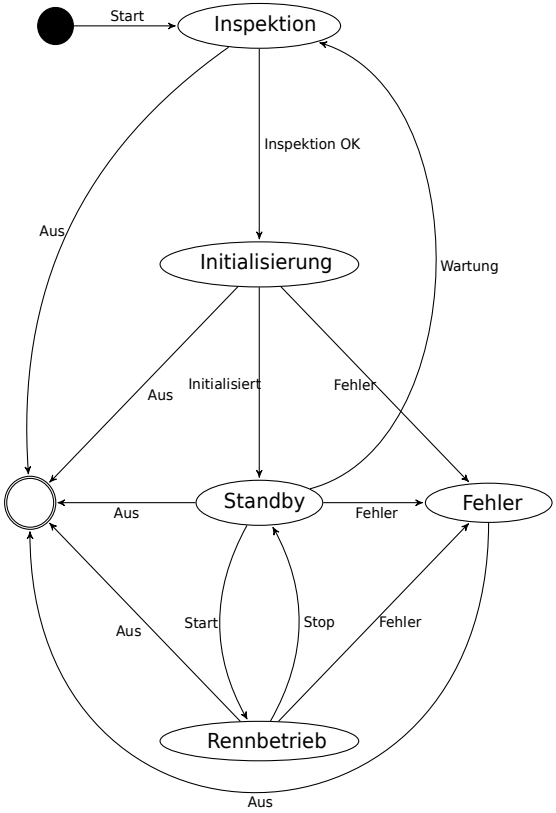


Systemarchitektur
Subsysteme



Systemarchitektur
Systemsteuerungssoftware

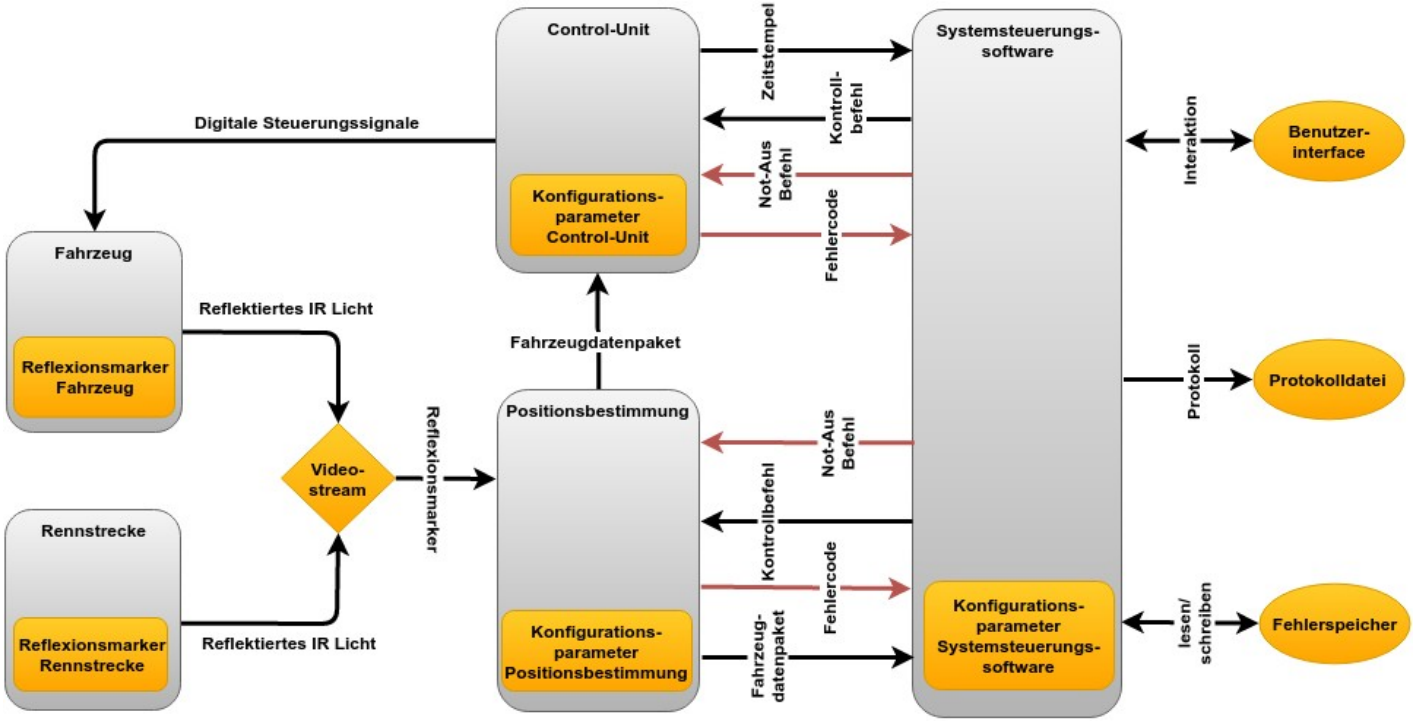
- Überwachung und Bedienung des Systems
  - durch den Administrator
- Benutzerinterface
  - Informationen:
    - Systemzustand
    - Fehlercode
  - Buttons:
    - Inspektion Bestätigen
    - Rennbetrieb starten/stoppen
    - Wartung einleiten
    - System beenden
    - Not-Aus



Systemarchitektur und Schnittstellen

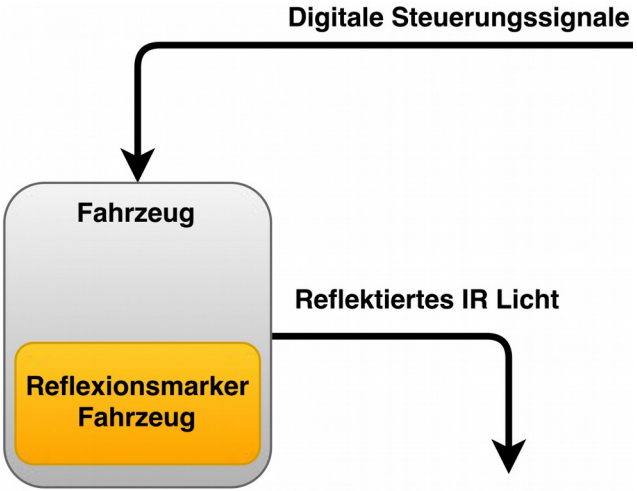
- Systemarchitektur (kurze Wiederholung)
- Schnittstellen
  - Subsystem Fahrzeug
  - Subsystem Positionsbestimmung
  - Subsystem Control-Unit
  - Subsystem Rennstrecke
  - Systemsteuerungssoftware
  - Kommunikationsschnittstelle

Schnittstellen



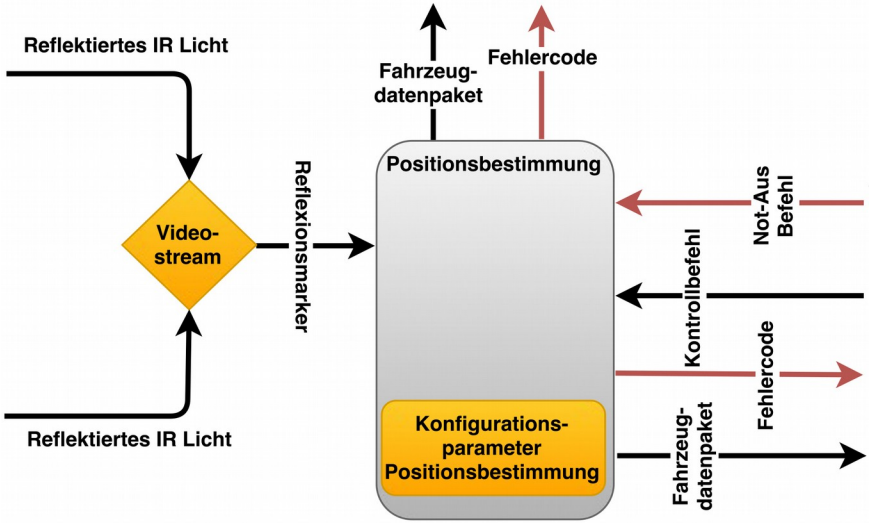
Schnittstellen
Subsystem Fahrzeug

- Eingabeschnittstellen
  - Digitale Steuerungssignale
  - Infrarotlicht
- Ausgabeschnittstellen
  - Reflektiertes Infrarotlicht



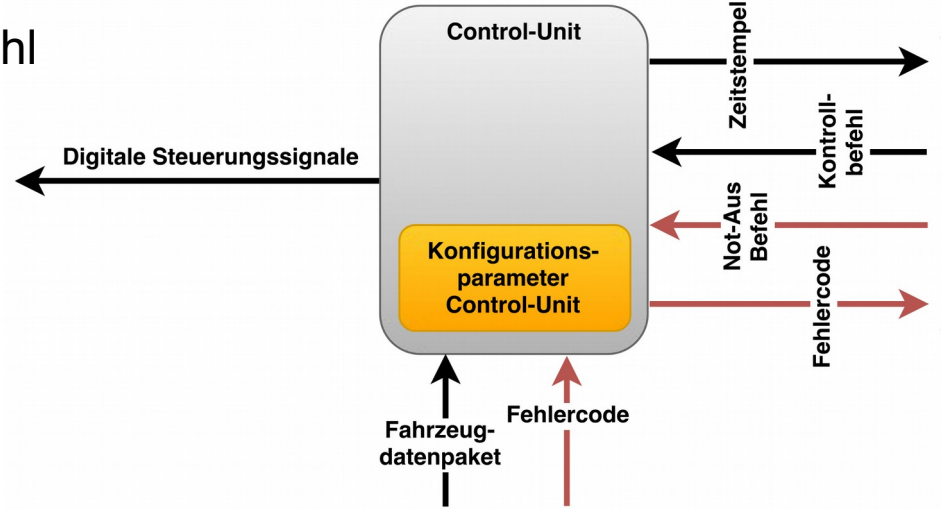
Schnittstellen
Subsystem Positionsbestimmung

- Eingabeschnittstellen
  - Videostream
  - Kontrollbefehl
  - Not-Aus Befehl
- Ausgabeschnittstellen
  - Fahrzeugdatenpaket
  - Fehlercode



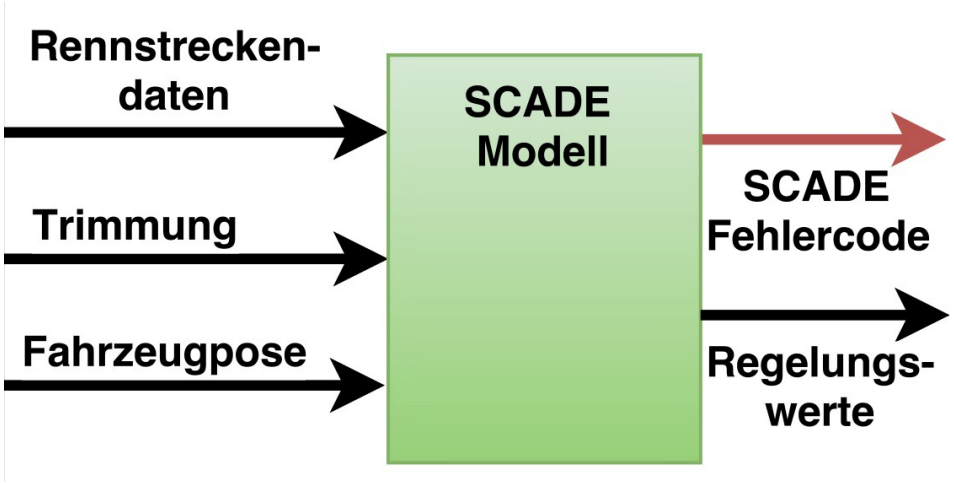
Schnittstellen
Subsystem Control-Unit

- Eingabeschnittstellen
  - Fahrzeugdatenpaket
  - Fehlercode
  - Not-Aus Befehl
  - Kontrollbefehl
- Ausgabeschnittstellen
  - Digitale Steuerungssignale
  - Fehlercode
  - Zeitstempel



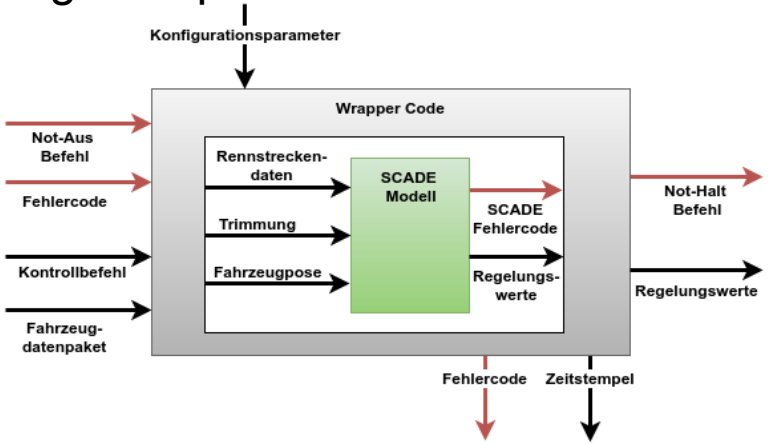
Schnittstellen
Subsystem Control-Unit (SCADE Modell)

- Eingabeschnittstellen
  - Rennstreckendaten
  - Trimmung
  - Fahrzeugpose
- Ausgabeschnittstellen
  - SCADE Fehlercode
  - Regelungswerte



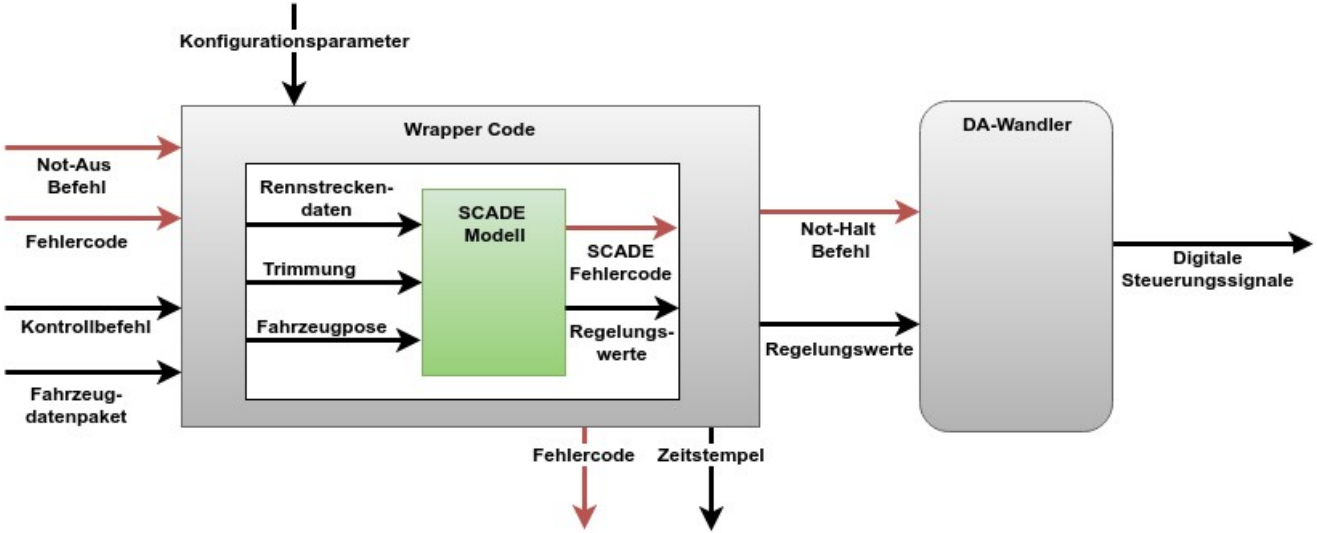
Schnittstellen
Subsystem Control-Unit (Wrapper Code)

- Eingabeschnittstellen
  - Not-Aus-Befehl
  - Fehlercode
  - Kontrollbefehl
  - Fahrzeugdatenpaket
- Ausgabeschnittstellen
  - Not-Halt Befehl
  - Regelungswerte
  - Zeitstempel
  - Fehlercode



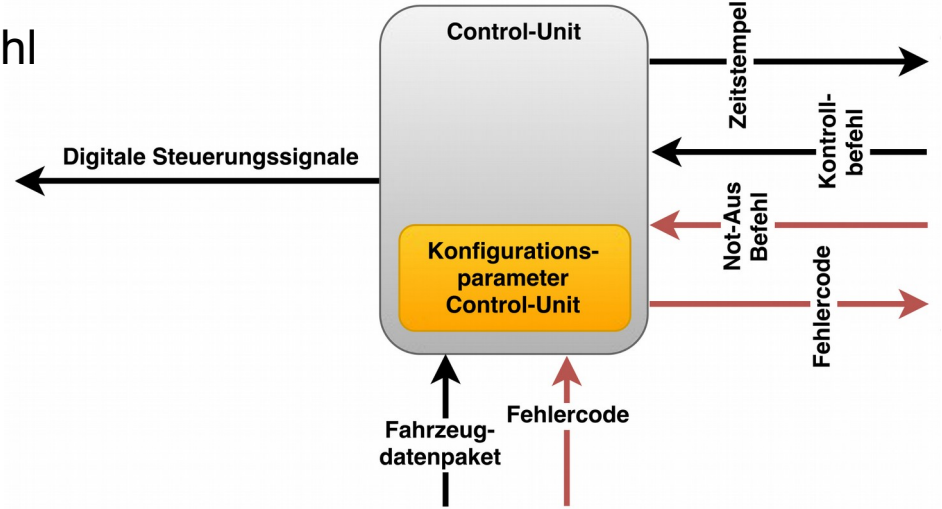
Schnittstellen
DA-Wandlungssoftware

- Eingabeschnittstellen
  - Regelungswerte
  - Hot-Halt Befehl
- Ausgabeschnittstellen
  - Digitale Steuerungssignale



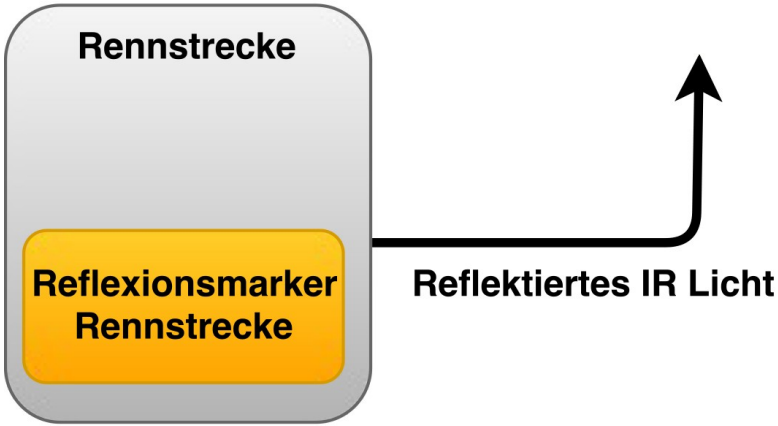
Schnittstellen
Subsystem Control-Unit

- Eingabeschnittstellen
  - Fahrzeugdatenpaket
  - Fehlercode
  - Not-Aus Befehl
  - Kontrollbefehl
- Ausgabeschnittstellen
  - Digitale Steuerungssignale
  - Fehlercode
  - Zeitstempel



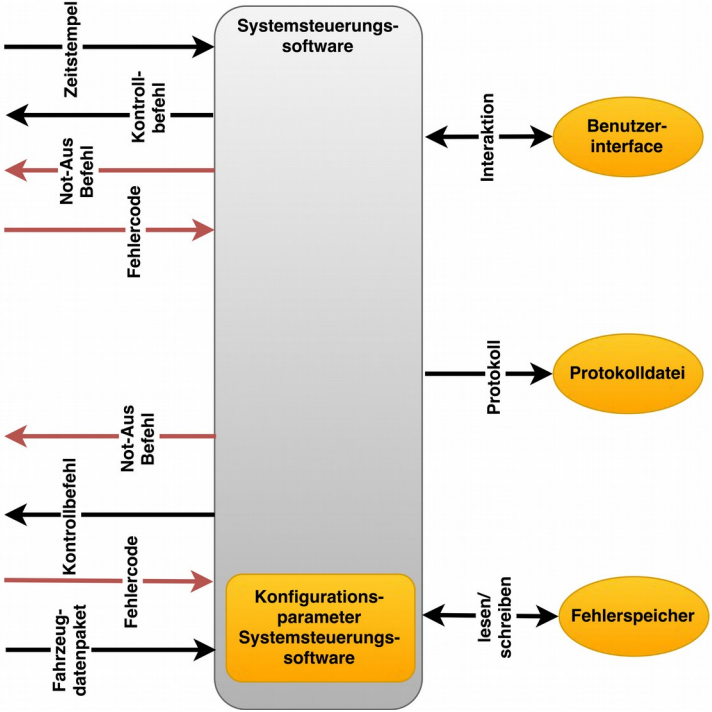
Schnittstellen
Subsystem Rennstrecke

- Eingabeschnittstellen
  - Infrarotlicht
- Ausgabeschnittstellen
  - Reflektiertes Infrarotlicht



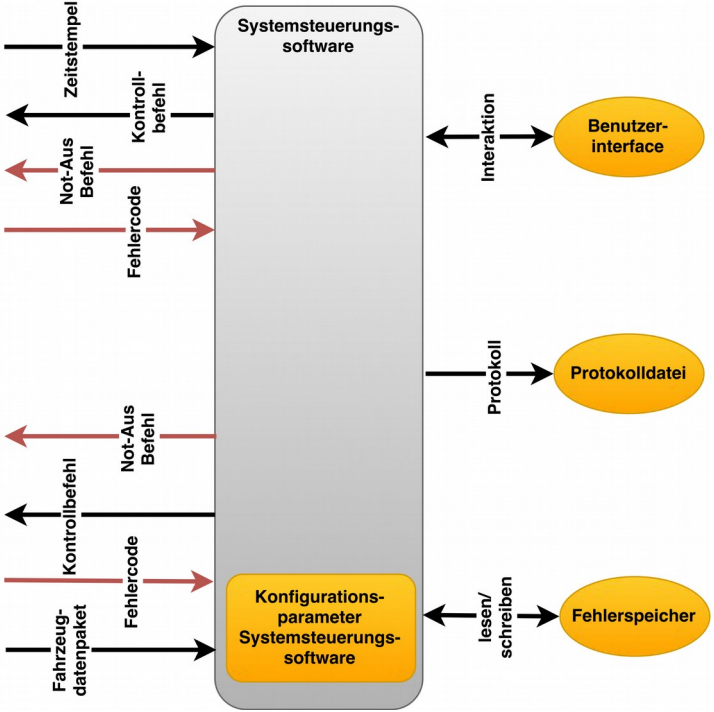
Schnittstellen
Systemsoftware

- Eingabeschnittstellen
  - Benutzerinterface
  - Fehlerspeicher
  - Fahrzeugdatenpaket
  - Fehlercode
  - Zeitstempel



Schnittstellen
Systemsoftware

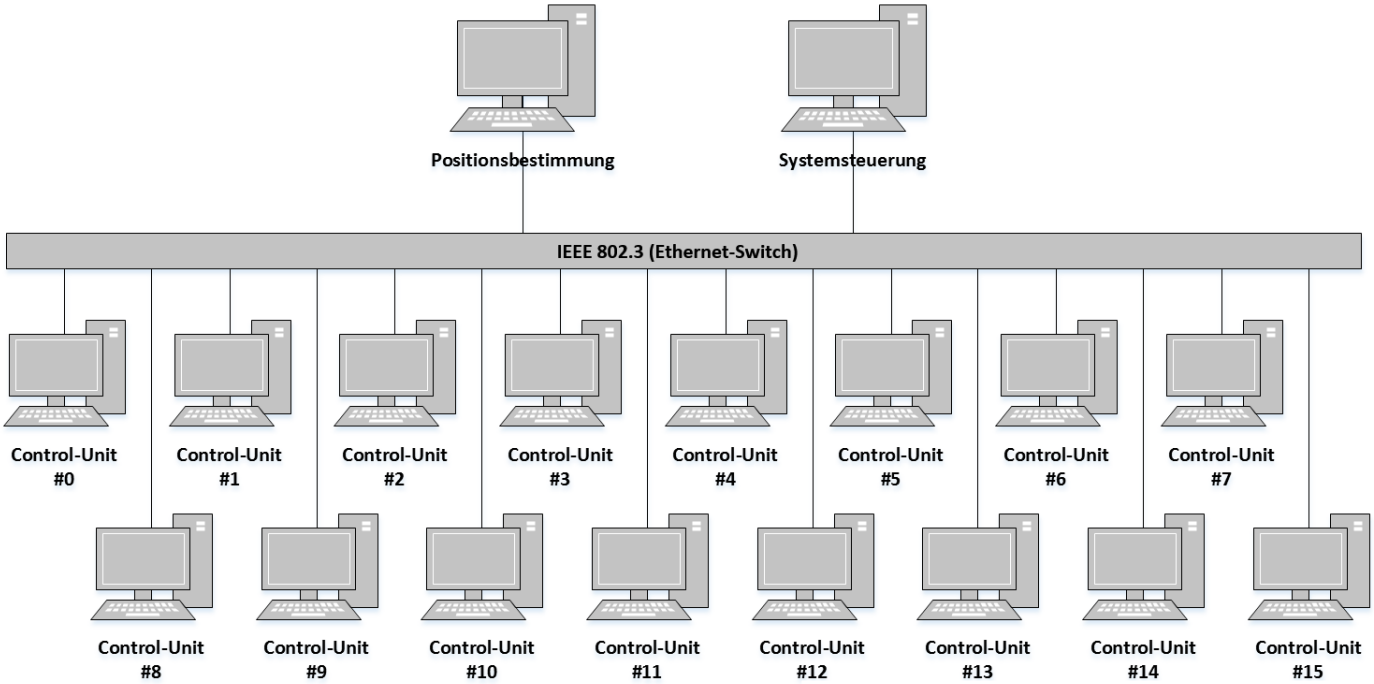
- Ausgabeschnittstellen
  - Benutzerinterface
  - Protokoll
  - Fehlerspeicher
  - Kontrollbefehl
  - Not-Aus Befehl



Schnittstellen
Kommunikationsschnittstelle

- Jede Komponente wird auf eigenem Rechner ausgeführt.
  - Systemsteuerungssoftware
  - Positionsbestimmung
  - Control-Unit's
- Verteiltes System
  - „Ein Zusammenschluss unabhängiger Computer, die sich für den Benutzer als ein einziges System präsentieren.“

Schnittstellen
Kommunikationsschnittstelle



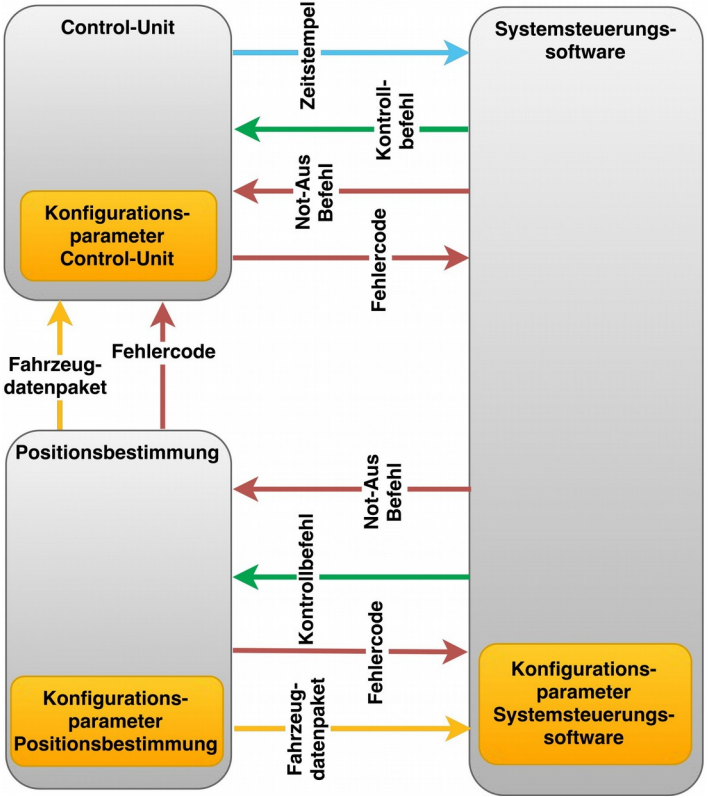
Schnittstellen
Zusammenfassend

Fahrzeugdatenpaket

Kontrolldatenpaket

Fehlerdatenpaket

Zeitstempeldatenpaket



Agenda

- Vorstellung
- Einleitung
- Projektorganisation
- Systemarchitektur und Schnittstellen
- Sicherheitskonzept
- Hardware-Evaluation
- Software-Entwicklung
- Testen
- Angepasste Anforderungen
- Aktueller Stand
- Ausblick
- Live-Demo

Realtime Controlled Cooperative Autonomous Racing System

Sicherheitskonzept

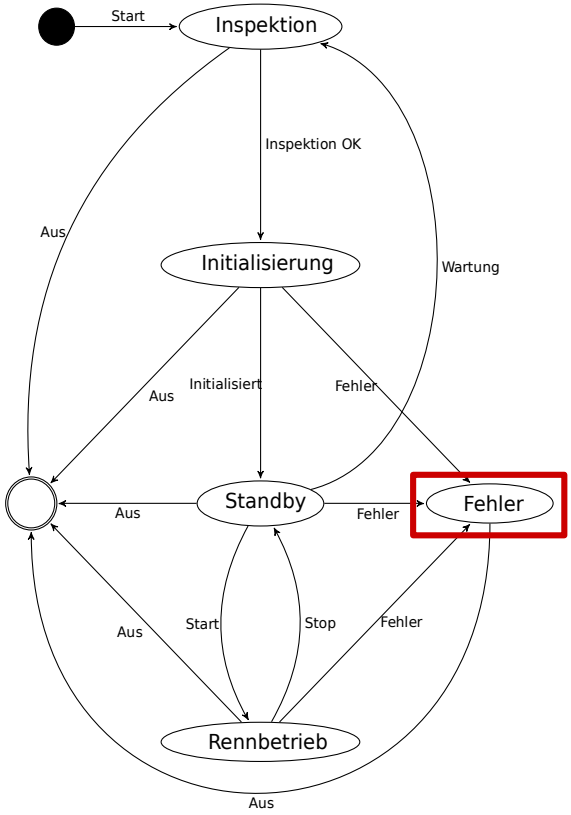
- Fahrzeug als sicherheits-kritisches Objekt im Fokus
- Fehlerfall
  - Komplettausfall
  - Fehlverhalten
- Darstellung möglicher Fehlerszenarien
  - **In wiefern kann in das System eingegriffen werden und das Fahrzeug angehalten werden?!**

Realtime Controlled Cooperative Autonomous Racing System

Sicherheitskonzept
Not-Aus Befehl

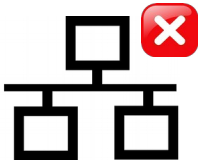
- Von Systemsteuerungssoftware auslösbar
  - System wird in sicheren Fehlerzustand überführt

- Manuell:
  - Betätigung eines Buttons
- Automatisch:
  - Durch Erkennung eines Fehlercodes



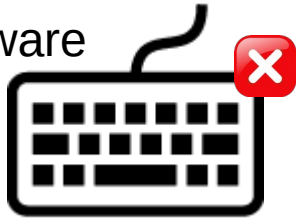
Sicherheitskonzept
Fehlerszenarien

- Fehler in der Kommunikation



- Fehler im Videostream

- Fehler in der Positionsbestimmung
- Fehler in der Systemsteuerungssoftware

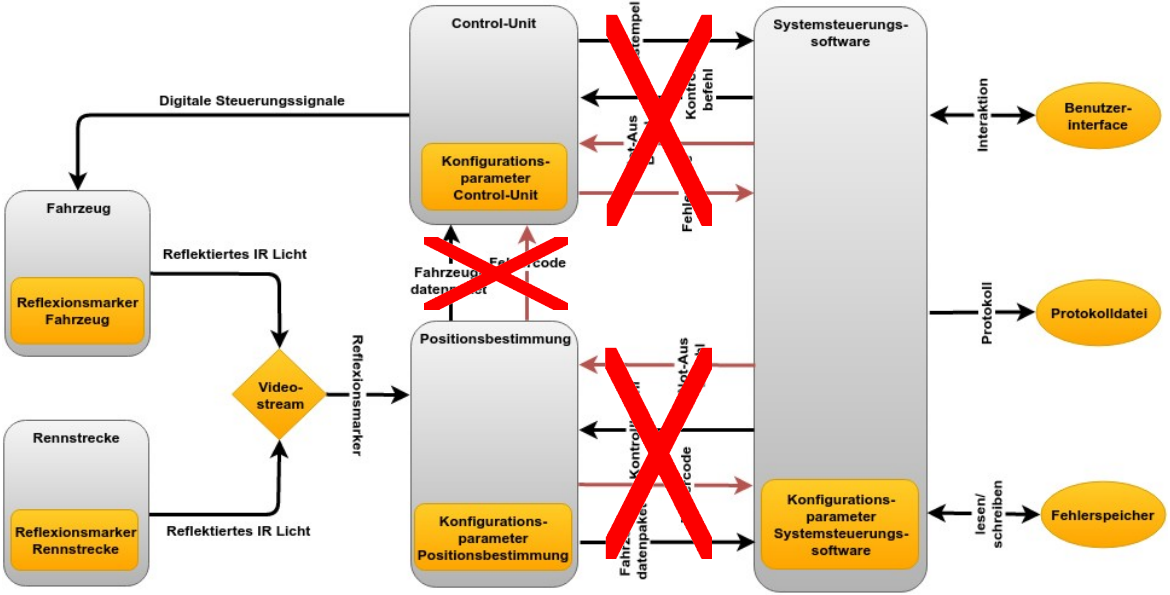


- Fehler in der Control-Unit

RCCARS
Realtime Controlled Cooperative Autonomous Racing System

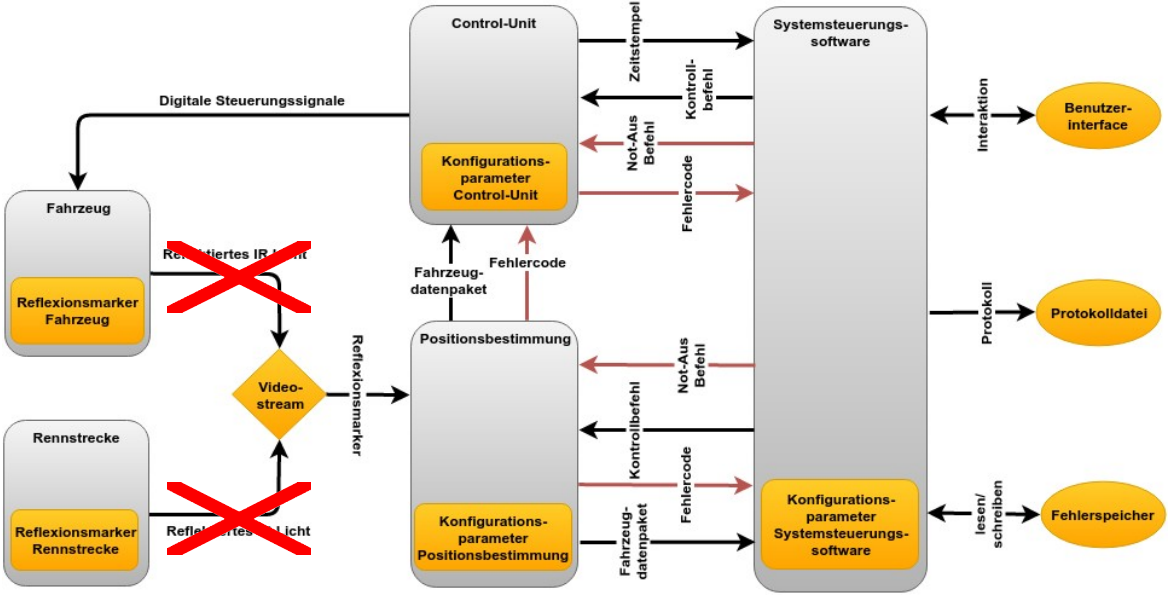
Sicherheitskonzept
Schadenszenario (Kommunikation)

- Ausfall der Kommunikation
- Datenintegrität



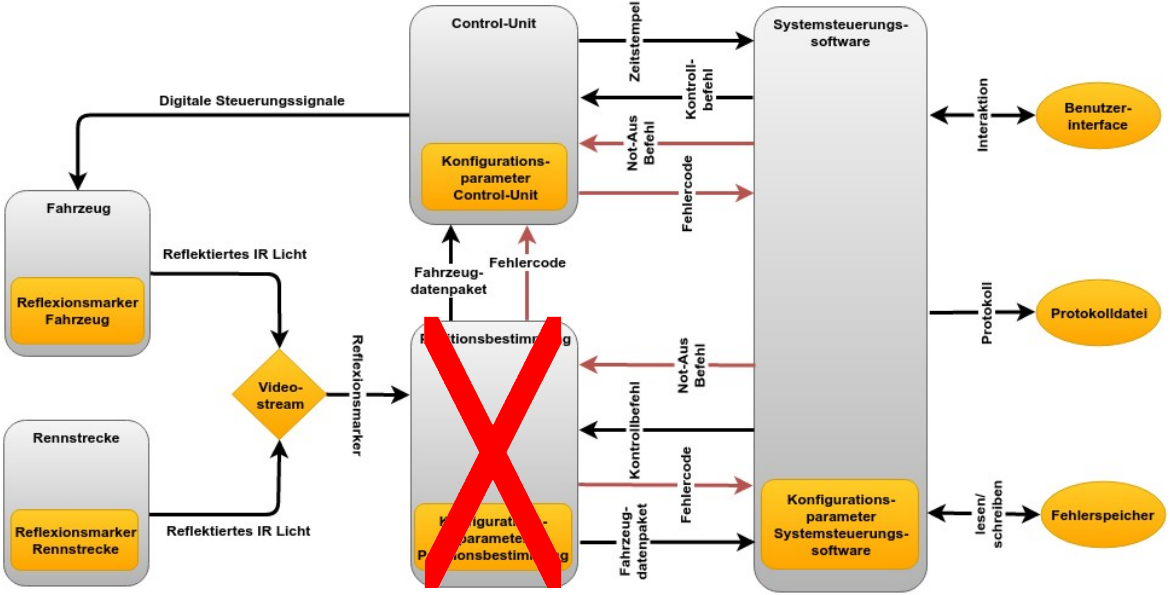
Sicherheitskonzept
Schadenszenario (Videostream)

- Fahrzeug wird nicht mehr erkannt
- Rennstrecke wird nicht mehr erkannt



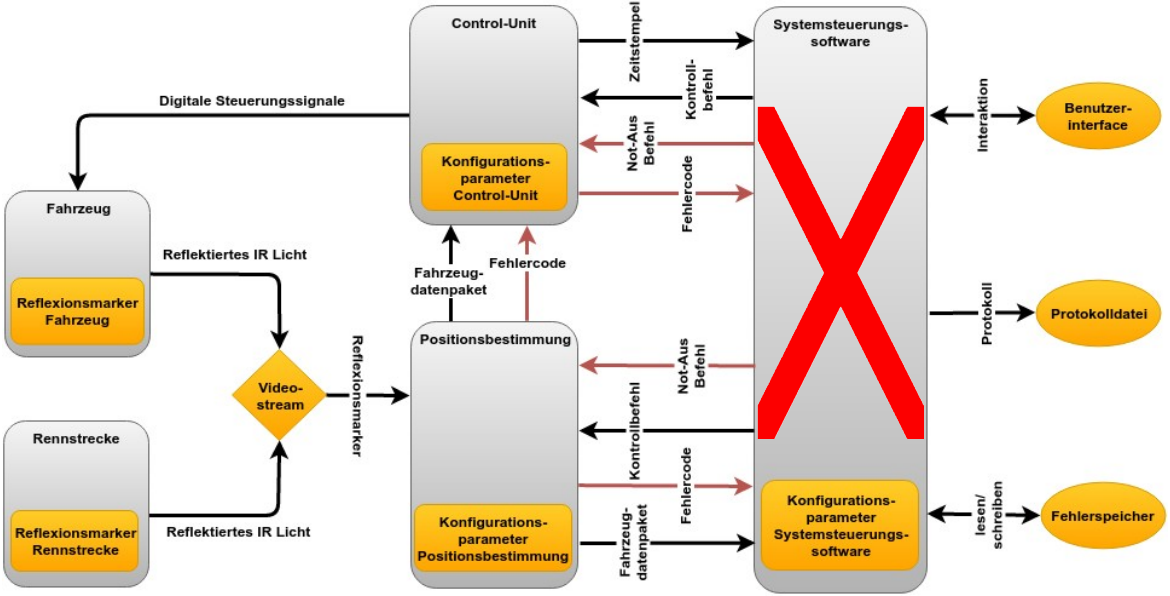
Sicherheitskonzept
Schadenszenario (Positionsbestimmung)

- Fahrzeugdatenpaket wird nicht mehr gesendet
- Fahrzeugdatenpaket ist fehlerhaft



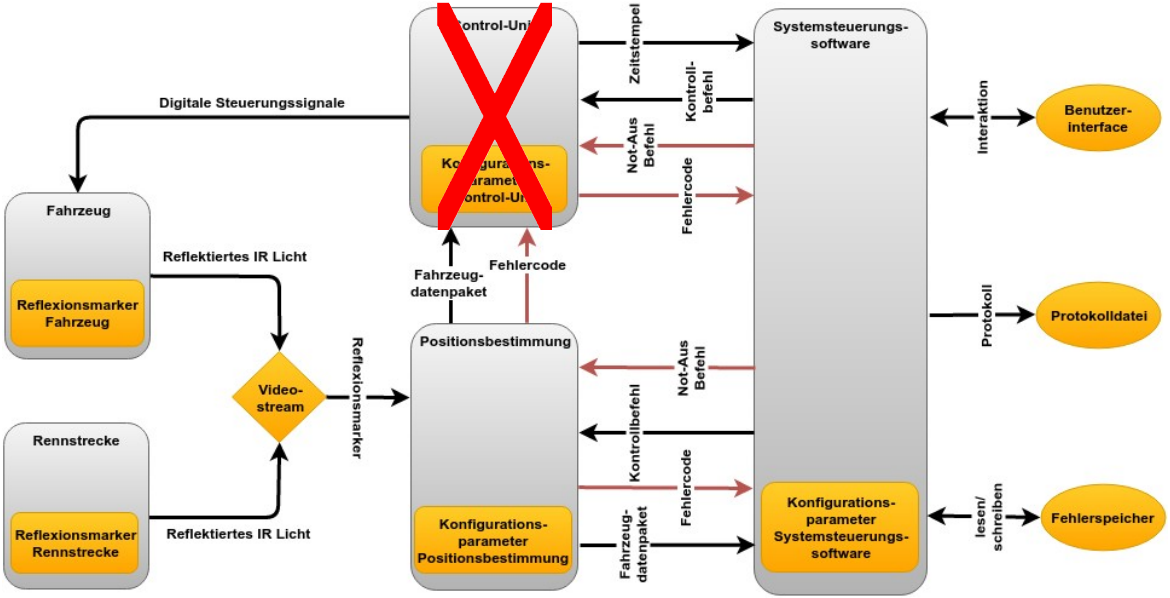
Sicherheitskonzept
Schadenszenario (Systemsteuerungssoftware)

- Systemsteuerungssoftware reagiert nicht mehr
  - Rennbetrieb stoppen nicht mehr möglich
  - Not-Aus Befehl nicht mehr möglich



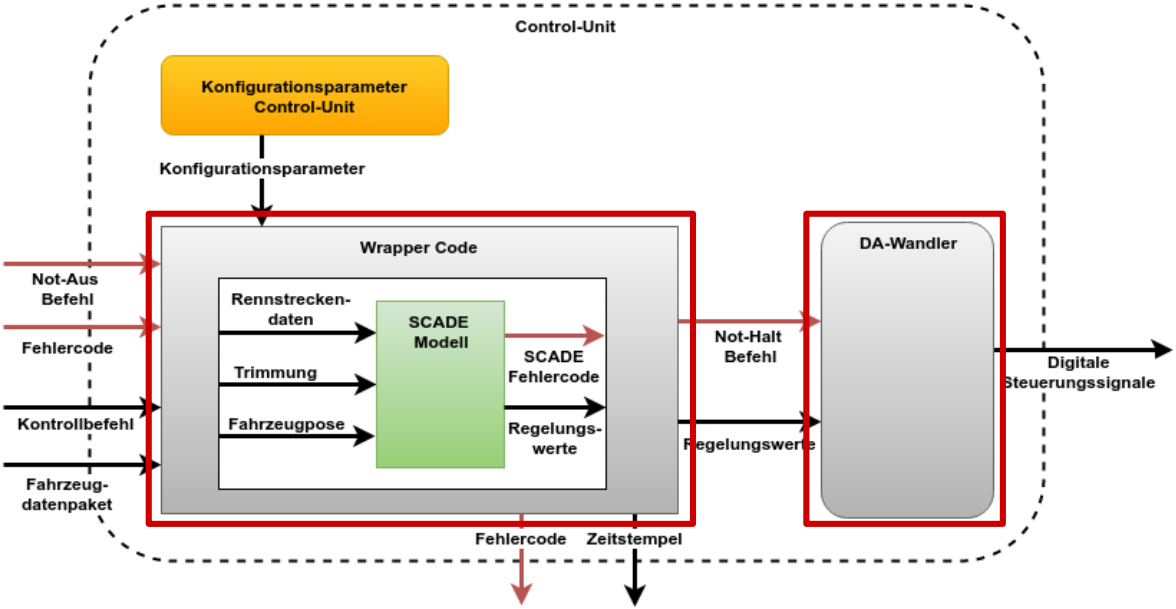
Sicherheitskonzept
Schadenszenario (Control-Unit)

- Control-Unit
  - Verbindung zum Fahrzeug geht verloren



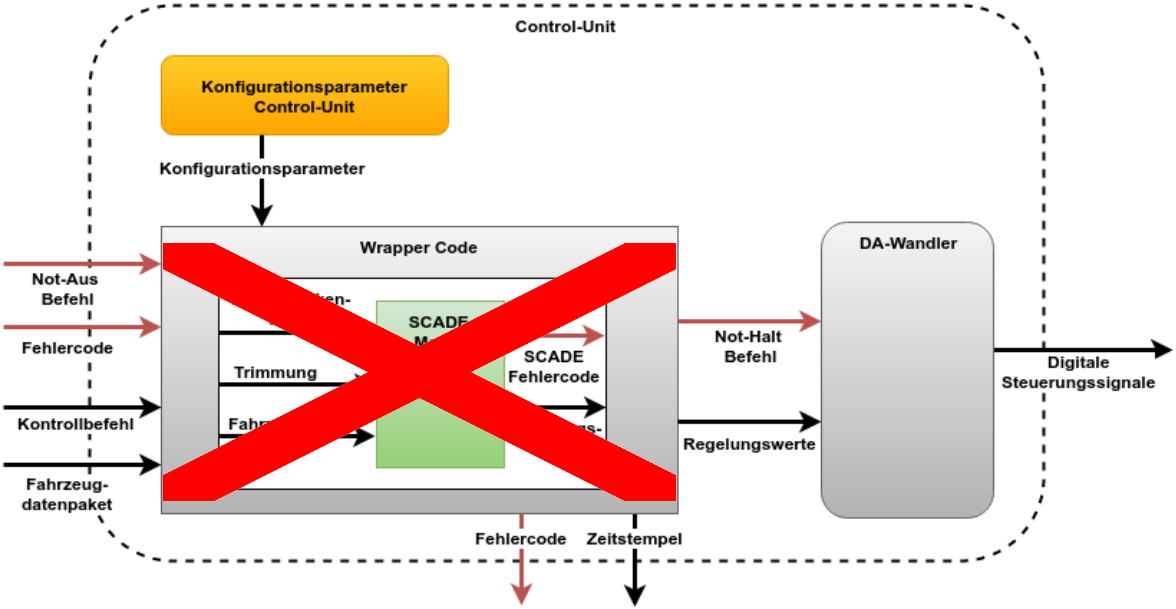
Sicherheitskonzept
Schadenszenario (Control-Unit)

- Control-Unit
  - Wrapper Code
  - DA-Wandler



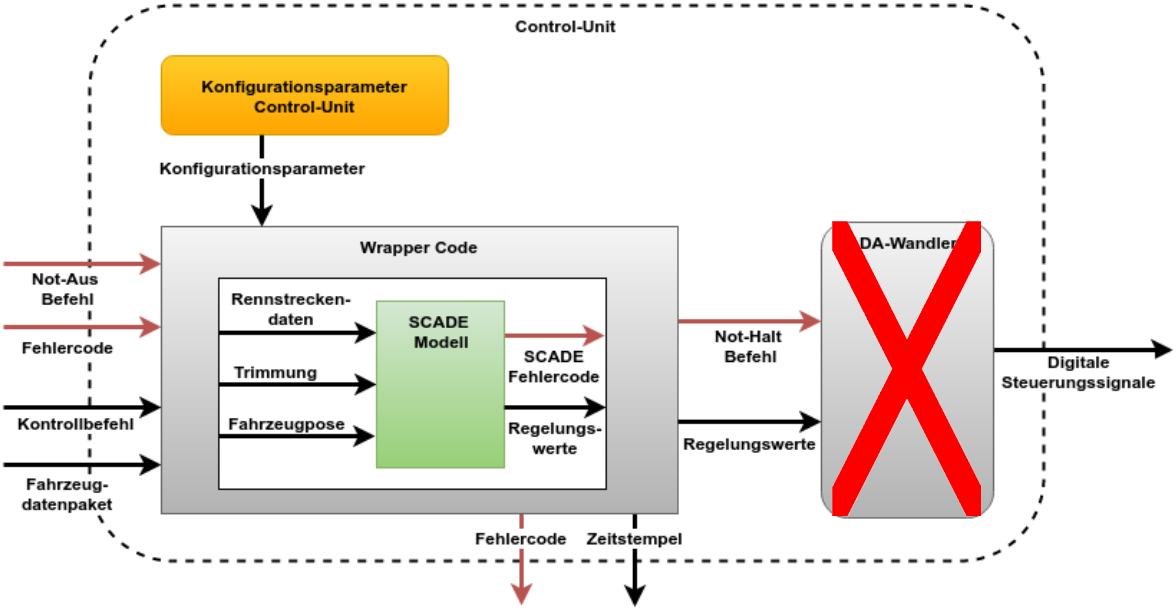
Sicherheitskonzept
Schadenszenario (Control-Unit)

- Control-Unit
  - Wrapper Code
  - DA-Wandler

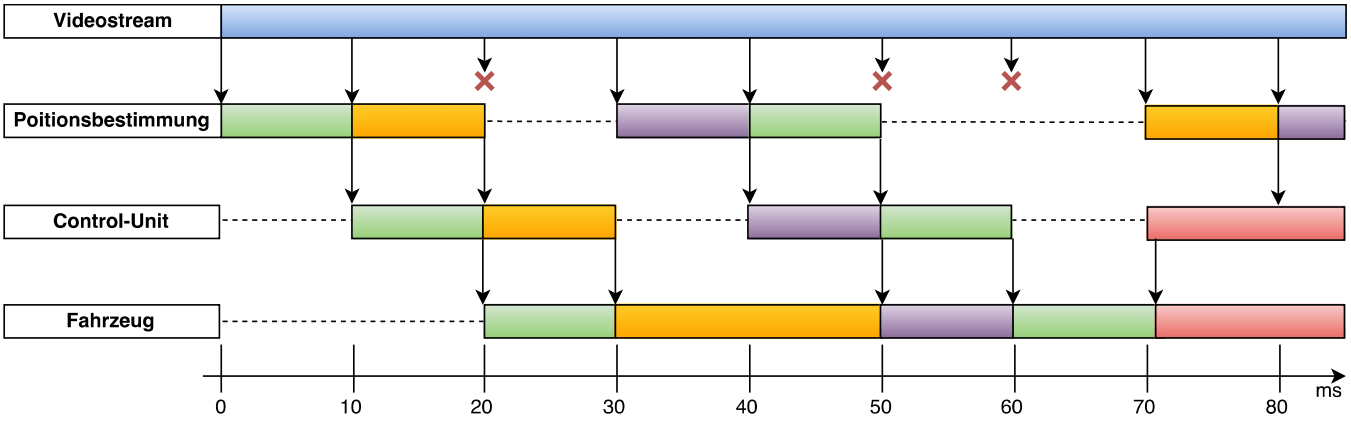


Sicherheitskonzept
Schadenszenario (Control-Unit)

- Control-Unit
  - Wrapper Code
  - DA-Wandler



Sicherheitskonzept
Realzeitanforderungen



Agenda

- Vorstellung
- Einleitung
- Projektorganisation
- Systemarchitektur und Schnittstellen
- Sicherheitskonzept
- Hardware-Evaluation
- Software-Entwicklung
- Testen
- Angepasste Anforderungen
- Aktueller Stand
- Ausblick
- Live-Demo

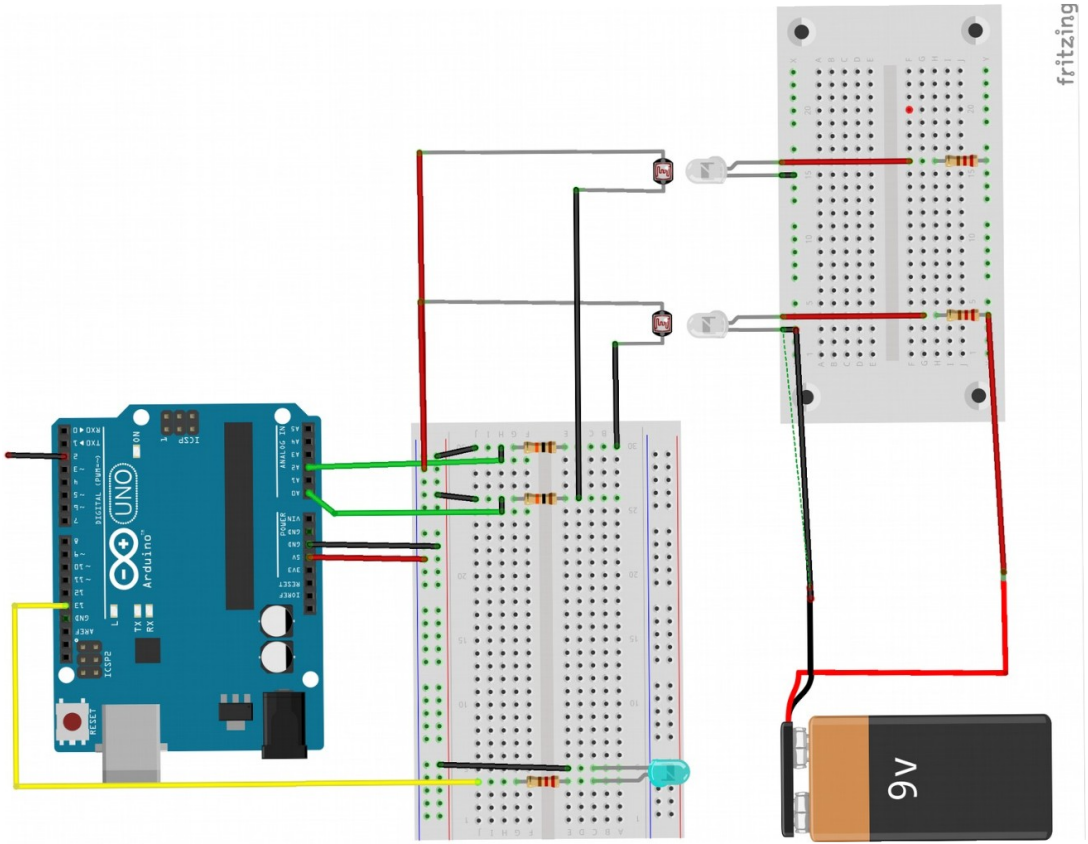
Hardware-Evaluation
Einstieg

- Hardware Anforderungen erfüllt?
- Müssen Hardware Anforderungen angepasst werden?
- Zuverlässig?
- Performant?

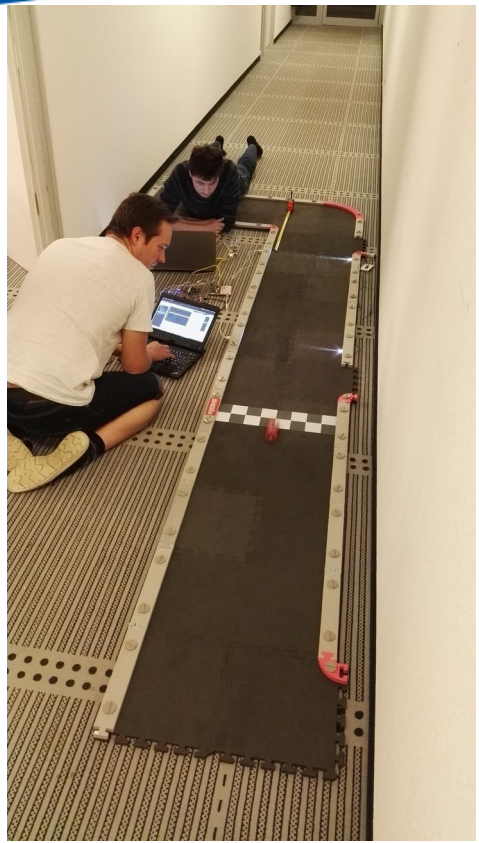
Hardware-Evaluation
Fahrzeug

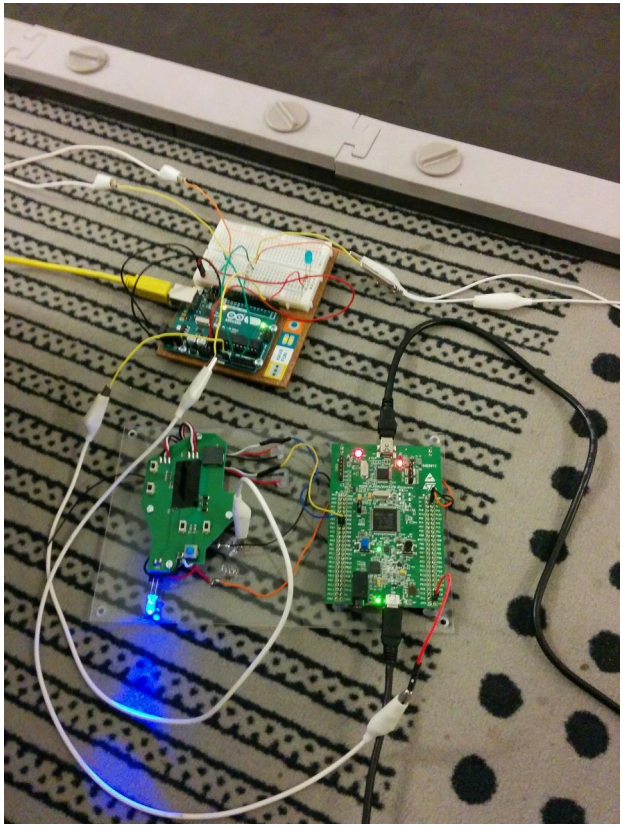
Realtime Controlled Cooperative Autonomous Racing System

Hardware-Evaluation
Fahrzeug

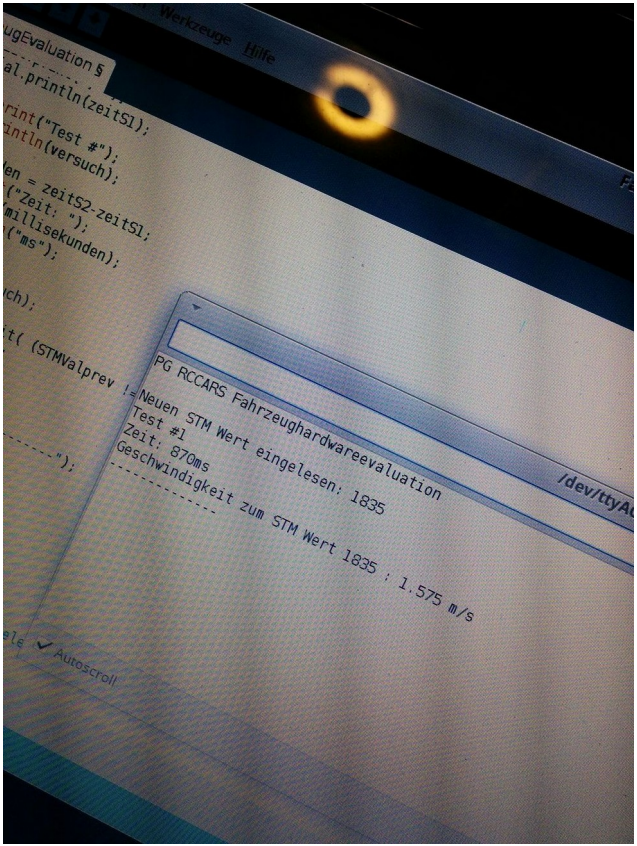


Hardware-Evaluation
Fahrzeug



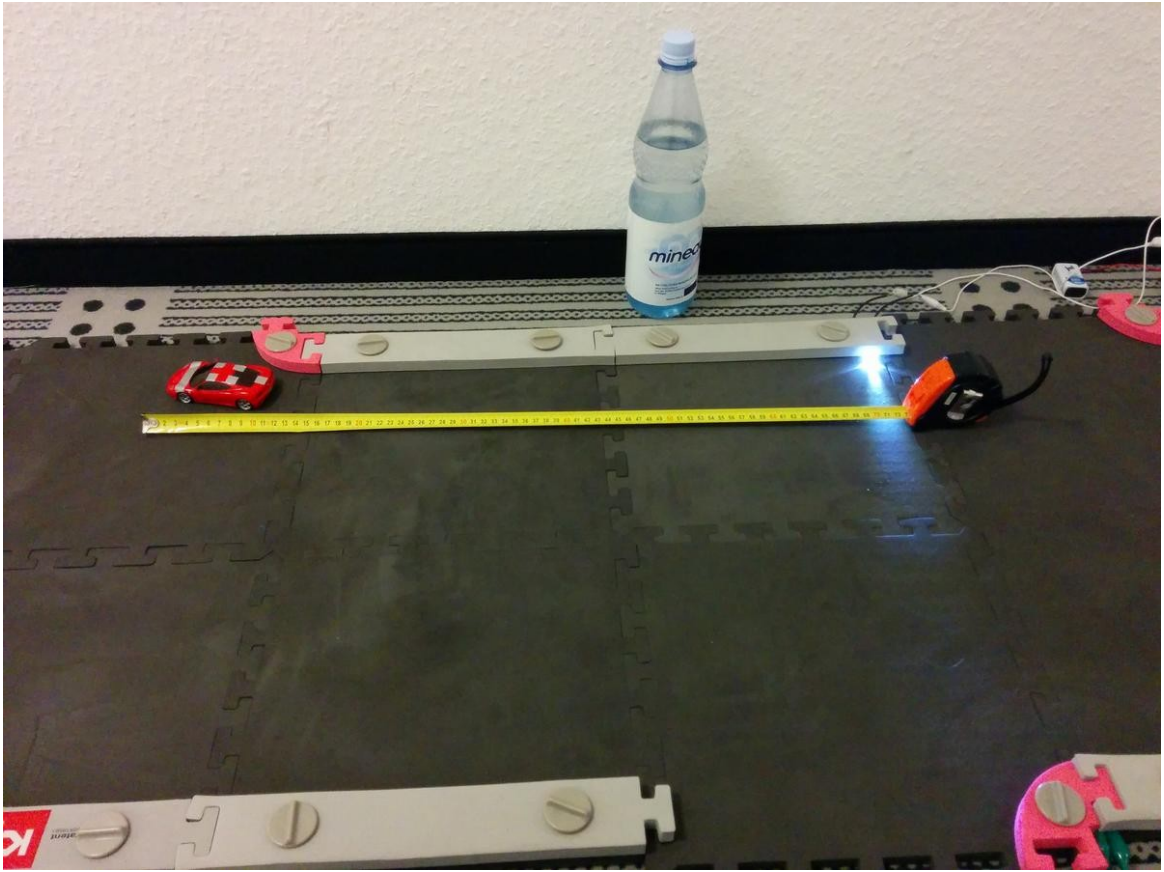


Hardware-Evaluation
Fahrzeug



Evaluation der Fahrzeuggeschwindigkeit	
Throttle	v in m/s
1300	3,470
1400	3,186
1500	3,139
1600	2,653
1700	2,185
1725	2,008
1750	1,926
1800	1,684
1825	1,539
1835	1,501
1840	1,480
1850	1,072
1875	0,999
1890	0,963
1900	0,097

Hardware-Evaluation
Fahrzeug



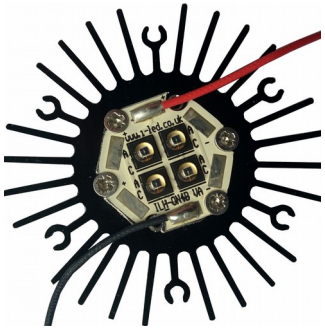
Hardware-Evaluation
Fahrzeug

Evaluation des Kurvenverhaltens	
v in m/s	Kurve ohne Kollision umfahren?
2,157	Nein
1,997	Nein
1,911	Nein
1,859	Nein
1,844	Nein
1,834	Nein
1,820	Nein
1,739	Ja
1,612	Nein
1,603	Nein
1,593	Ja
1,555	Ja
1,538	Nein
1,521	Ja
1,519	Nein
1,394	Ja
1,286	Ja
1,250	Ja
1,163	Ja
1,021	Ja
0,963	Ja
0,700	Ja

Evaluation des Bremswegs	
v in m/s	Bremsweg in cm
3,470	>110
3,186	>110
3,139	>110
2,653	>100
2,231	110
2,088	104
1,968	96,5
1,911	100,5
1,612	73,5
1,539	74
1,470	72
0,999	42
0,963	40
0,097	< 40

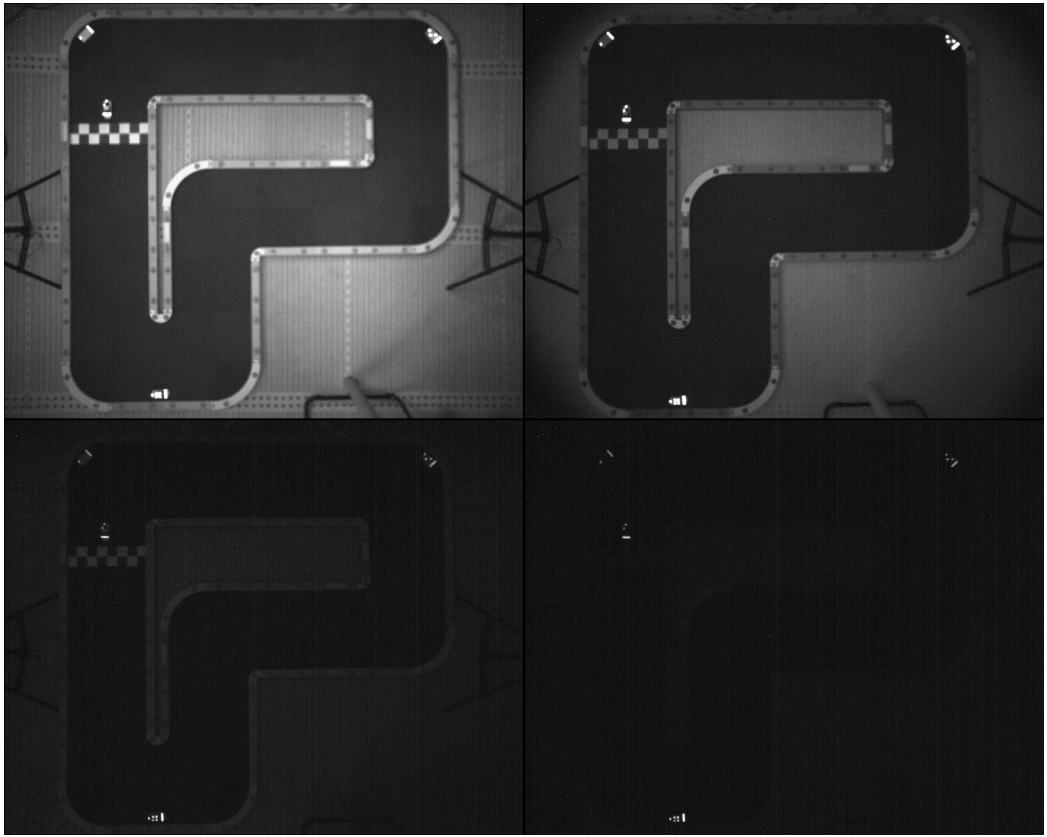
Hardware-Evaluation
Positionsbestimmung

- Kamera
- Objektiv
- Filter
- Gestell
- Beleuchtung

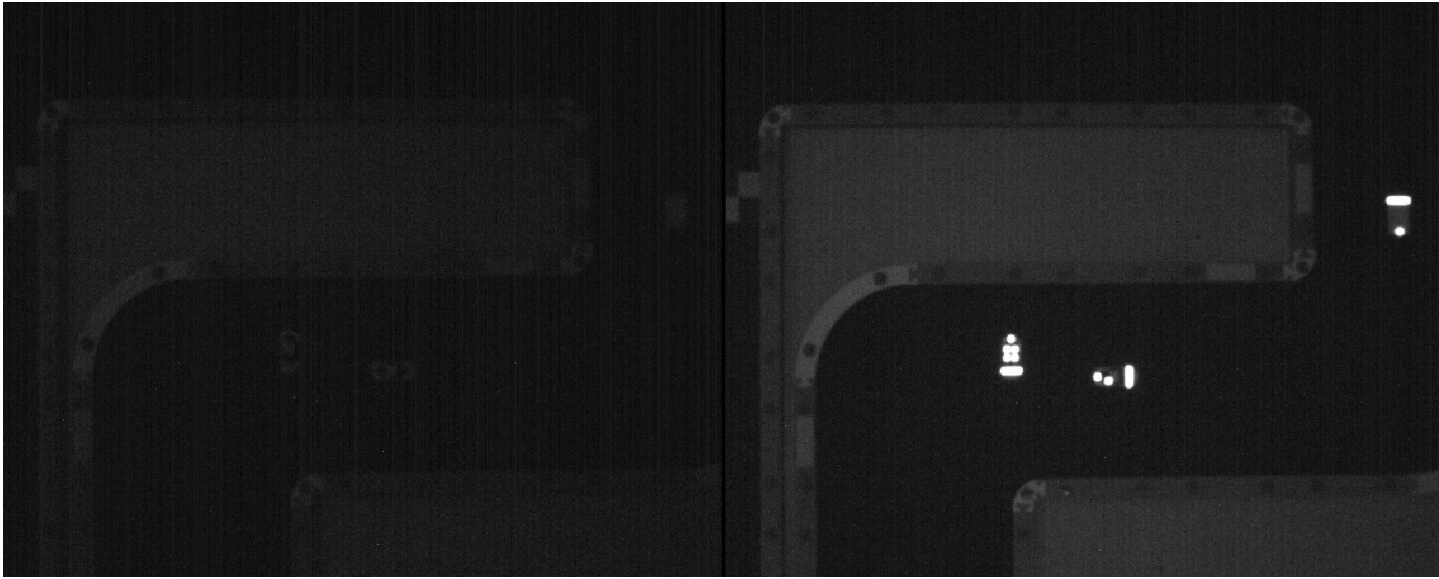


Realtime Controlled Cooperative Autonomous Racing System

Hardware-Evaluation
Positionsbestimmung



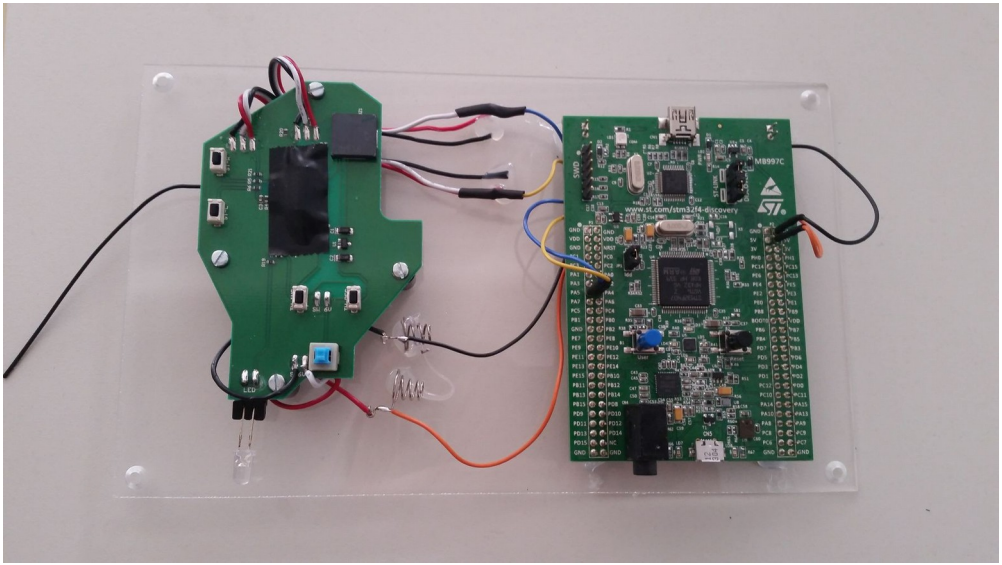
Hardware-Evaluation
Positionsbestimmung



Realtime Controlled Cooperative Autonomous Racing System

Hardware-Evaluation
Control-Unit

- Rechner
- DA-Wandler
- Fernsteuerung



Agenda

- Vorstellung
- Einleitung
- Projektorganisation
- Systemarchitektur und Schnittstellen
- Sicherheitskonzept
- Hardware-Evaluation
- Software-Entwicklung
- Testen
- Angepasste Anforderungen
- Aktueller Stand
- Ausblick
- Live-Demo

Software-Entwicklung

- Positionsbestimmung
- Kommunikation
- Control-Unit
  - Wrapper Code
  - DA-Wandlungssoftware
  - SCADE Modell
- Systemsteuerungssoftware

Software-Entwicklung
Einleitung

- C / C++
- Eclipse, Coocox, Arduino IDE, Scade
- Modularer Aufbau
- Erweiterbarkeit
- Wartbarkeit
- Performanz
- Dokumentation

Software-Entwicklung

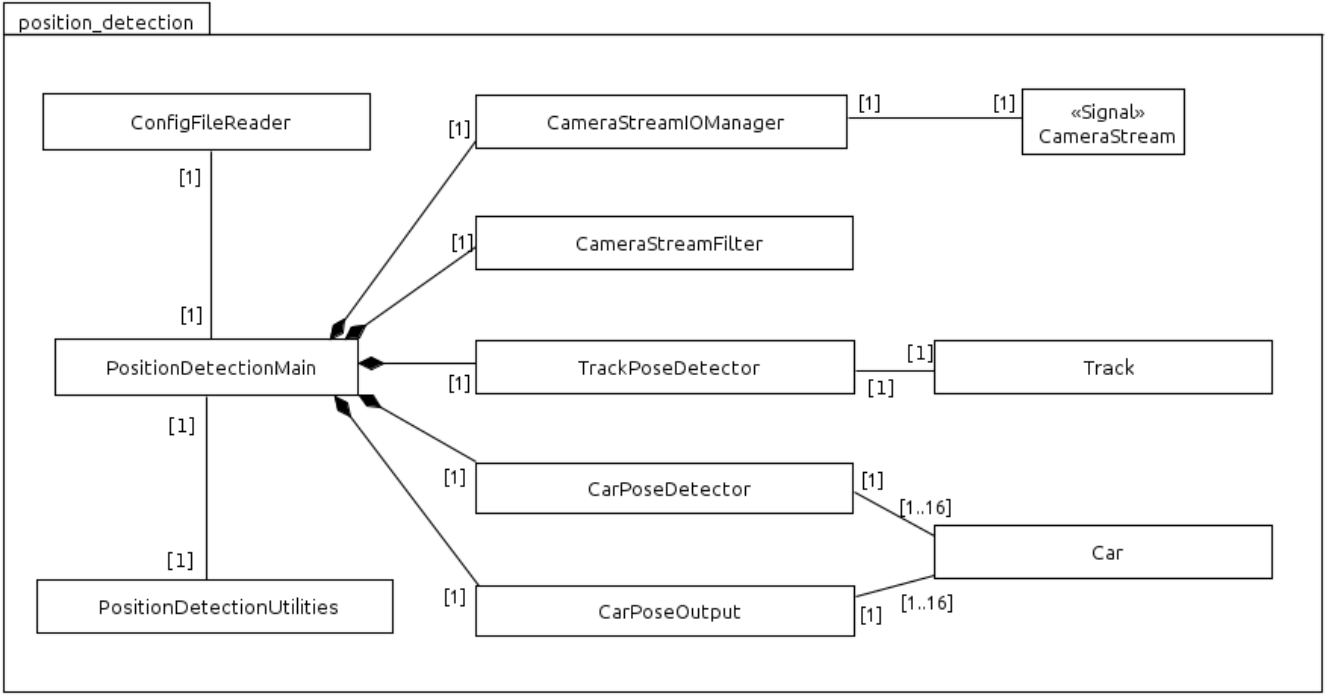
- Positionsbestimmung
- Kommunikation
- Control-Unit
  - Wrapper Code
  - DA-Wandlungssoftware
  - SCADE Modell
- Systemsteuerungssoftware

Realtime Controlled Cooperative Autonomous Racing System

Software-Entwicklung
Positionsbestimmungssoftware Einstieg

- Erfassung der Fahrzeugpose
  - Kamerabild einlesen
  - Kamerabild bearbeiten
  - Objekterkennung durchführen
- Erfassung Rennstreckenpose
- Fahrzeugpose versenden
- Fehler melden

Software-Entwicklung
Positionsbestimmungssoftware Übersicht



Software-Entwicklung
CameraStreamIOManager

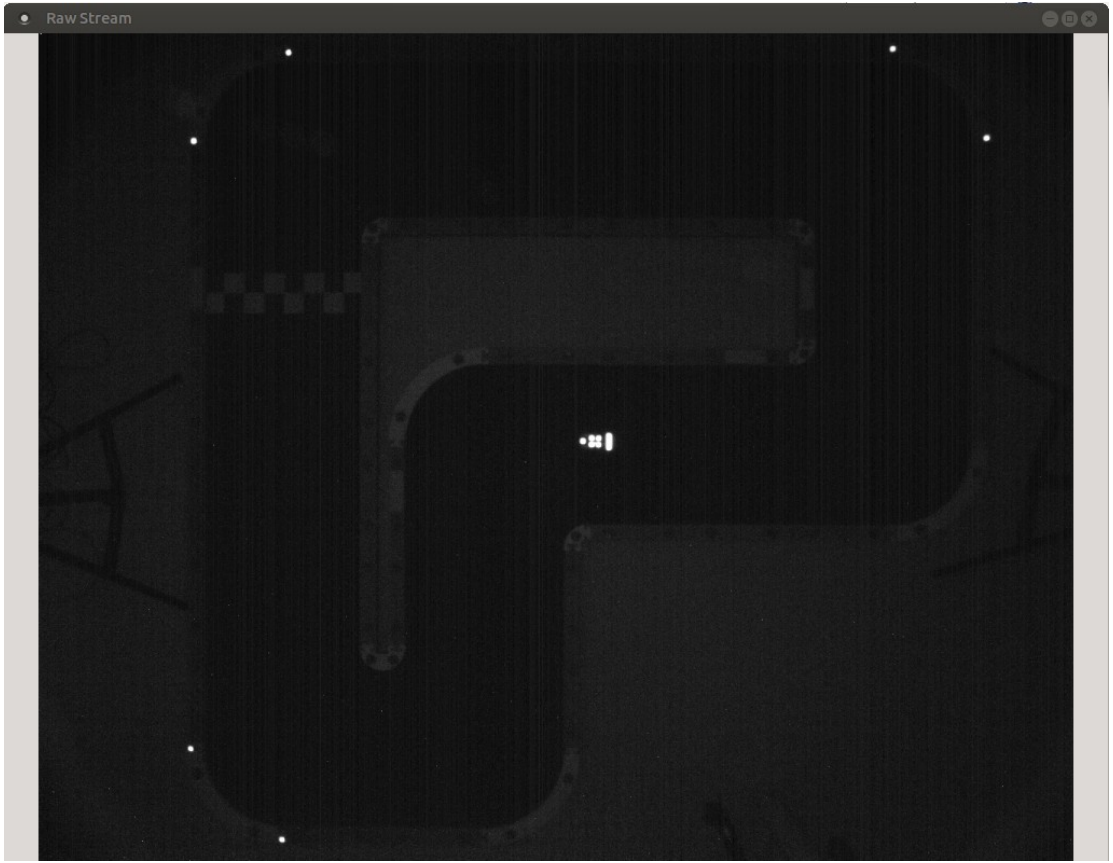
- Verbindung mit Kamera herstellen
- Einlesen des Kamerastreams

```
Mat CameraStreamIOManager::readVideoStream(Camera* camera) {
    Image rawImage;
    Mat camStream;

    camera->RetrieveBuffer(&rawImage);
    unsigned int rowBytes = (double) rawImage.GetReceivedDataSize()
            / (double) rawImage.GetRows();

    /*CV_8UC1  8-bit single channel image*/
    camStream = Mat(rawImage.GetRows(), rawImage.GetCols(), CV_8UC1,
            rawImage.GetData(), rowBytes);

    return camStream;
}
```

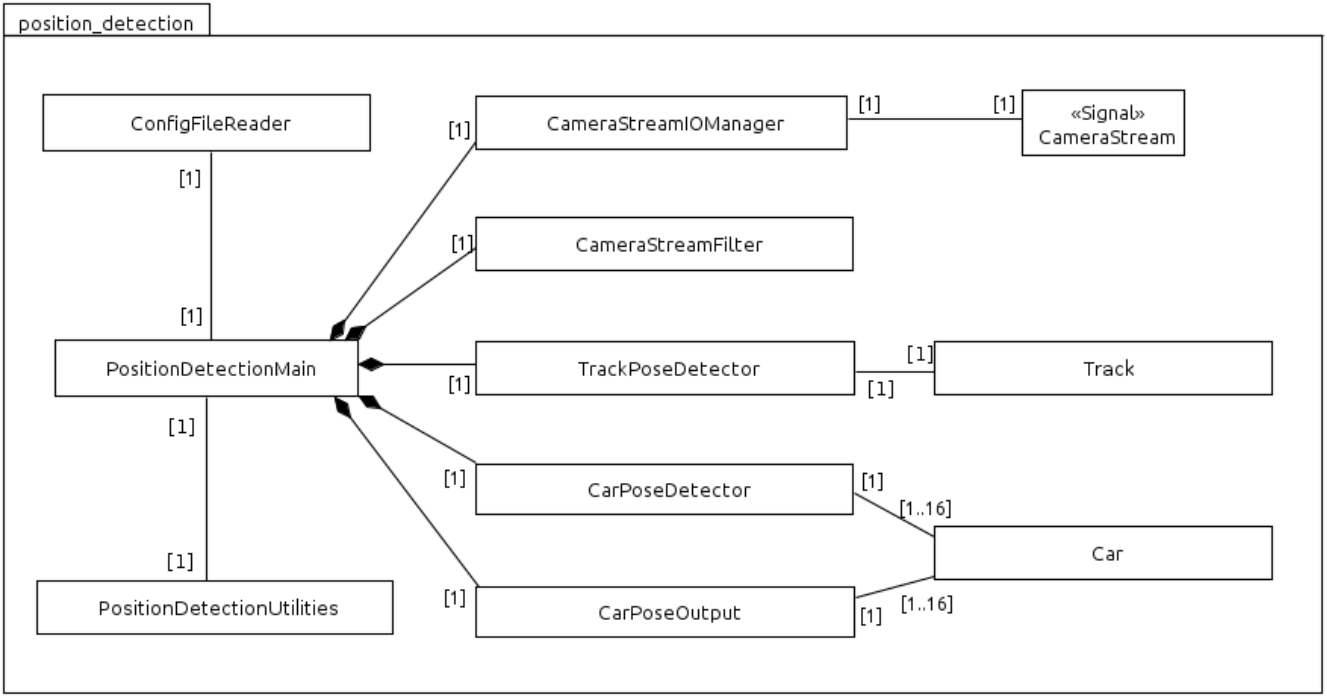
- Fenstererzeugung
- Bildausgabe

Realtime Controlled Cooperative Autonomous Racing System

Software-Entwicklung
CameraStreamIOManager



Software-Entwicklung
Positionsbestimmungssoftware Übersicht



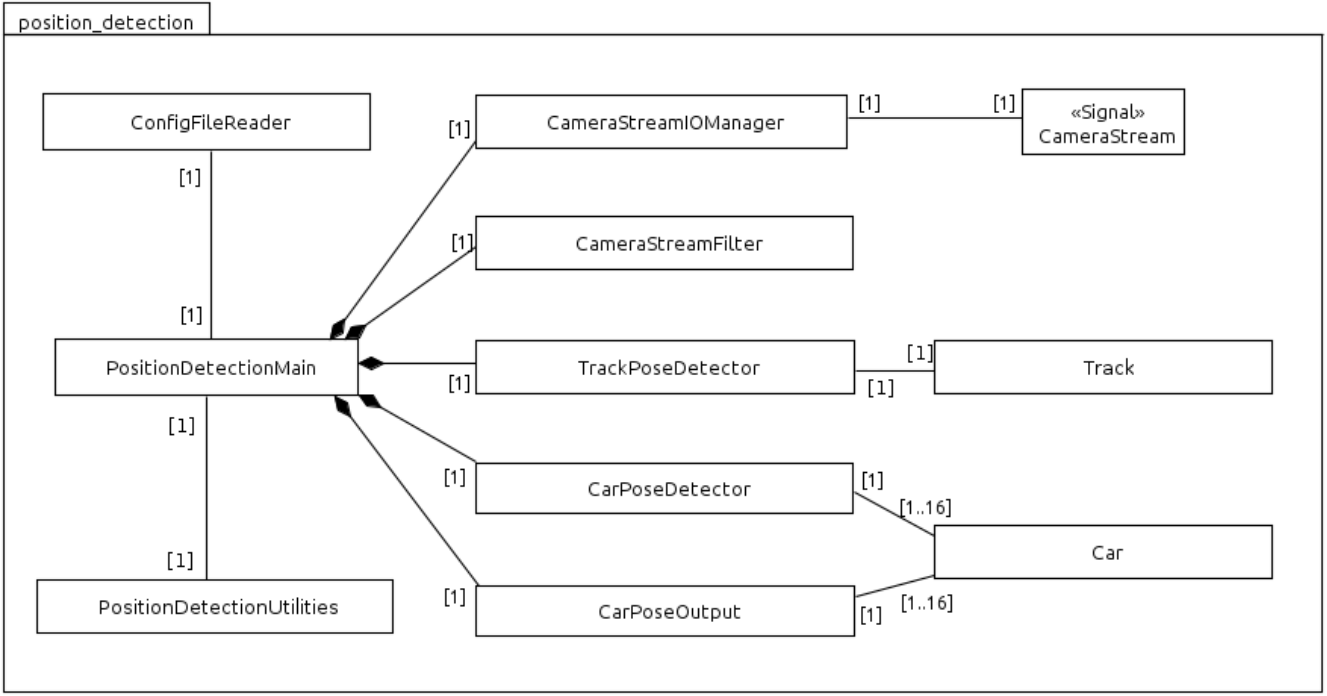
Software-Entwicklung
CameraStreamFilter

- Bearbeitung der Bilder
- Umwandlung Binärbild
- Vorbereitung Objekterkennung
- Threshold Verfahren
- Blur
- Dilate und Erode

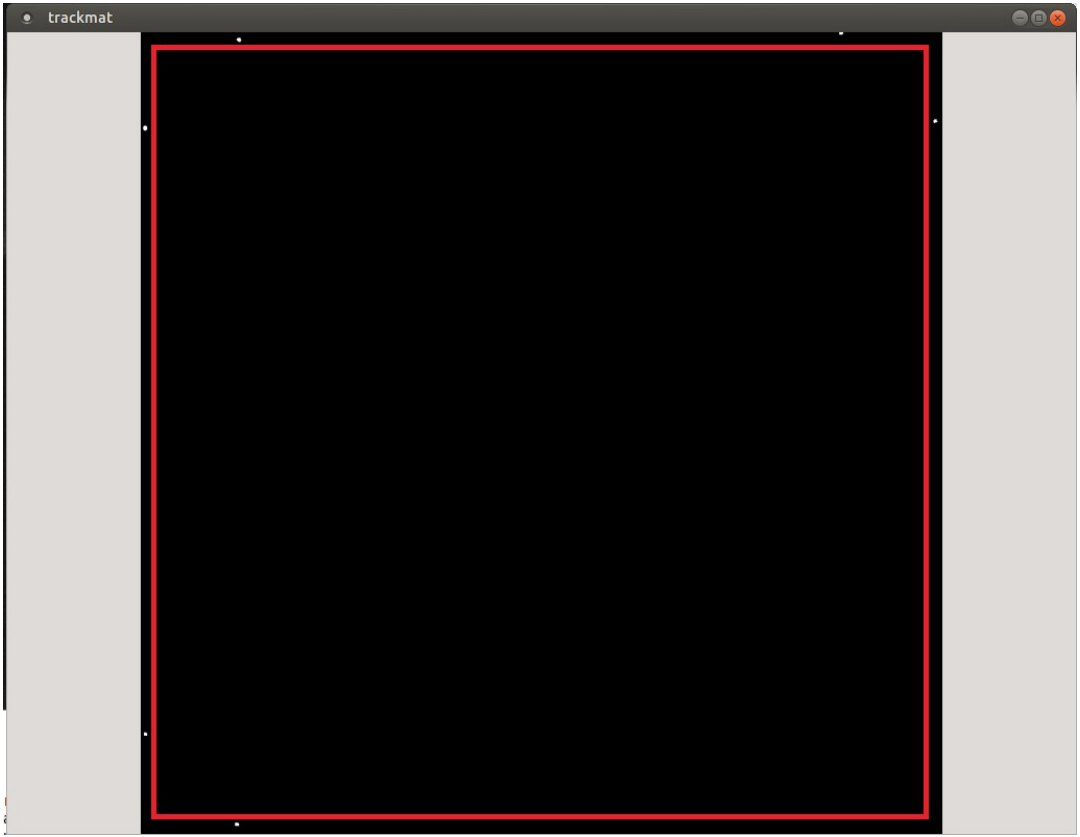
Software-Entwicklung
CameraStreamFilter

```
109 Mat CameraStreamFilter::applyBinaryPictureFilterThreshold(Mat input, Mat output,
110         int thresholdLowerBound, int thresholdUpperBound, int thresholdType,
111         int blurValue, int inRangeLowerBound, int inRangeUpperBound,
112         bool useInRange, bool useMorphOps, CameraStreamIOManager *camIO) {
113     threshold(input, output, thresholdLowerBound, thresholdUpperBound,
114             thresholdType);
115     medianBlur(output, output, (2 * blurValue) + 1);
116     if (useInRange) {
117         inRange(output, inRangeLowerBound, inRangeUpperBound, output);
118     }
119     if (useMorphOps) {
120 
121         applyMorphologicalOperation(output, camIO->getErodeValueLight(),
122                 camIO->getDilateValueLight());
123     }
124 
125     return output;
126 }
127 
```

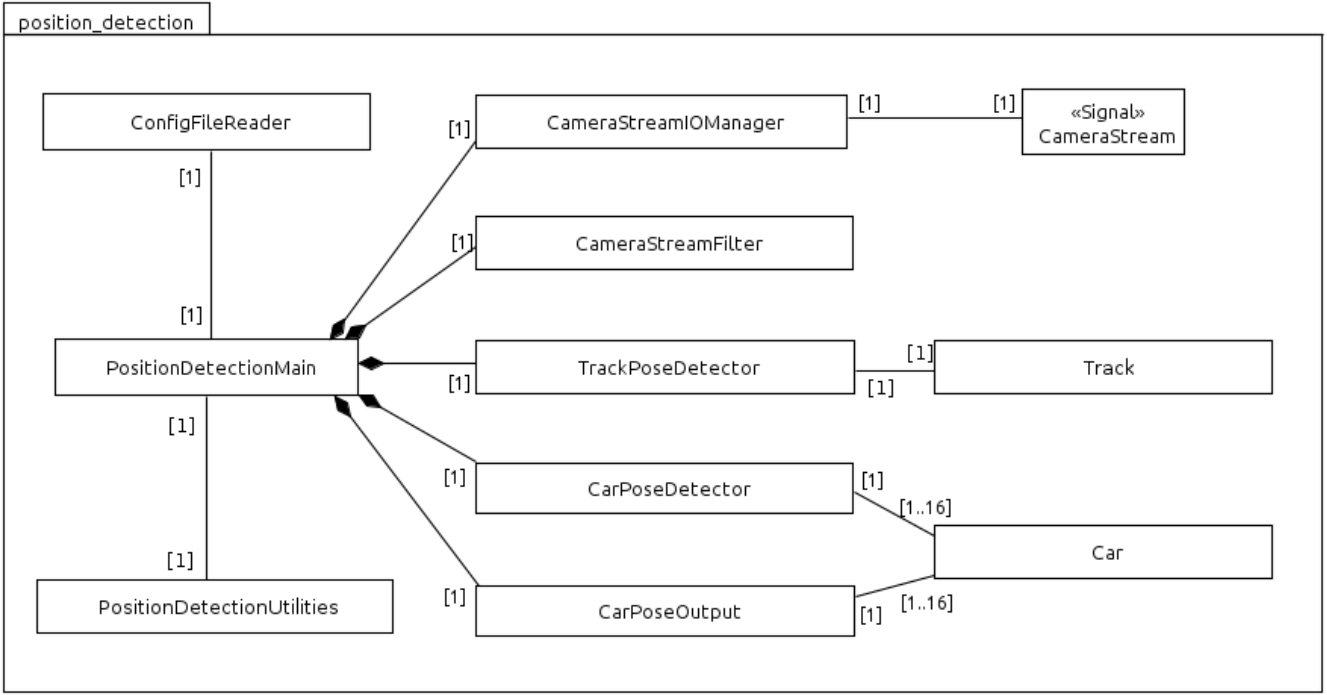
Software-Entwicklung
Positionsbestimmungssoftware Übersicht



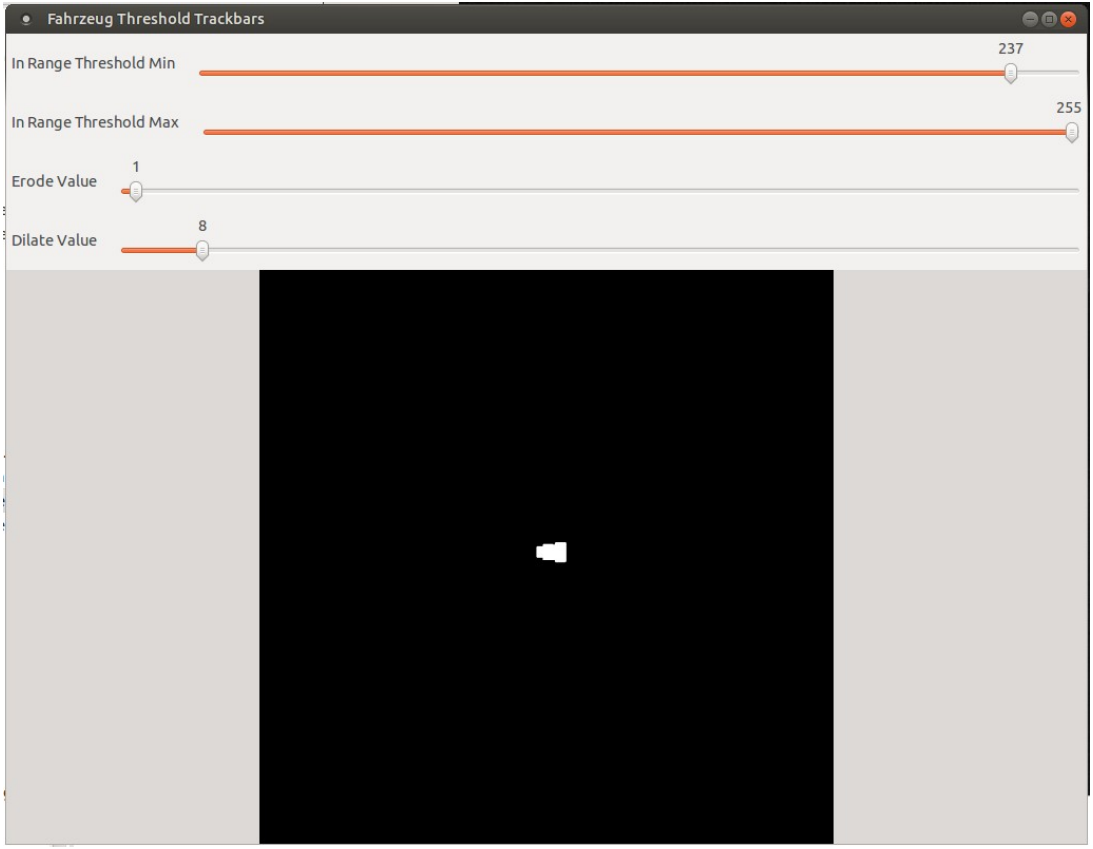
Software-Entwicklung
TrackPoseDetector



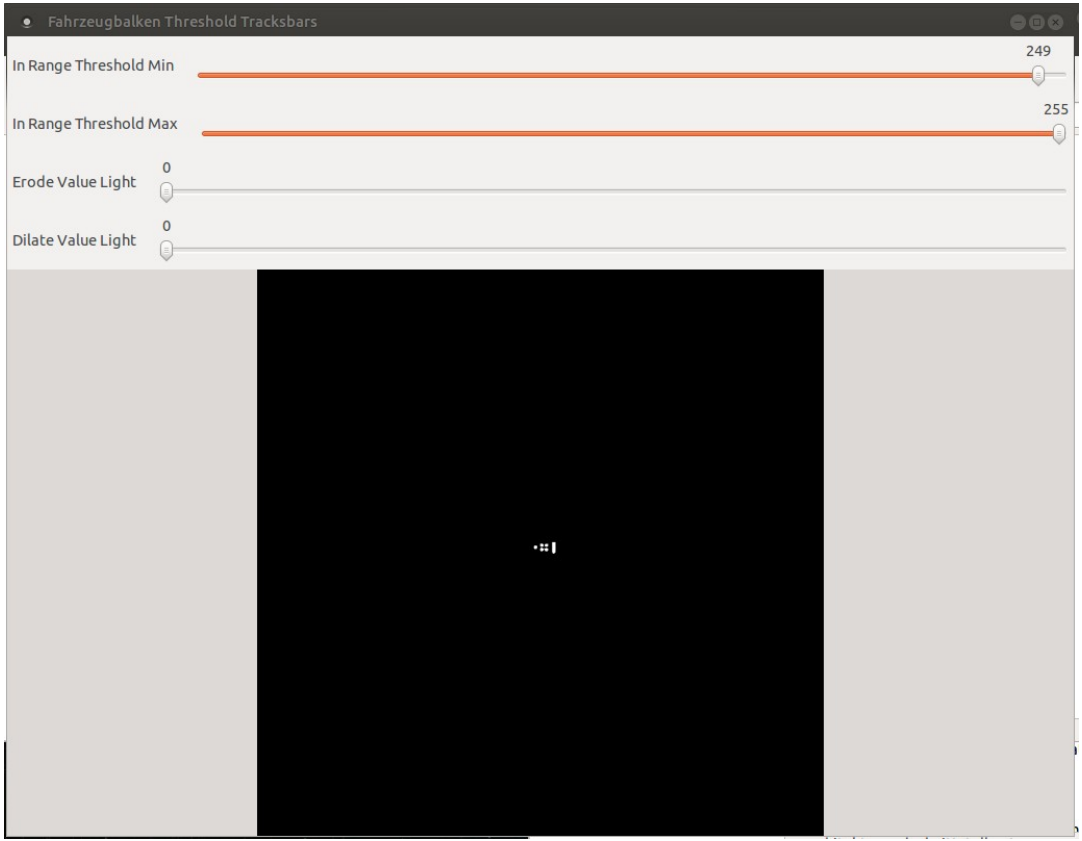
Software-Entwicklung
Positionsbestimmungssoftware Übersicht



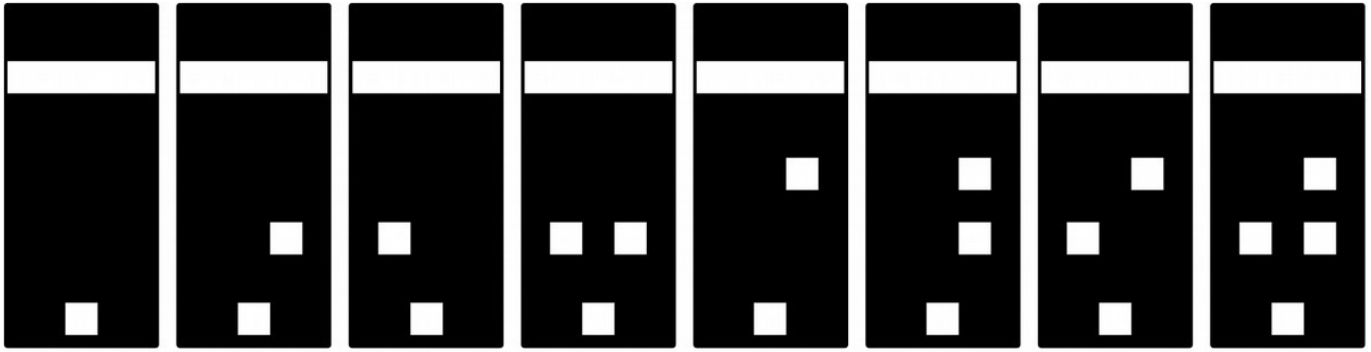
Software-Entwicklung
CarPoseDetector

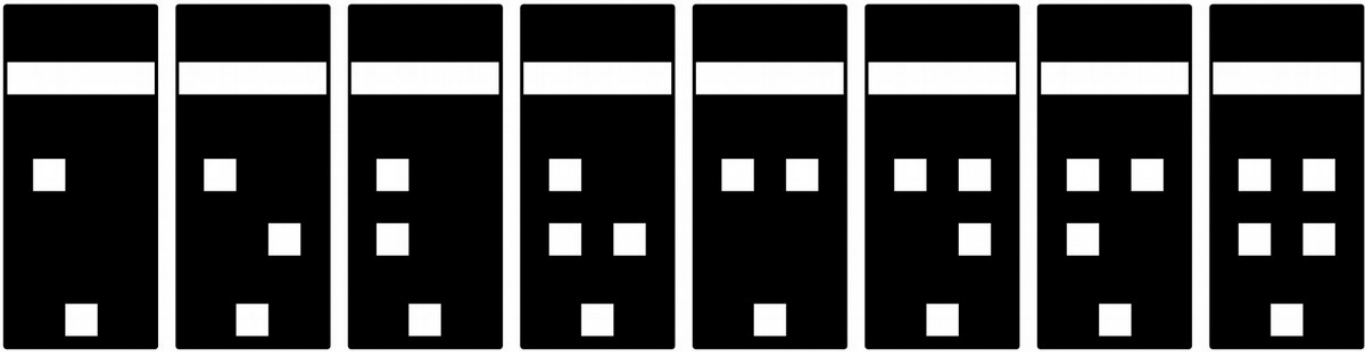


Software-Entwicklung
CarPoseDetector

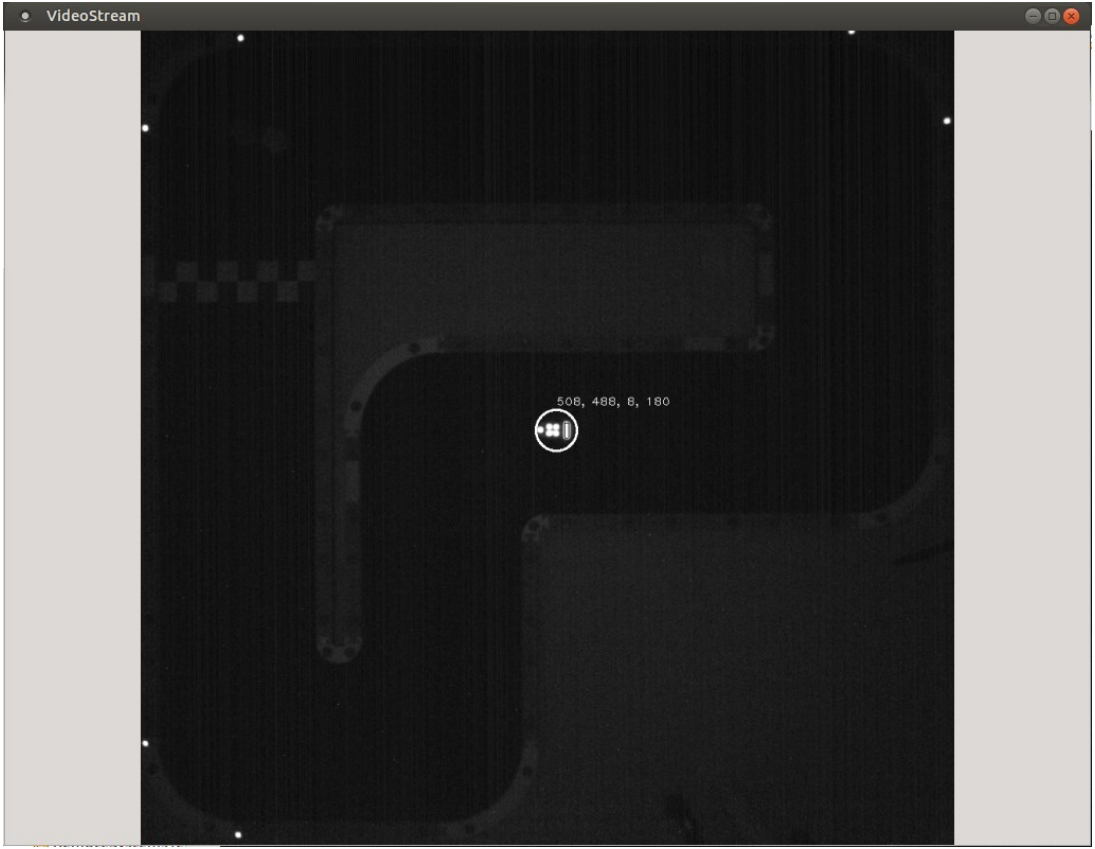


Software-Entwicklung
CarPoseDetector

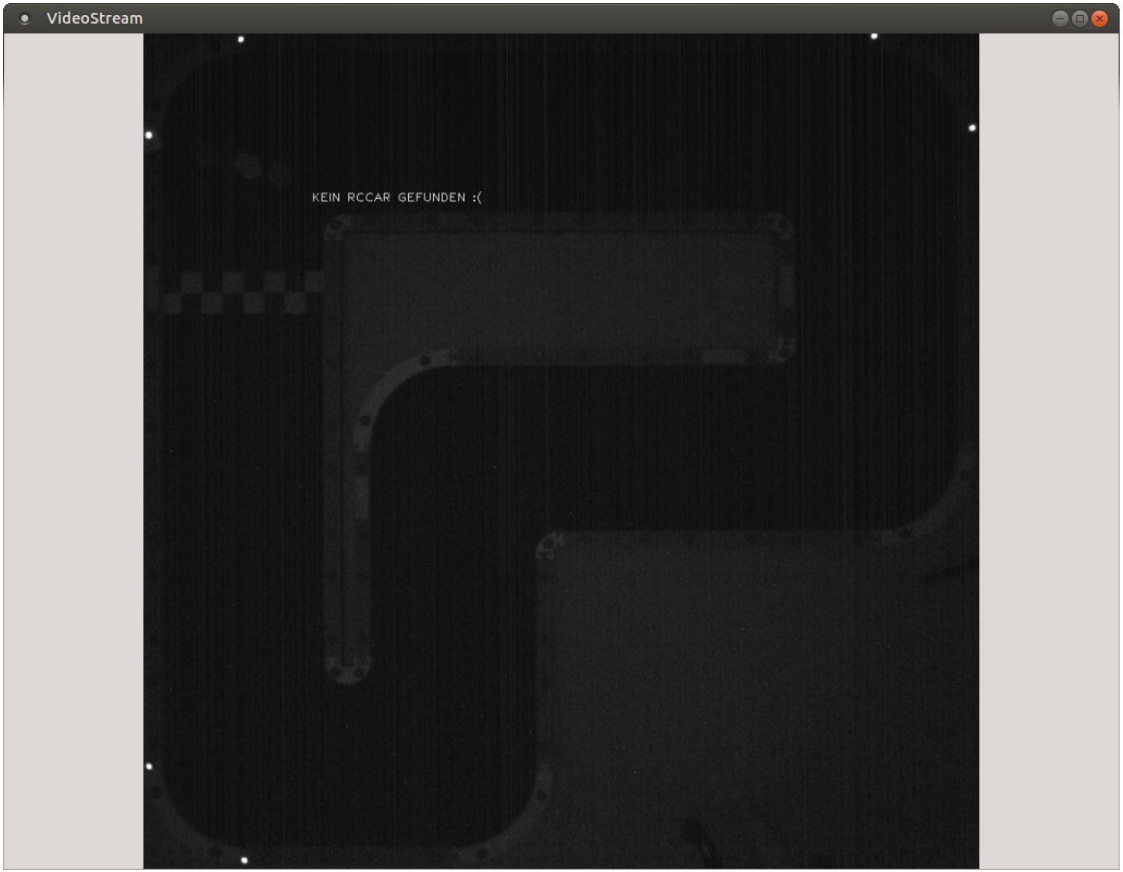




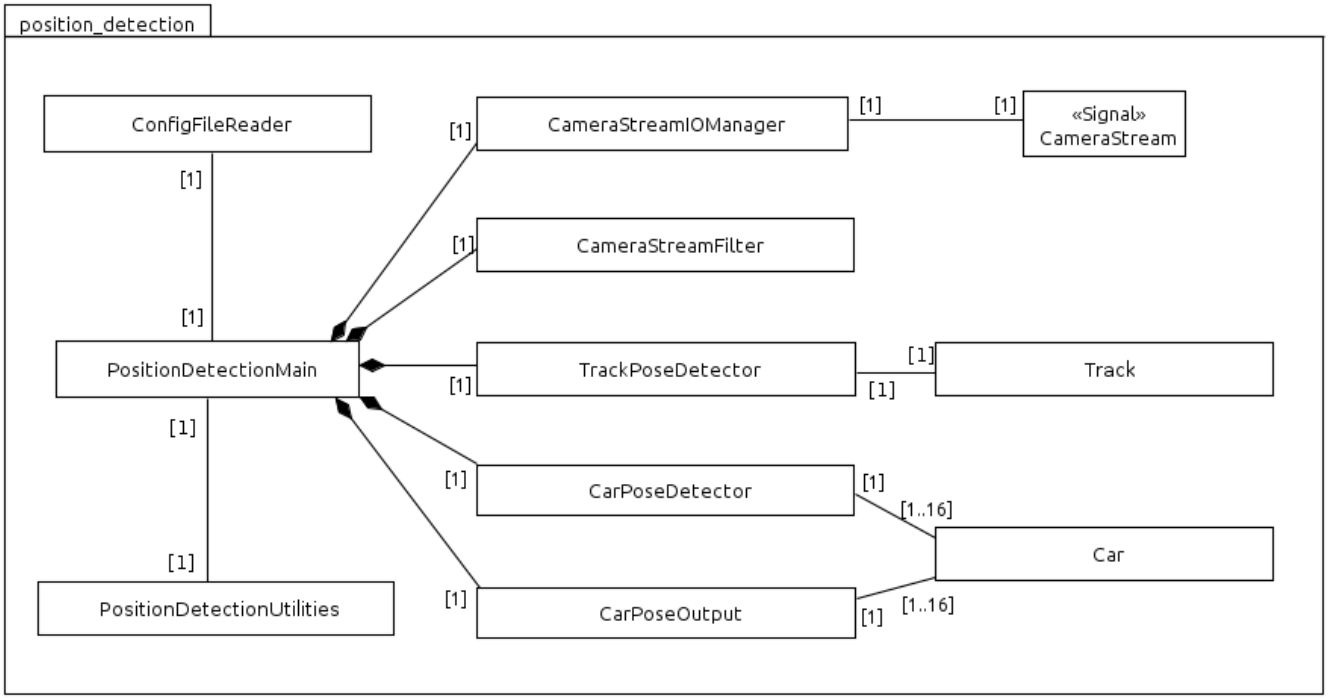
Software-Entwicklung
VideoStream



Software-Entwicklung
VideoStream



Software-Entwicklung
Positionsbestimmungssoftware Übersicht



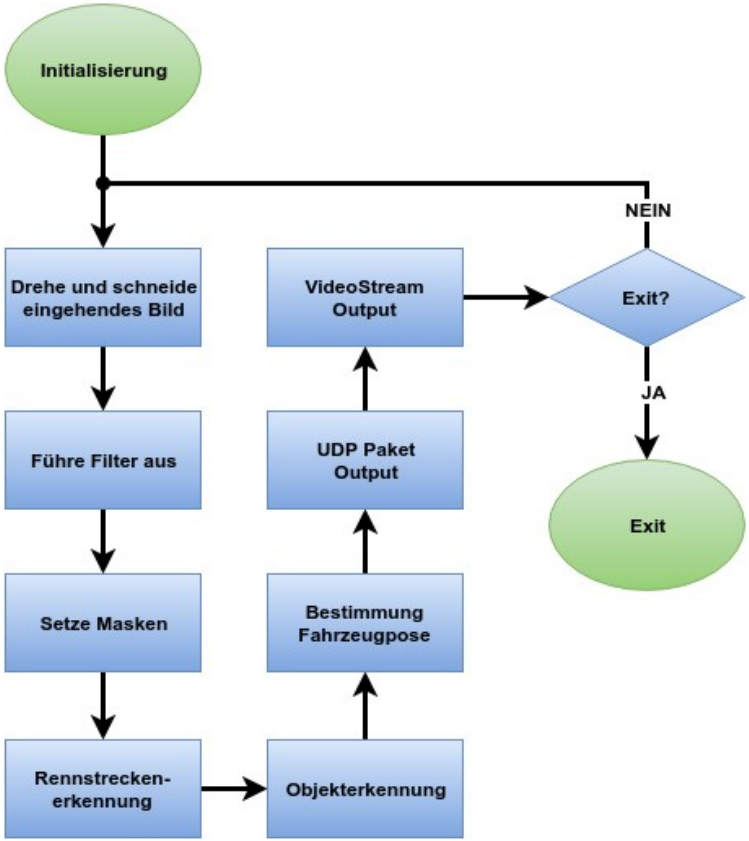
Software-Entwicklung
CarPoseOutput

rccars@dell-rccars: ./DemoControlUnit

Datei Bearbeiten Ansicht Suchen Terminal Hilfe

```
Identifier: 1
Timestamp: 1466775677 602846

Vehicle         VehicleCheck    X-Koords        Y-Koords        Alignments
#0              1               116             116             94.763642
#1              0               0               0               0.000000
#2              0               0               0               0.000000
#3              0               0               0               0.000000
#4              0               0               0               0.000000
#5              0               0               0               0.000000
#6              0               0               0               0.000000
#7              0               0               0               0.000000
#8              0               0               0               0.000000
#9              0               0               0               0.000000
#10             0               0               0               0.000000
#11             0               0               0               0.000000
#12             0               0               0               0.000000
#13             0               0               0               0.000000
#14             0               0               0               0.000000
#15             0               0               0               0.000000
```

Software-Entwicklung
Ablauf



Software-Entwicklung
Laufzeit und Optimierung

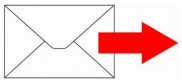
- Laufzeit
  - 70 ms mit Debug Ausgaben und Fenstern
  - 26 ms ohne Debug Ausgaben und einem Fenster
- Optimierung
  - Viele Operationen auf großen Matrizen
    - Fahrzeugmarker, Fahrzeugbalken und Rennstreckenmarker jeweils auf eigener Matrix gesucht
  - Objekterkennung sucht immer im ganzen Bild
  - Kaum Pointer

Software-Entwicklung

- Positionsbestimmung
- Kommunikation
- Control-Unit
  - Wrapper Code
  - DA-Wandlungssoftware
  - SCADE Modell
- Systemsteuerungssoftware

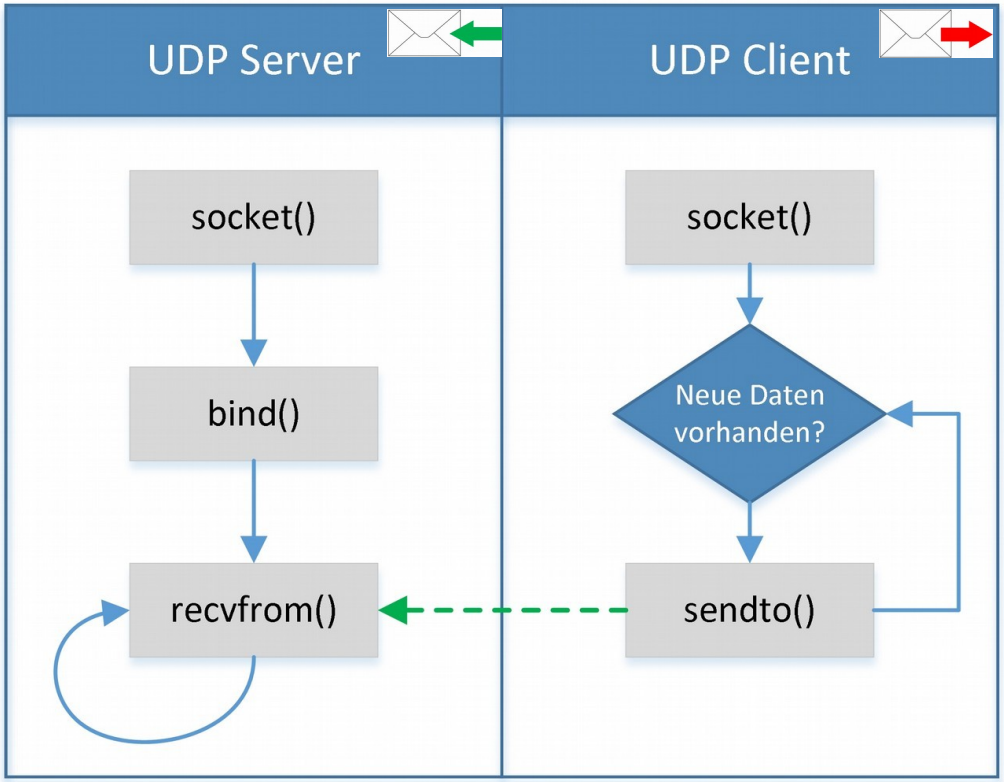
Realtime Controlled Cooperative Autonomous Racing System

Kommunikation

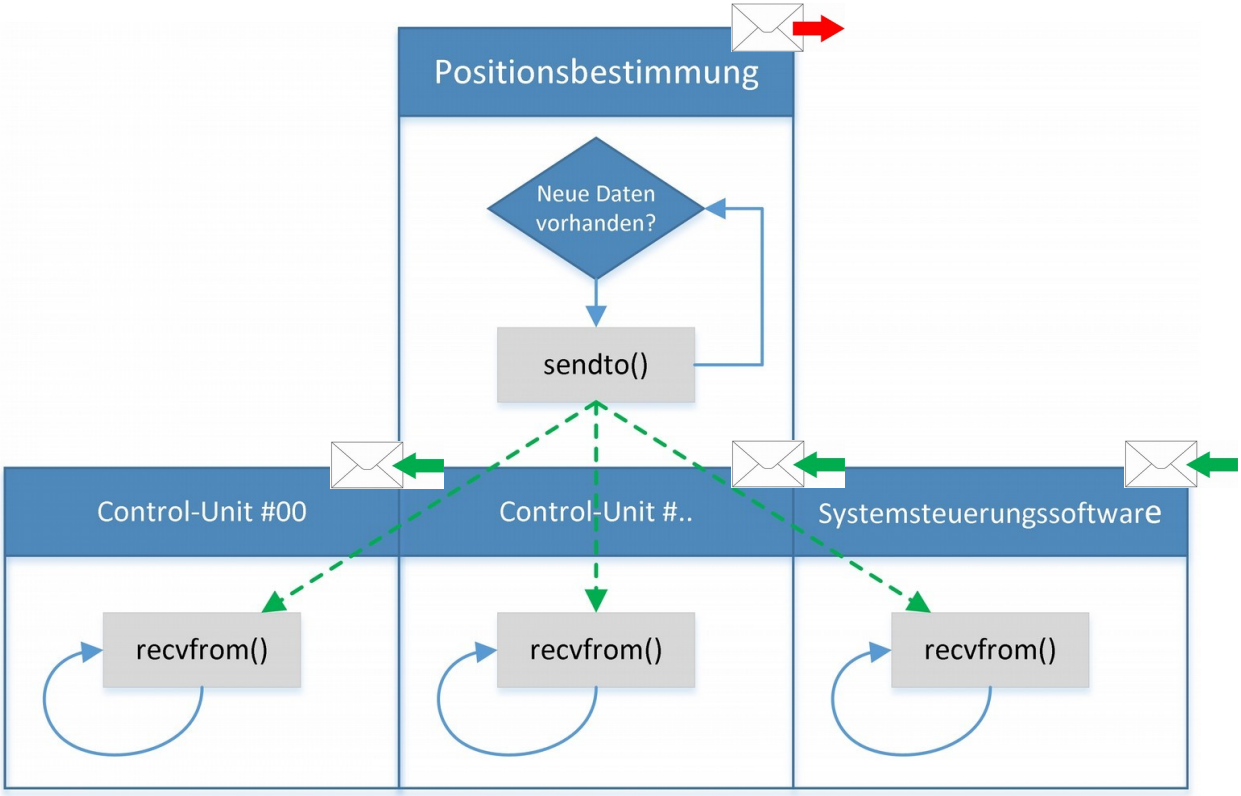
- Realisierung mit UDP-Sockets
  - UDP Client zum Senden 
  - UDP Server zum Empfangen
- Implementierung in C
  - Universaler Einsatz gewährleistet in
    - Positionsbestimmung (C++)
    - Control-Unit (C)
    - Systemsteuerungssoftware (C/C++)

Realtime Controlled Cooperative Autonomous Racing System

Kommunikation
UDP Netzwerkdatenpakete



Kommunikation
UDP Netzwerkdatenpakete



Kommunikation
Definiertes Netzwerkdatenpaket

```
1.  struct Networkpackage {
2.      int identifier;
3.
4.      struct timeval dataTimestamp;
5.      int availableCardata[16];
6.      int xCoord[16];
7.      int yCoord[16];
8.      double alignment[16];
9.
10.     int controlSignal;
11.
12.     int emergencyStop;
13.     int faultCode;
14.
15.     struct timeval startTimeStamp;
16.     struct timeval endTimestamp;
17.
18.     int checksum;
19. }
```

`int identifier;`	Identifikator

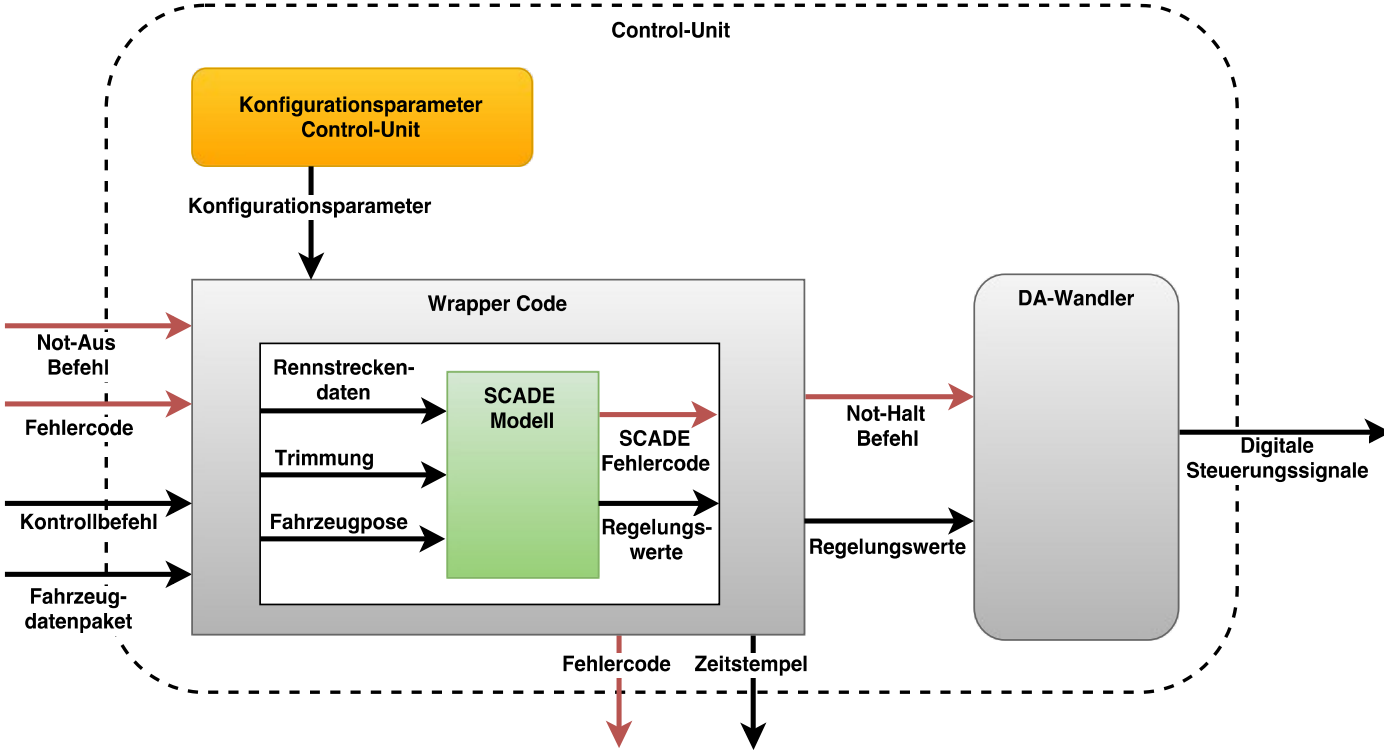
`struct timeval dataTimestamp;`
`int availableCardata[16];`
`int xCoord[16];`
`int yCoord[16];`
`double alignment[16];`
	(1) Fahrzeugdaten

`int controlSignal;`	(2) Kontrollbefehl
`int emergencyStop;` `int faultCode;`	(3) Fehlerdaten
`struct timeval startTimeStamp;` `struct timeval endTimestamp;`	(4) Zeitstempel
`int checksum;`	Prüfsumme

Software-Entwicklung

- Positionsbestimmung
- Kommunikation
- Control-Unit
  - Wrapper Code
  - DA-Wandlungssoftware
  - SCADE Modell
- Systemsteuerungssoftware

Software-Entwicklung
Control-Unit



Software-Entwicklung

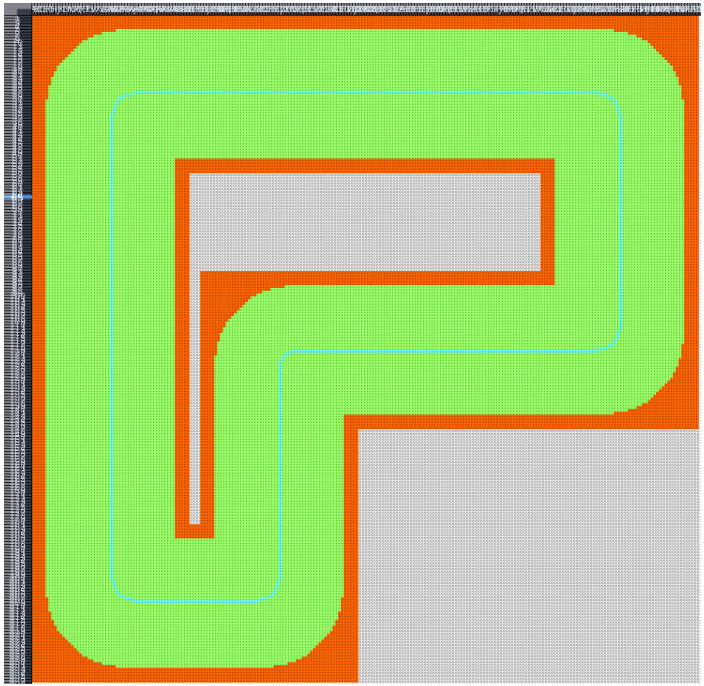
- Positionsbestimmung
- Kommunikation
- Control-Unit
  - Wrapper Code
  - DA-Wandlungssoftware
  - SCADE Modell
- Systemsteuerungssoftware

Software-Entwicklung
Wrapper-Code

- Aktueller Funktionsumfang
  - Einlesen von Rennstreckendaten aus Datei
  - Emfpang der Fahrzeugdatenpaket über das Netzwerk
    - Multithreading
  - Umgang mit SCADE Modell
    - Eingabevariablen verknüpfen
    - Ausführen
    - Ausgabevariablen auslesen
  - Regelungswerte über serielle Schnittstelle versenden
    - Not-Halt auslösen

Software-Entwicklung
Wrapper-Code (Rennstreckendaten)

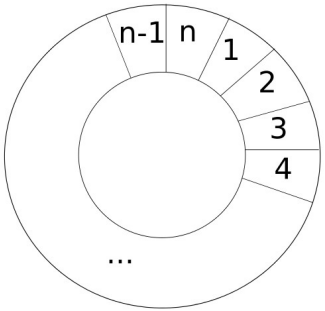
- Excel Datei
- 1 cm Rasterung der Rennstrecke
  - 237 Spalten
  - 237 Zeilen
- Inhalt der Zellen
  - 0: nicht befahrbar (rot)
  - 1: befahrbar (grün)
  - 2: Trajektorie (blau)



- Export als kommaseparierte CSV Datei
- In ein zwei-dimensionales Array eingelesen

Software-Entwicklung
Wrapper-Code (Empfang - Fahrzeugdaten)

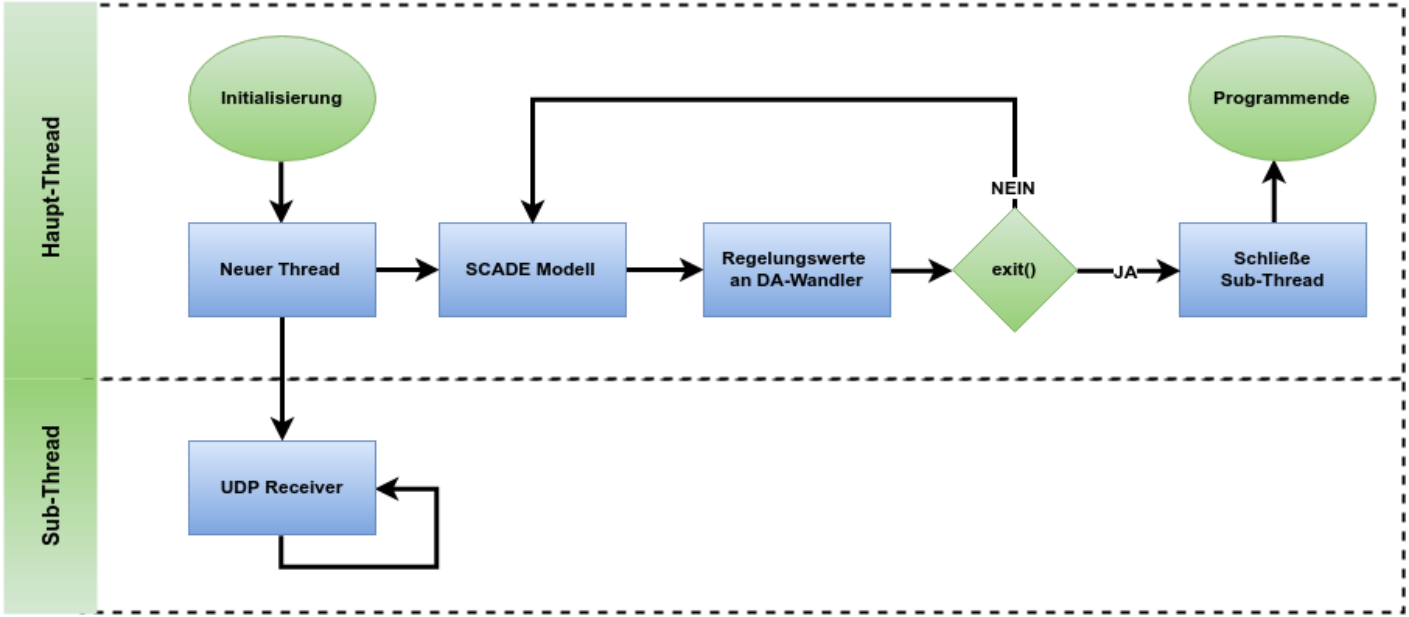
- Initialisierung eines UDP Servers
  - An Port 55555 gebunden
  - Broadcast IP „127.255.255.255“
- Funktion recvfrom() in while-Schleife
  - In eigenem Sub-Thread
    - pthread-Libary
  - Empfangene Fahrzeugdaten in globaler Variable bereitgestellt
    - Nutzung eines Ringpuffers



Software-Entwicklung
Wrapper-Code (Senden von Regelungsdaten)

- Serielle Schnittstelle initialisieren
  - Port öffnen
  - Einstellungen vornehmen
- Zu sendende Datenpakete
  - Regelungswert Längsführung
  - Regelungswert Querführung
- Nachricht (Byte-Folge) auf serielle Schnittstelle schreiben
  - Konvertierung der Regelungswerte (int zu byte)

Software-Entwicklung
Wrapper-Code (Programmablauf)

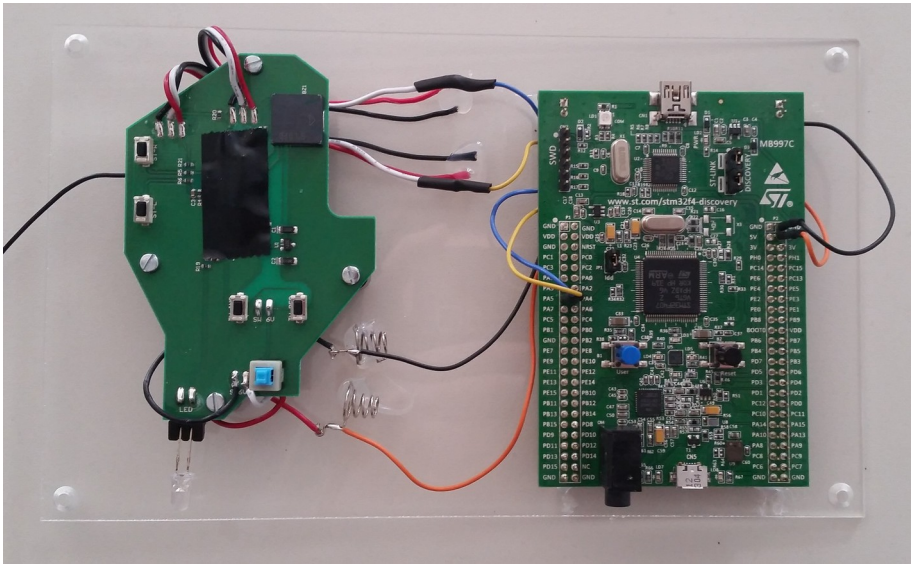


Software-Entwicklung

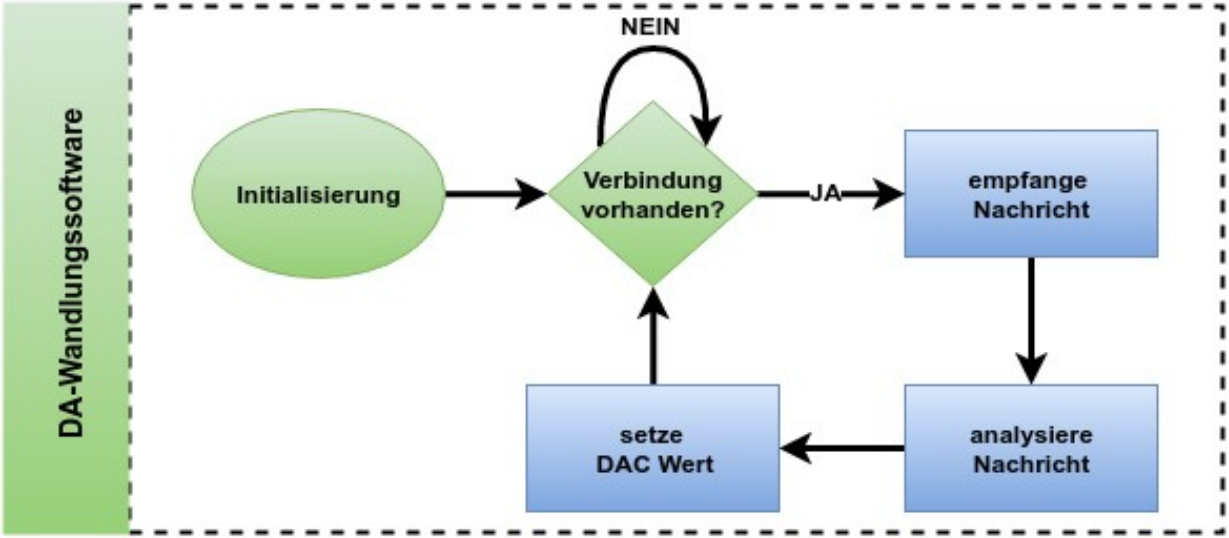
- Positionsbestimmung
- Kommunikation
- Control-Unit
  - Wrapper Code
  - DA-Wandlungssoftware
  - SCADE Modell
- Systemsteuerungssoftware

Software-Entwicklung
DA-Wandlungssoftware

- Wird auf DA-Wandler betrieben
  - Entwicklungsboard und umgebaute Fernsteuerung



Software-Entwicklung
DA-Wandlungssoftware (Programmablauf)

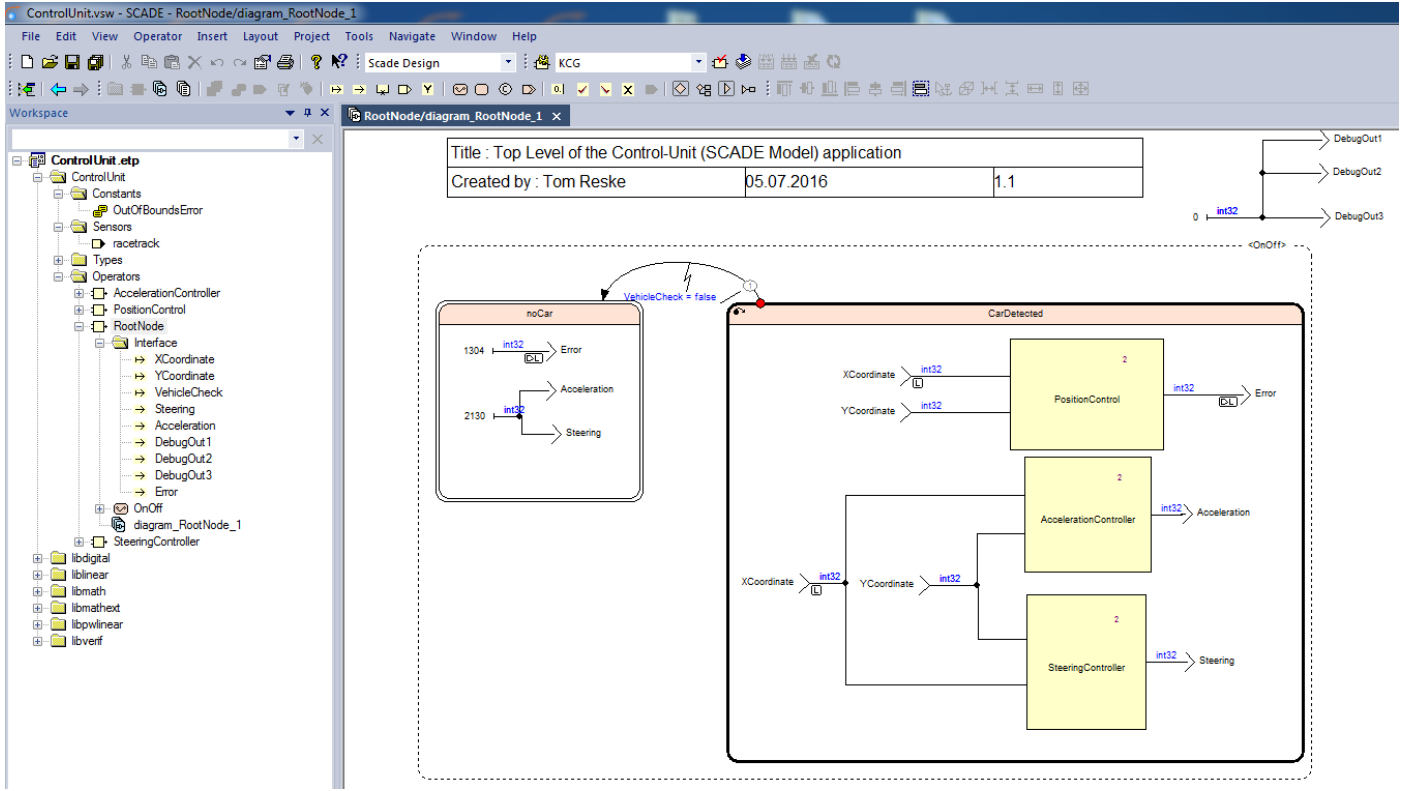


Realtime Controlled Cooperative Autonomous Racing System

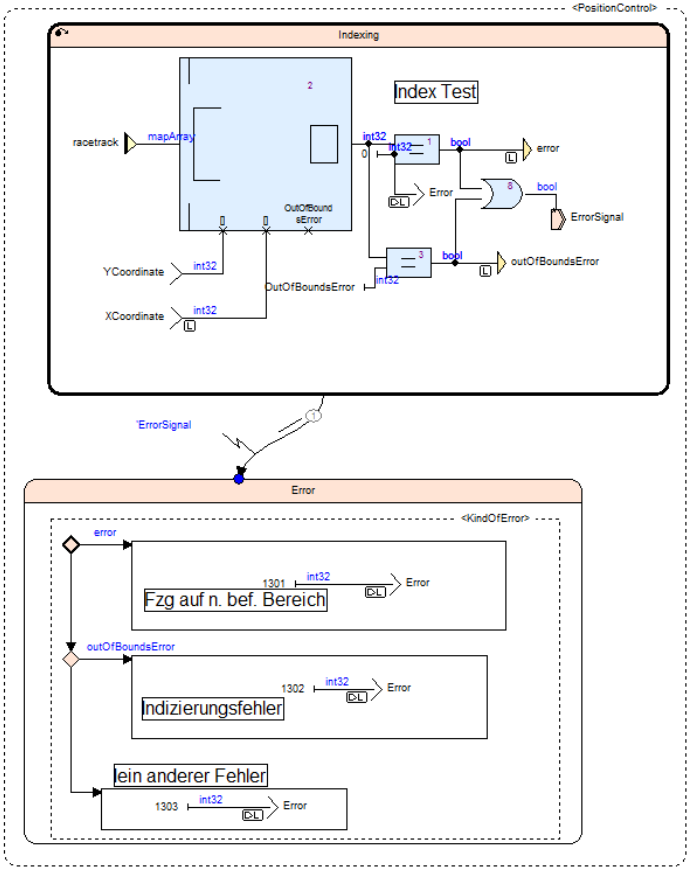
Software-Entwicklung

- Positionsbestimmung
- Kommunikation
- Control-Unit
  - Wrapper Code
  - DA-Wandlungssoftware
  - SCADE Modell
- Systemsteuerungssoftware

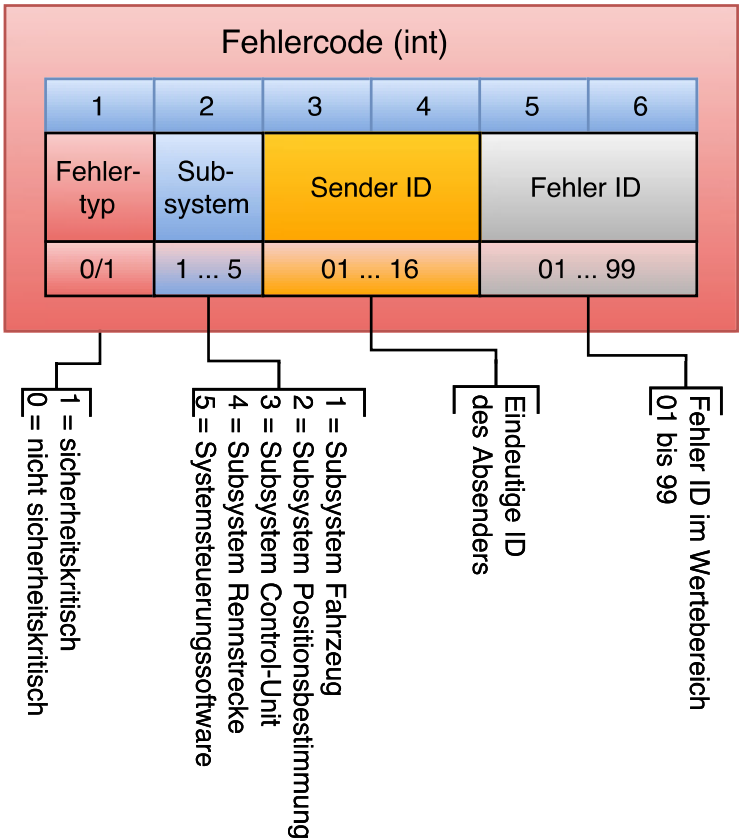
Software-Entwicklung
SCADE Modell – RootNode Operator



Software-Entwicklung
SCADE Modell – PositionControl Operator



Software-Entwicklung
Fehlercodes



Software-Entwicklung

- Positionsbestimmung
- Kommunikation
- Control-Unit
  - Wrapper Code
  - DA-Wandlungssoftware
  - SCADE Modell
- Systemsteuerungssoftware

Software-Entwicklung
Systemsteuerungssoftware

- **Entwicklung folgt** ab Sprint 3 (nach 2tem Review)
- Systeme werden zur Zeit vom **Administrator gestartet** und **gestoppt**

Coming soon...

Agenda

- Vorstellung
- Einleitung
- Projektorganisation
- Systemarchitektur und Schnittstellen
- Sicherheitskonzept
- Hardware-Evaluation
- Software-Entwicklung
- Testen
- Angepasste Anforderungen
- Aktueller Stand
- Ausblick
- Live-Demo

Testen
Agenda

- Testarten
- Testautomatisierung
- Testdokumentation
- Statisches Testen
- Testorganisation

Testen
Testarten

- Blackboxtests
  - Tests ohne Kenntnisse über innere Funktionsweise des Systems / der Systemkomponenten
  - Definition von Testfällen nicht gemäß der Implementierung sondern anhand der Spezifikation
- Whiteboxtests
  - Durchführung von Softwaretests mit Kenntnissen über innere Funktionsweise / Implementierung der Systemkomponenten
  - Überprüfung des Quellcodes
  - Überdeckungskriterien

Testen
Testarten

- Äquivalenzklassentests
  - Wertebereich der zu testenden Variablen oftmals zu groß
  - deshalb vor allem Grenzwerte testen
    - Beispiel:
      [-2147483648, -51], [-50, -1], [0], [1, 50], [51, 2147483647]
- Performanztests
  - testen ob Realzeitanforderungen erfüllt werden

Testen
Testarten

- Im Projekt RCCARS verwendete Testarten
  - Blackboxtests
    - Subsystemtests, Integrationstests, Systemabnahmetest
  - Whiteboxtests
    - vollständige Pfadüberdeckung allgemein unrealistisch
    - Kanten-, Bedingungsüberdeckung ebenfalls nicht machbar
    - → Äquivalenzklassentests zur Anweisungsüberdeckung
  - Performanztests
    - für die Anforderungen HW5.1.2, SW1.4, SW2.3, SW3.6

Testautomatisierung

- Motivation
  - Effizienz des Testvorgangs steigern
- Verwendung eines Testframeworks
  - Durchführung von Unit-Tests
    - Ausführung von Whiteboxtests automatisieren
    - effiziente Ausführung von Regressionstests

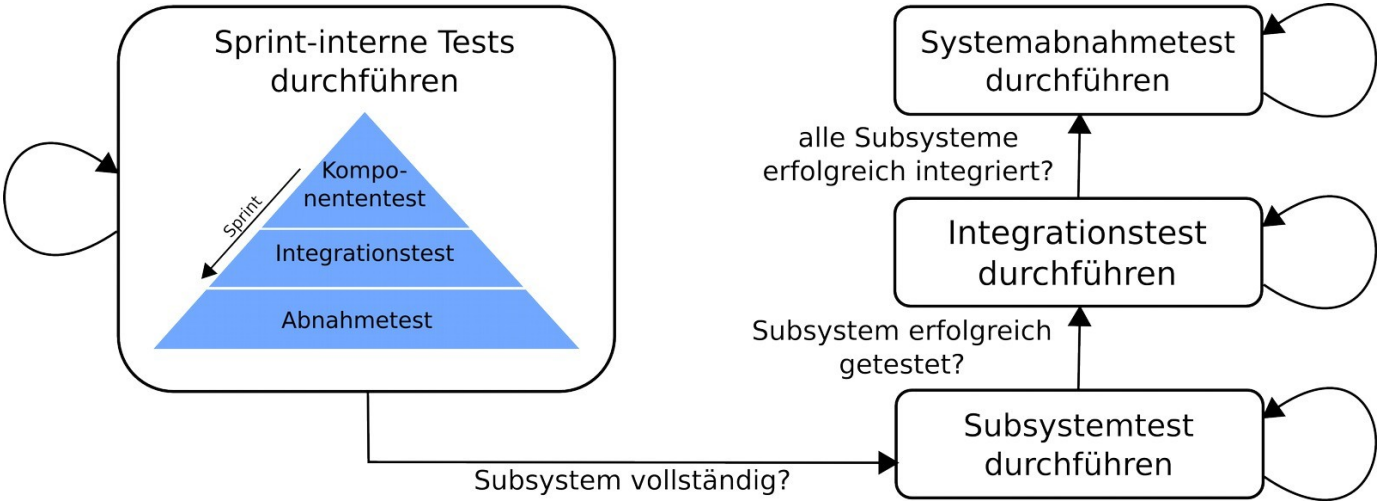
Unit-Testing im Projekt RCCARS

- Verwendung des Testframeworks *CUTE*
  - als Eclipse-Plugin in Entwicklungsumgebung integriert
  - automatisierte Erzeugung von Testsuits
  - zu testende Klassen / Structs lassen sich als Bibliotheken einbinden
  - zum Testen von C++/C-Code geeignet

Testen
Testorganisation

- Sprint-interne Tests
- Subsystemtests
- Integrationstests
- Systemabnahmetest

Testen
Testorganisation



Realtime Controlled Cooperative Autonomous Racing System

Testdokumentation

- Tests sorgfältig und fristgerecht durchführen
- strukturierte Definition und Dokumentation von Testfällen und deren Ergebnissen
  - fördert Vollständigkeit und Reproduzierbarkeit
- Testbericht muss alle Testergebnisse in strukturierter Form dokumentieren
  - auch Fehlschläge und Ergebnisse vorheriger Testläufe

Testdatenbank

- Verwendung des Setups der Projektgruppe *MIPSwarm*
- Vorgehen bei der Testfalldefinition:
  - grundlegende Testfälle und -Szenarien aus der Anforderungsdefinition bilden die Basis
  - Verfeinerung durch anforderungsspezifische Tests der jeweiligen Sprints
  - muss alle für die Reproduzierbarkeit relevanten Informationen beinhalten

https://knuth.informatik.uni-oldenburg.de/uni/ses/pg/rccars/index.php

Testen
Testdokumentation

Menü

Testfälle

Berichte

Testbericht (LaTeX)
Testbericht (PDF)
Download als ZIP

Statusübersicht

Typ	#	✓	!
Gesamt	66	30	
Systemabnahmetest	2	1	0
Subsystemtest	6	0	0
Integrationstest	2	0	0
Komponententest	28	12	0
Abnahmetest (Sprint)	0	0	0
Integrationstest (Sprint)	3	1	0
Komponententest (Sprint)	25	16	0

Legende

	Ergebnisse anzeigen
✓	Gültiges Ergebnis vorhanden
!	Kein gültiges Ergebnis vorhanden
✗	Kein Testergebnis vorhanden
	Eintrag bearbeiten
	Neuen Untertestfall anlegen
	Neues Testergebnis eintragen
	Testfall rekursiv löschen
	Version erhöhen

uID	ID	Bezeichnung	Testtyp	Version	Verantwortlich	Deadline	Ergebnisse	Neuen Testfall anlegen
26	GT-1	Positionsbestimmung getestet	Subsystemtest	0.2	Nikolai Braeuer	2016-09-19	0 ✗	
8	B01_T	➥ Kamerabild einlesen	Komponententest	0.2	Anatolij Fandrich	2016-06-23	1 ✓	
9	B02_T	➥ Fahrzeugpose bestimmen	Komponententest	0.2	Anatolij Fandrich	2016-06-23	1 ✓	
34	SW1.2.1	➥ Kamerabilder nachbearbeiten	Komponententest (Sprint)	0.6	Anatolij Fandrich	2016-04-30	1 ✓	
35	SW1.2.2	➥ Fahrzeuge erfassen	Komponententest (Sprint)	0.6	Anatolij Fandrich	2016-04-30	1 ✓	
37	SW1.2.3	➥ Fahrzeugkoordinaten ermitteln	Komponententest (Sprint)	0.4	Anatolij Fandrich	2016-04-30	1 ✓	
38	SW1.2.4	➥ Ausrichtung feststellen	Komponententest (Sprint)	0.4	Anatolij Fandrich	2016-04-30	2 ✓	
39	SW1.2.5	➥ Identität festellen	Komponententest (Sprint)	0.4	Anatolij Fandrich	2016-10-19	0 ✗	
84	SW1.4	➥ Realzeitanforderungen einhalten	Komponententest (Sprint)	0.3	Anatolij Fandrich	2016-10-20	0 ✗	
85	SW1.5	➥ Rennstreckenpose erfassen	Komponententest (Sprint)	0.1	Anatolij Fandrich	2016-06-23	1 ✓	
86	SW1.6	➥ Störungen erkennen	Komponententest (Sprint)	0.2	Nikolai Braeuer	2016-10-20	0 ✗	
87	SW1.6.1	➥ Verschiebung der Rennstrecke erkennen	Komponententest (Sprint)	0.4	Nikolai Braeuer	2016-06-23	2 ✓	
88	SW1.6.2	➥ Störung melden	Komponententest (Sprint)	0.2	Nikolai Braeuer	2016-10-20	0 ✗	
10	B03_T1	➥ Fahrzeugpose ausgeben	Komponententest	0.2	Anatolij Fandrich	2016-06-23	1 ✓	

Testen
Testfalldefinition

Test-Case Nr: SW1.6.1
Identifier für den Testfall

Name: Verschiebung der Rennstrecke erkennen
Aussagekräftiger Name des Testfalls

Testtyp: Komponententest (Sprint)
Was ist es für ein Testtyp

UseCase: -
Welcher Usecase wird durch den Testfall abgedeckt?

Bezug zu:
Hat der Testfall bezug zu einem anderen UseCase?

Testobjekt: Positionsbestimmungssoftware
Welche Komponente, Obejekt wird getestet?

Verantwortlich: Nikolai Braeuer

Deadline: 2016-06-23
Deadline bis zur Fertigstellung!
Angabe: YYYY-MM-TT

Vorbedingung:
Die Positionsbestimmungssoftware wurde erfolgreich gestartet und die Rennstrecke wird korrekt erfasst.
Es muss die Methode isTrackMoved() mit Hilfe des Unit-Tests isTrackMovedTest überprüft werden. Dabei werden Parametereingaben für identische Rennstreckenmarkerkoordinaten getestet und Eingaben für abweichende Koordinaten (sowohl oberhalb als auch unterhalb des Toleranzwertes.
Es muss außerdem empirisch getestet werden, ob das Bewegen der Rennstrecke bzw. der Kamera eine entsprechende Fehlermeldung auslöst, da hier auch das Zusammenspiel der Bildverarbeitung und der Erfassung der Rennstreckenpose relevant sind.

Vorbedingungen damit der Testfall ausgeführt werden kann.
In der Form <ID> : <Beschreibung> oder als Fließtext eintragen.

Auslösendes Ereignis: Die Rennstrecke oder die Kamera wird bewegt (mit Verschiebung größer als T
Auslösendes Ereignis zur Testdurchführung

Das Format für die AEKs lautet <AEK ID> : <Beschreibung>

AEKs: Die Positionsbestimmungssoftware gibt eine entsprechende Fehlermeldung aus.

Testen
Beispiel

2016-06-22 18:04:48	Nikolai Braeuer,	0.4	✓	Die Verschiebung wird entweder durch die Methode isTrackMoved() oder im Rahmen der Erfassung der Reflexionsmarker der Rennstrecke erkannt. Bei den emprischen Tests (Bewegen der Rennstrecke bzw. der Kamera) wurden die entsprechenden Fehlermeldungen ausgegeben: 1. FEHLER: Rennstrecke oder Kamera wurde bewegt! 2. FEHLER: Nicht alle Rennstrecken-Marker gefunden!	Kein Upload vorhanden	
2016-06-22 18:03:46	Nikolai Braeuer,	0.4	✓	Ausgabe des Testframeworks (+ Testausgabe des ConfigFileReaders): #beginning PositionDetectionTestSuite 1 #starting isTrackMovedTest MAX_THRESHOLD_TYPE: 4 MAX_THRESHOLD_VALUE: 255 MAX_BLUR_K_VALUE: 19 DEFAULT_WINDOW_WIDTH: 1024 DEFAULT_WINDOW_HEIGHT: 768 MAX_ERODE_VALUE: 100 MAX_DILATE_VALUE: 100 defaultThresholdType: 0 defaultThresholdLowerBound: 0 defaultBlurKValue: 0 defaultLowerInRangeThresholdValue: 237 defaultUpperInRangeThresholdValue: 255 defaultBarThresholdType: 3 defaultBarThresholdLowerBound: 95 defaultBarBlurKValue: 1 defaultBarLowerInRangeThresholdValue: 255 defaultBarUpperInRangeThresholdValue: 255 defaultErodeValue: 1 defaultDilateValue: 10 defaultErodeValueLight: 1 defaultDilateValueLight: 1 MAX_NUM_OBJECTS: 16 MIN_OBJECT_AREA: 225 MAX_OBJECT_AREA: 873813 TRACKSIZE: 995 BORDER_SIZE: 20 MAX_TRACK_MOVEMENT_TOLLERANCE: 3 #success isTrackMovedTest OK #ending PositionDetectionTestSuite	Kein Upload vorhanden	

Testdatenbank

- Erweiterung und Anpassung für RCCARS
  - Anpassung der Testtypen
  - Format der Testfall-ID
  - übergeordnete Testfälle abhängig von Untertestfällen
  - sortierbare Testfalltabelle

SCADE® ACADEMIC

Simulation mit SCADE

- Testen der Regelungssoftware
- gezieltes Testen von Eingangsbelegungen der Operatoren des Modells
- Testen der Statemachines des Modells
  - gezieltes Herbeiführen von Zuständen
  - Simulation nach Korrekturen und Änderungen
  - unabhängiges Testen des Modells

Codereviews

- Begutachtung des Programmcodes durch ein anderes Gruppenmitglied
- Rückmeldungen über mögliche Fehlerquellen und Optimierungspotentiale
- auch Unit-Tests müssen durch Codereviews verifiziert werden

Agenda

- Vorstellung
- Einleitung
- Projektorganisation
- Systemarchitektur und Schnittstellen
- Sicherheitskonzept
- Hardware-Evaluation
- Software-Entwicklung
- Testen
- Angepasste Anforderungen
- Aktueller Stand
- Ausblick
- Live-Demo

Angepasste Anforderungen

Anpassungen der Anforderungsspezifikation

- Klärung von Missverständnis bzgl. der Rennstrecke
  - **SW2.1.4** - Die Regelungssoftware muss den **Verlauf der Rennstrecke** und die vom Fahrzeug zu fahrende Route **als Eingabeparameter** empfangen.
- Anpassungen bzgl. der Erweiterbarkeit in Folgeprojekten
  - **Sys1** - Das System muss fähig sein, **mindestens ein Fahrzeug** autonom über eine vorgegebene Rennstrecke zu steuern.
  - **SW1** - Die Software des Subsystems 2 muss fähig sein, die Pose der **Fahrzeuge** zu bestimmen.

Angepasste Anforderungen

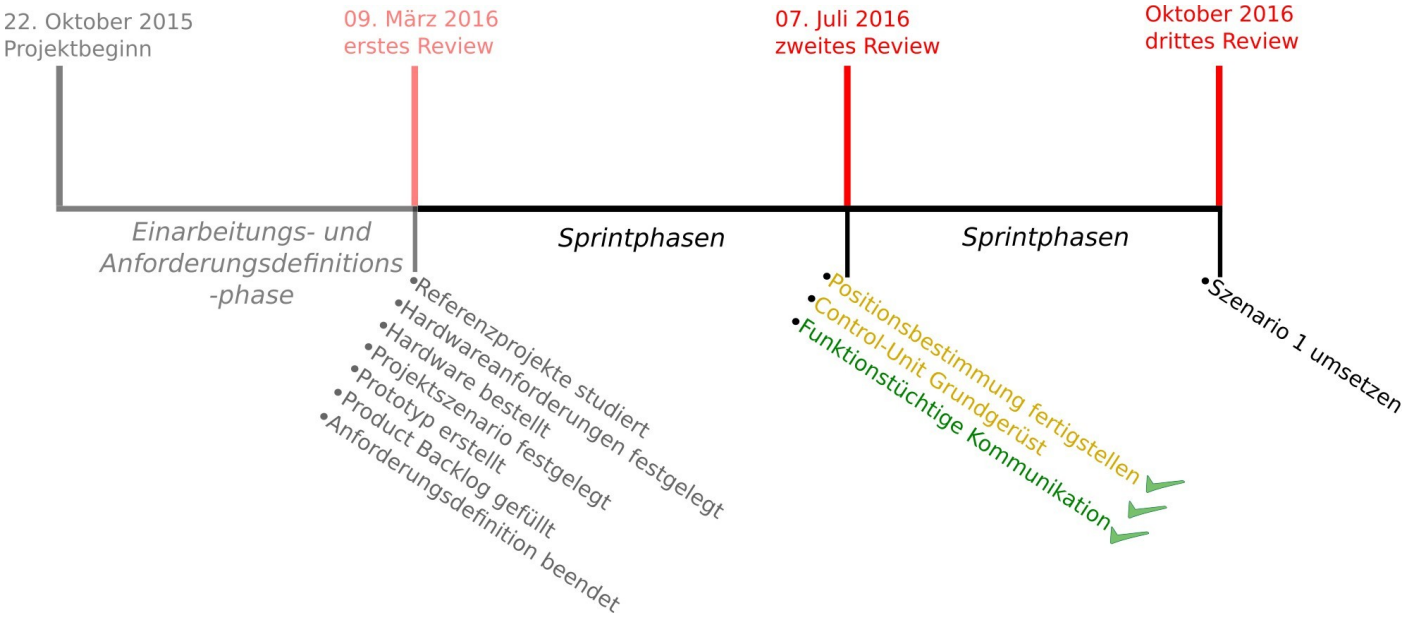
- Anforderungen bzgl. der Fahrzeugphysik
  - **SW2.1.7** - Das Fahrzeug darf die **Kurvengeschwindigkeit** von 1,2 m/s nicht überschreiten.

- ergänzte Rahmenbedingung
  - **Rah1.3** - Die Minimum Bounding Box der Rennstrecke darf während des Systembetriebs nicht direkter **Sonneneinstrahlung** ausgesetzt sein.

Agenda

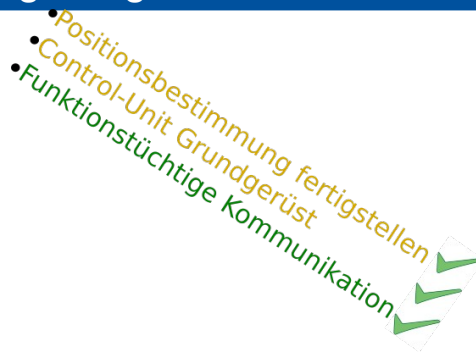
- Vorstellung
- Einleitung
- Projektorganisation
- Systemarchitektur und Schnittstellen
- Sicherheitskonzept
- Hardware-Evaluation
- Software-Entwicklung
- Testen
- Angepasste Anforderungen
- Aktueller Stand
- Ausblick
- Live-Demo

Aktueller Stand



Realtime Controlled Cooperative Autonomous Racing System

Aktueller Stand
Ziel 1 – Positionsbestimmung fertigstellen



- Erreicht:
  - Bilder einlesen
  - Bilder verarbeiten
  - Fahrzeug detektieren
  - Fahrzeugpose bestimmen
  - Rennstrecke erfassen
  - Störungen erkennen (bspw. Fahrzeug verloren)
- Einschränkungen
  - Performance muss optimiert werden
  - Robustere Rennstreckendetektierung (Marker)
  - Erweiterbarkeit (Mustererkennung)

RCCARS
Realtime Controlled Cooperative Autonomous Racing System

Aktueller Stand
Ziel 2 – Control-Unit Grundgerüst

- Positionsbestimmung fertigstellen
- Control-Unit Grundgerüst
- Funktionstüchtige Kommunikation

- Erreicht:
  - Daten können angenommen werden
  - Rennstrecke inkl. Zieltrajektorie visualisiert
  - Rennstrecke wird geladen
  - Kollisionserkennung
  - Erkennung über Verlassen des befahrbaren Bereichs
  - Fehlererzeugung und -behandlung
  - Senden an DA-Wandler / allgemeine Fahrzeugsteuerung
  - *Watchdog Ansatz*
- Einschränkungen:
  - Plausibilitätsüberprüfung
  - Fahrtrichtungserkennung

Realtime Controlled Cooperative Autonomous Racing System

Aktueller Stand
Ziel 3 – Funktionstüchtige Kommunikation

- Positionsbestimmung fertigstellen
- Control-Unit Grundgerüst
- Funktionstüchtige Kommunikation

- Erreicht:
  - Positionsbestimmung UDP Client
  - Positionsbestimmung sendet Datenpakete
  - Control-Unit UDP Server
  - Control-Unit empfängt Datenpakete
- Daraus ergibt sich:
  - Funktionierender Kommunikationskreislauf

Aktueller Stand
Probleme und Problembehandlungen

- Positionsbestimmung fertigstellen
- Control-Unit Grundgerüst
- Funktionstüchtige Kommunikation

- Zeitprobleme
  - Unvorhersehbare Verzögerungen in der Entwicklung (bspw. Mustererkennung)
  - Personalausfall
  - Personalmangel
- Prinzip der geteilten Ressourcen
- Flexibilität von ScrVm genutzt
  - Anforderungen zurückgeführt
  - Anforderungen erhoben

Realtime Controlled Cooperative Autonomous Racing System

Aktueller Stand
ScrVm Dokument

1 Einführung 1

2 Product Backlog 2
2.1 Funktionale Anforderungen . . . 2
2.1.1 Generelle Systemanforderungen . . . 2
2.1.2 Hardwareanforderungen . . . 6
2.1.3 Softwareanforderungen . . . 14
2.2 Nicht funktionale Anforderungen . . . 20
2.2.1 Hardwareanforderungen . . . 20
2.2.2 Softwareanforderungen . . . 21

3 Höher priorisierte Anforderungen 22
3.1 Anforderung 1 . . . 22
3.2 Anforderung 2 . . . 22
3.3 Anforderung 3 . . . 22

4 Sprints 24
4.1 Sprint 1.1 . . . 24
4.1.1 Ausführendes Team . . . 24
4.1.2 Zeitraum . . . 24
4.1.3 Sprint Backlog . . . 24
4.1.4 Testfälle . . . 25
4.1.5 Nicht erfüllte Anforderungen . . . 25
4.1.6 Neu aufgenommene Anforderungen . . . 25
4.2 Sprint 1.2 . . . 26
4.2.1 Ausführendes Team . . . 26
4.2.2 Zeitraum . . . 26
4.2.3 Sprint Backlog . . . 26
4.2.4 Testfälle . . . 27
4.2.5 Nicht erfüllte Anforderungen . . . 27
4.2.6 Neu aufgenommene Anforderungen . . . 27
4.3 Sprint 2.1 . . . 28
4.3.1 Ausführendes Team . . . 28
4.3.2 Zeitraum . . . 28
4.3.3 Sprint Backlog . . . 28
4.3.4 Testfälle . . . 28
4.3.5 Nicht erfüllte Anforderungen . . . 29
4.3.6 Neu aufgenommene Anforderungen . . . 29
4.4 Sprint 2.2 . . . 30
4.4.1 Ausführendes Team . . . 30
4.4.2 Zeitraum . . . 30
4.4.3 Sprint Backlog . . . 30
4.4.4 Testfälle . . . 31
4.4.5 Nicht erfüllte Anforderungen . . . 31
4.4.6 Neu aufgenommene Anforderungen . . . 31

5 Erfüllte Anforderungen 32

Literaturverzeichnis 35

- ScrVm Prozess verfolgen
- Übersicht über Projektfortschritt
- Anforderungsmanagement
- Priorisierte Anforderungen
  - Grund
  - Status
- Eigenständiges Dokument
- Hoher Nutzenfaktor
- Empfehlenswert!

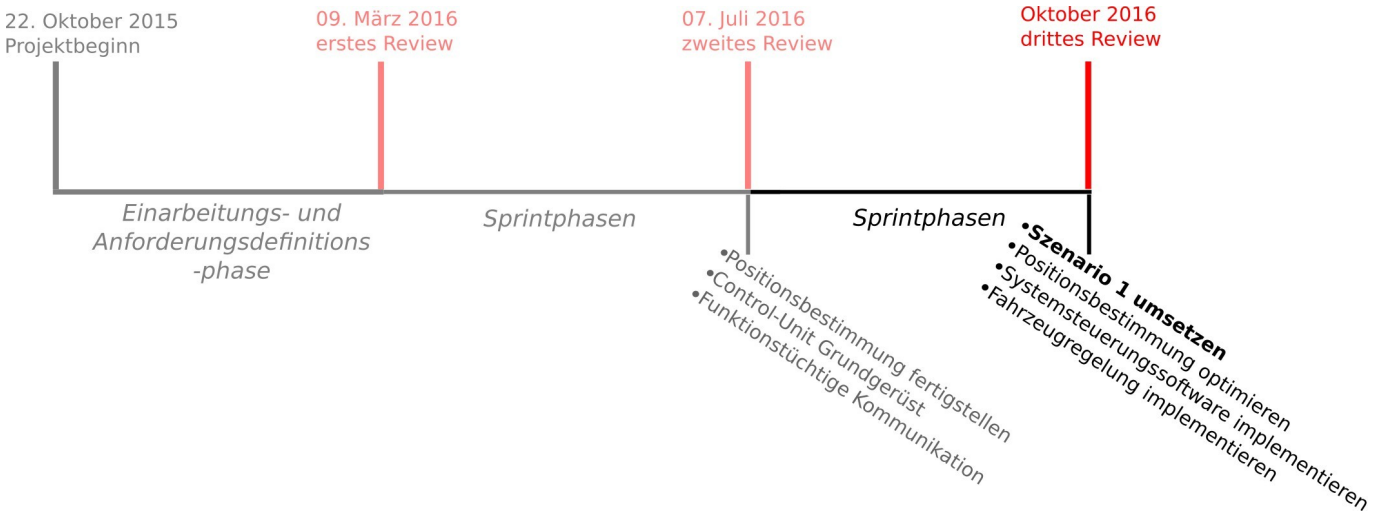
RCCARS
Realtime Controlled Cooperative Autonomous Racing System

Agenda

- Vorstellung
- Einleitung
- Projektorganisation
- Systemarchitektur und Schnittstellen
- Sicherheitskonzept
- Hardware-Evaluation
- Software-Entwicklung
- Testen
- Angepasste Anforderungen
- Aktueller Stand
- Ausblick
- Live-Demo

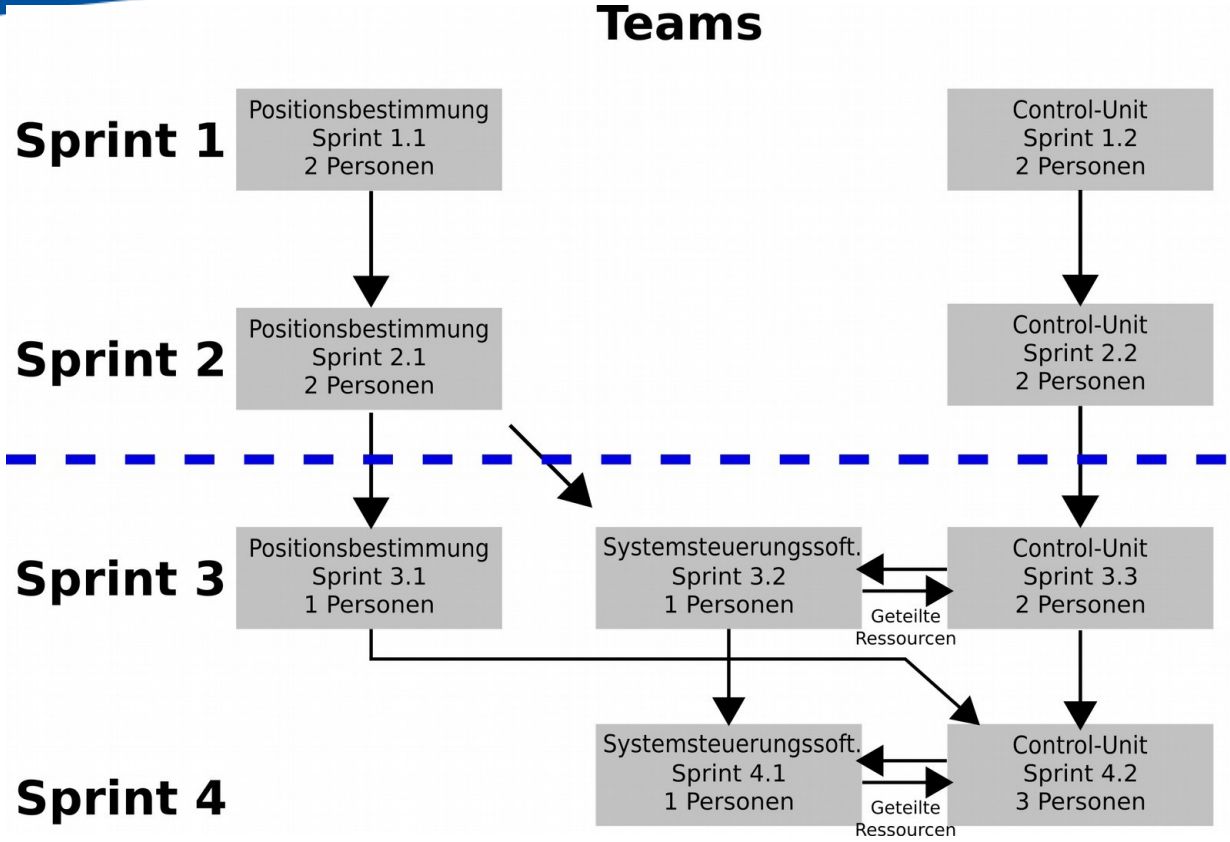
Realtime Controlled Cooperative Autonomous Racing System

Ausblick
Ziele

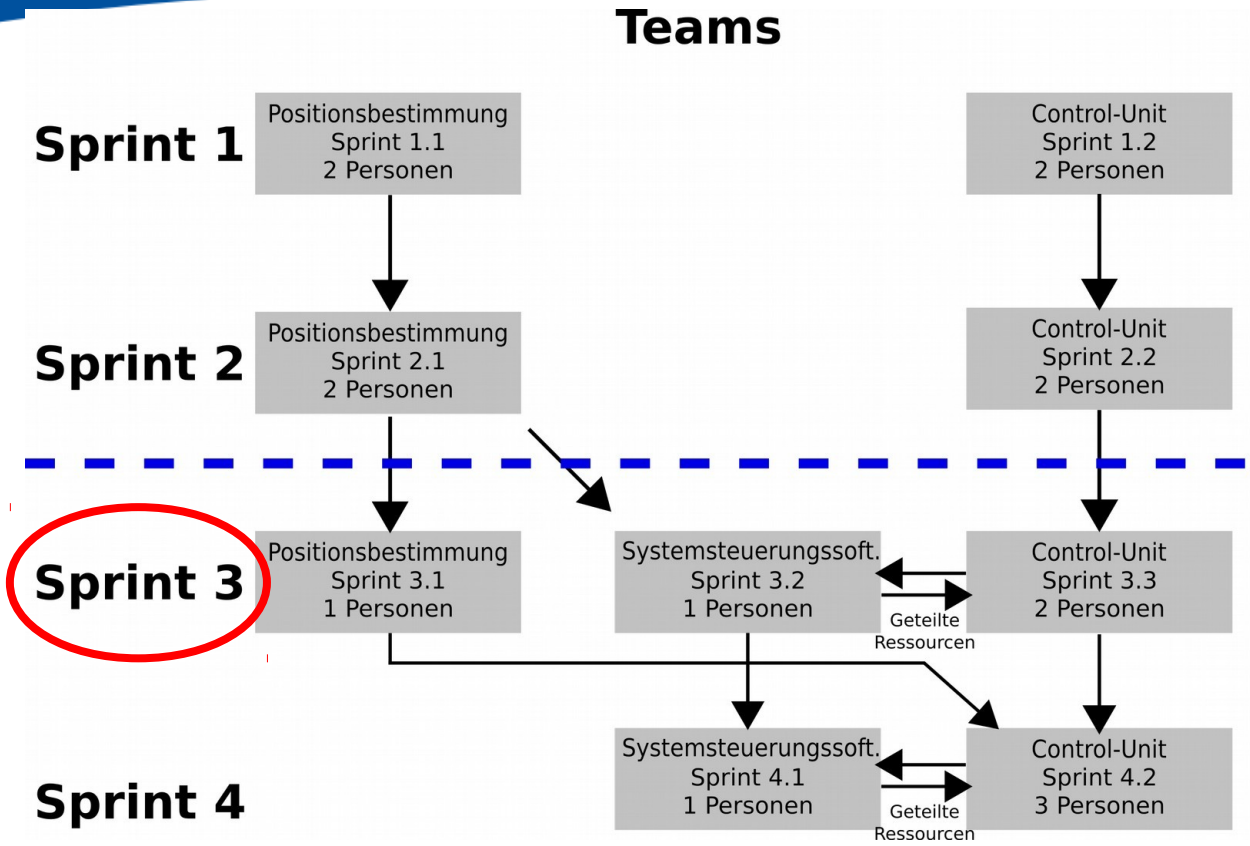


Realtime Controlled Cooperative Autonomous Racing System

Ausblick
Sprintplanung



Ausblick
Sprintplanung



Ausblick

Sprintplanung – Sprint 3 – 7 Wochen

Positionsbestimmung

- SW1.2.5 […] Identität […] feststellen.
- SW1.4 […] 10ms […].

Control-Unit

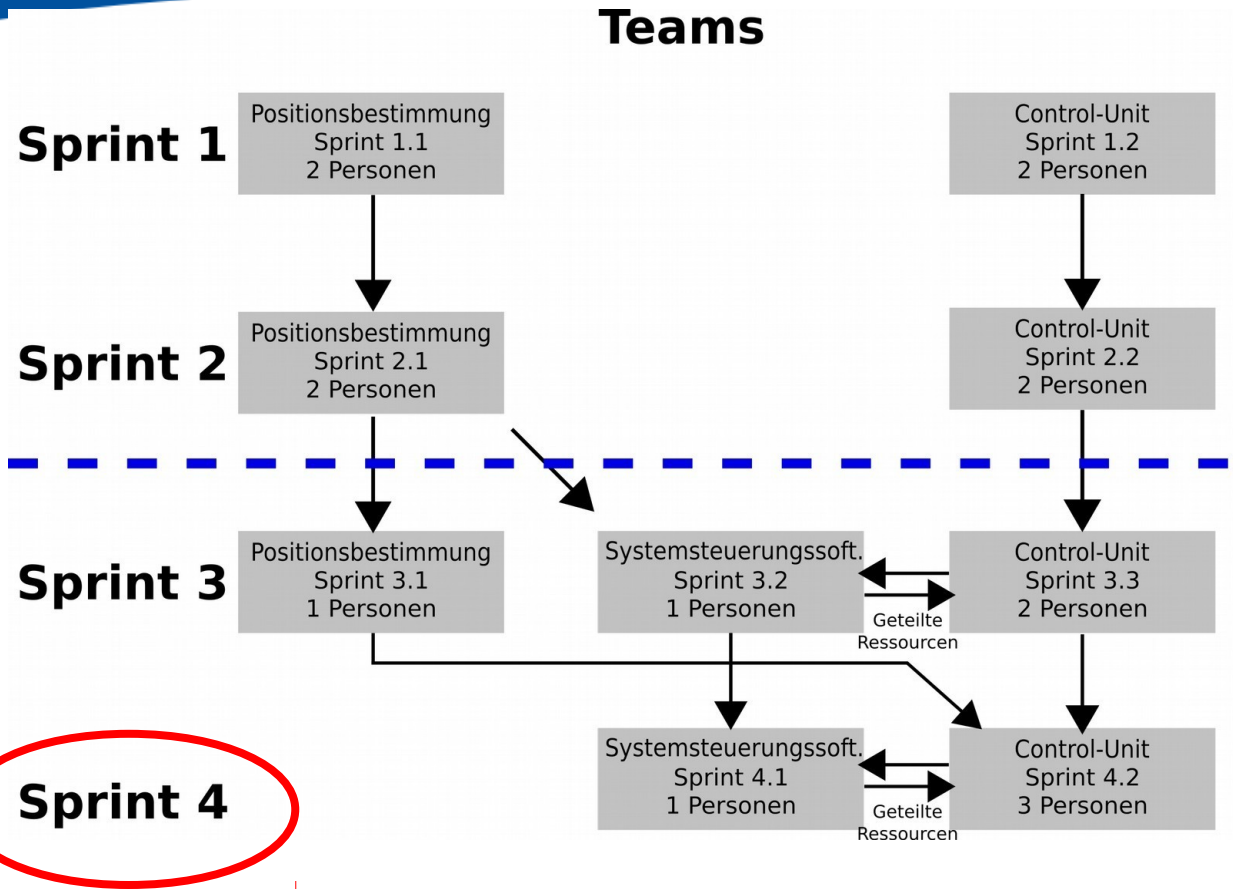
- SW2.1.8.3 […] richtige Richtung […].
- SW2.1.8.9 […] Plausibilitätsüberprüfung […]
- SW2.1.8.9 […] Störung Systemsteuerungssoftware […].
- SW2.4 […] 10ms […] .

Ausblick
Sprintplanung – Sprint 3 – 7 Wochen

Systemsteuerungssoftware

- SW 3.1 […] Systemkomponenten […] starten.
- SW3.2.1 […] Benutzerinterface […].
- SW3.2.2 […] Initialisierung […] starten.
- SW3.2.3 […] Rennbetrieb […] starten.
- SW3.2.4 […] Rennbetrieb […] stoppen.
- ...

Ausblick
Sprintplanung



Ausblick
Sprintplanung – Sprint 4 – 7 Wochen

Systemsteuerungssoftware

- SW3.3 […] Fehlermeldungen […] empfangen.
- SW3.4 […] System in sicheren Fehlerzustand.
- SW3.5 […] Fehler […] Fahrzeuge sofort stoppen.
- ...

Control-Unit

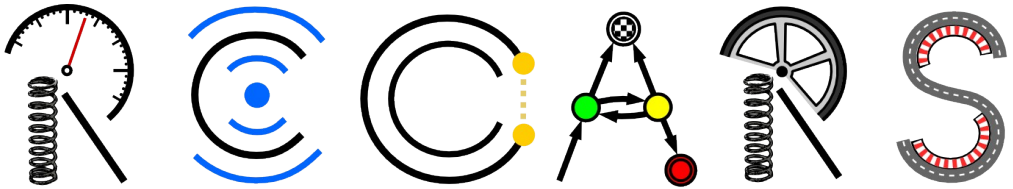
- SW2.1 […] Regelungs-software für die autonome Steuerung […].

Agenda

- Vorstellung
- Einleitung
- Projektorganisation
- Systemarchitektur und Schnittstellen
- Sicherheitskonzept
- Hardware-Evaluation
- Software-Entwicklung
- Testen
- Angepasste Anforderungen
- Aktueller Stand
- Ausblick
- Live-Demo

Live Demonstration

Live Demo



Realtime Controlled Cooperative Autonomous Racing System

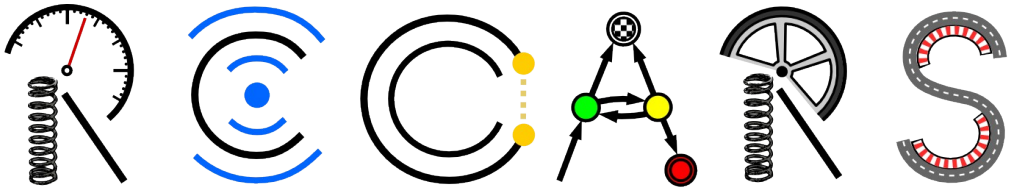
Fazit

- ScrVm größtenteils gelebt
- Überblick über das Projekt & Anforderungen behalten
- Bisher gesetzte Ziele (mit Einschränkungen) erreicht
- Praktischer Zeitaufwand höher als theoretischer Zeitaufw.
- Hoher Arbeitsaufwand
- Wenige Arbeitsressourcen
- Stets hilfsbereite Betreuer
- Kritische aber **optimistische** Betrachtung das gesetzte Ziele zu erreichen

Realtime Controlled Cooperative Autonomous Racing System

Ende

Vielen Dank für die Aufmerksamkeit!



Realtime Controlled Cooperative Autonomous Racing System