

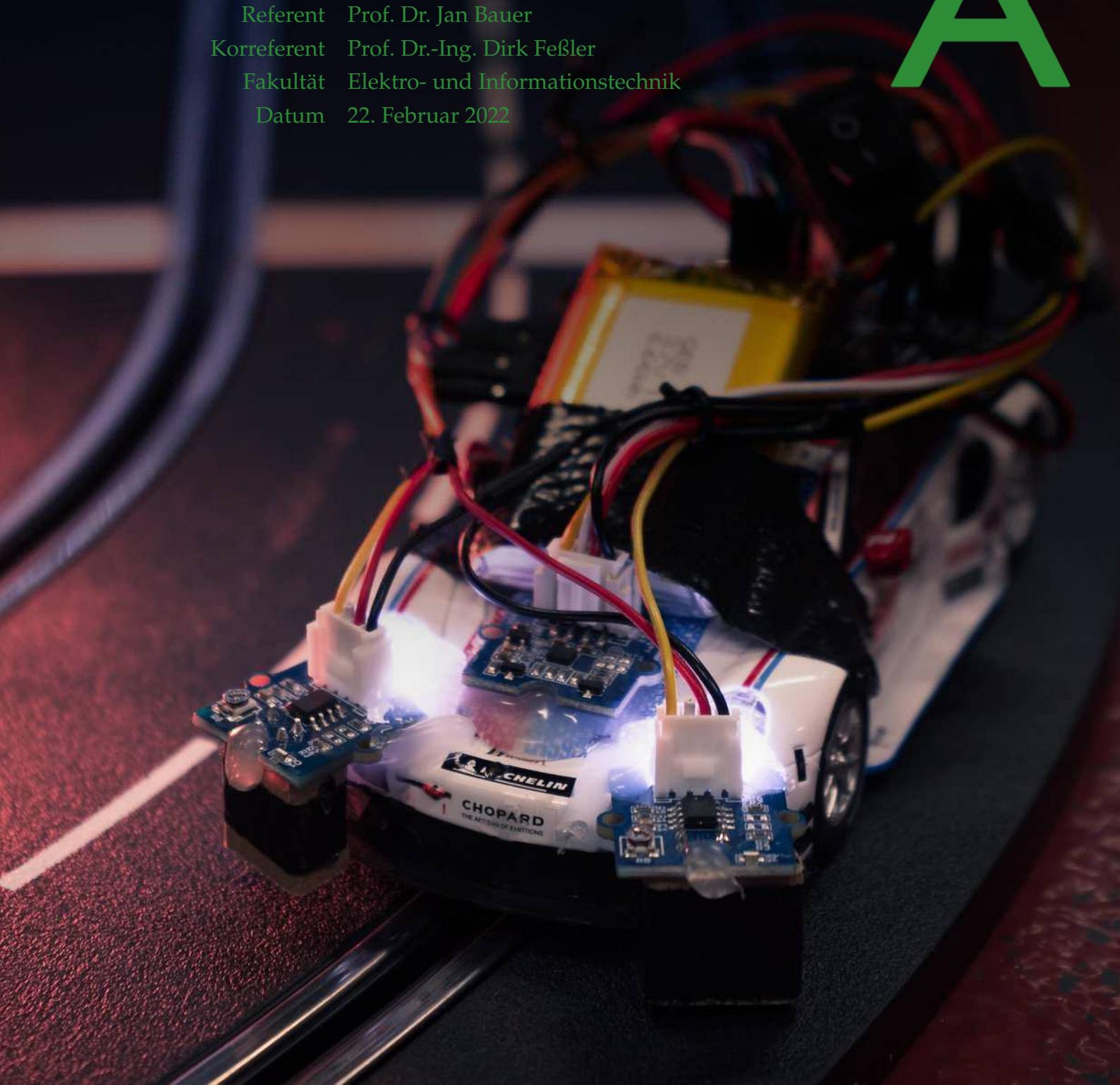
EVA - Electric slot Vehicle Automation

H
I
K
A

METHODEN UND HERAUSFORDERUNGEN
BEI DER AUTOMATISIERUNG VON CARRERA
DIGITAL 124/132 AUTORENNBAHNEN

Thesis Bachelor of Engineering

Autor Benedikt Reinberger-Eyer
Referent Prof. Dr. Jan Bauer
Korreferent Prof. Dr.-Ing. Dirk Feßler
Fakultät Elektro- und Informationstechnik
Datum 22. Februar 2022



Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne unzulässige fremde Hilfe selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben.

Karlsruhe, 22. Februar 2022

Benedikt Reinberger-Eyer

Abstract (english version)

Slot car racing enjoys great popularity among casual and competitive players, even sparking national championships. Even though the use of slot car tracks is prominent, it has some drawbacks. If individual time trial is not the preferred mode of operation, playing it requires two players.

As a solution to this hindrance and a means to connect engineering to the everyday excitement of slot car racing, project EVA was launched. It provides an automated car to race against. EVA is a backronym for „Electric slot Vehicle Automation“ . A successful and performant implementation requires prior knowledge in several fields of engineering. Project EVA is a means to get a general idea of what slot car automation is all about. Close attention is paid to the intricacies and challenges during the implementation process. This is achieved developing and documenting an automation for a digital race track. The acquired knowledge can then be utilized when a new platform for slot vehicle automation is built.

The final implementation is handicapped by the choice of track: using a digital race track has some implications rarely covered by literature. Thus, resulting performance is rather poor compared to other projects using analog race tracks, apart from consistency.

Note that this document is penned in the german language. If the contents intrigue you, do not hesitate to translate it using one of the many free translation tools.

Abstract (deutsche Version)

Autorennbahnen sind ein populäres Spielzeug für den Hobbybetrieb sowie bei nationalen Meisterschaften. Während das Spielen mit einer Autorennbahn beliebt ist, hat es leider einen grundlegenden Nachteil: Wenn nicht nur gegen die eigene Zeit gefahren werden soll, wird eine Konkurrenz benötigt, gegen die die eigenen Fähigkeiten gemessen werden können. Um diesem Problem entgegenzuwirken und zusätzlich die Brücke von der Ingenieursdisziplin zum alltäglichen Spaß mit einer Rennbahn zu schlagen, wurde das EVA-Projekt gestartet. Diese implementiert ein autonom fahrendes Auto als Rennopposition. EVA ist ein Backronym und steht für „Electric slot Vehicle Automation“.

Um eine erfolgreiche und besonders leistungsstarke Realisierung zu erwirken, ist es nötig, einen Überblick über die relevanten Bereiche der Ingenieursdisziplinen zu erhalten. Das Projekt EVA dient dieser Nische, indem eine Implementierung an einer digitalen Rennstrecke durchgeführt wird. Besonderes Augenmerk wird hierbei auf die Feinheiten und Herausforderungen bei der Ausgestaltung gelegt. Anschließend kann auf der Basis des erlernten Wissens eine neue Plattform zur Weiterentwicklung der Konzepte und Algorithmen entworfen und implementiert werden.

Die endgültige Implementierung weist durch die Wahl einer digitalen Rennstrecke verglichen mit der bestehenden Literatur einige neue Herausforderungen auf. Diese Einschränkungen führen zu einer vergleichsweise schwachen aber konsistenten Rennopposition.

Inhaltsverzeichnis

1 Motivation	5
1.1 Projektziele	6
1.2 Funktionskonzept und technische Ziele	7
2 Stand der Technik	8
3 Auswahl der Hardware	11
3.1 Wahl der Sensoren	11
3.2 Wahl der Rennbahn	14
4 Funktionsweise und Features der digitalen Rennbahn	15
4.1 Features	15
4.2 Carrera D124/132 Protokolle	16
4.2.1 Control Unit Datenprotokoll	16
4.2.2 CU-Rundenzähler-Protokoll	20
5 Schnittstelle mit der Strecke	22
6 Systemarchitektur	24
6.1 Aufbau des EVA-Systems	24
6.2 Integration der Sensoren	30
6.3 Algorithmus: Wie muss die Rennbahn angesteuert werden?	36
6.3.1 Streckenverlauf bestimmen	37
6.3.2 Segmentweise Position bestimmen	39
6.3.3 Geschwindigkeit mit Accelerometern bestimmen	40
6.3.4 Geschwindigkeit mit Reflexionssensoren bestimmen	43
6.3.5 Systemverhalten bestimmen: mathematische Modellbildung .	48
6.3.6 Algorithmen für die Ansteuerung der Rennstrecke	52
6.3.7 Simulation des Systems aus EVA und der Rennstrecke	56
6.3.8 Ergebnisse	60
7 Ausblick	61
8 Fazit	63
9 Quellen	64

1 Motivation

Autorennbahnen erfreuen sich seit ihrer Erfindung in den frühen 1910ern [1] bis heute einer großen Beliebtheit bei Jung und Alt.

Das Funktionsprinzip ist simpel: Ein kleines, mit einem Elektromotor ausgestattetes Auto fährt auf einer elektrifizierten Schiene. Die Spannung wird mithilfe von Kontaktschleifern von der Schiene in das Auto übertragen. Die Mitspielen können mithilfe eines Handreglers, manchmal auch „Drücker“ genannt, die Spannung auf der Schiene verändern, und somit die Geschwindigkeit des Autos anpassen.

Die meisten Streckensysteme verfügen über mindestens zwei Schienen, sodass zwei Leute Rennen gegeneinander fahren können. Ziel ist es, eine vorgegebene Rundenzahl vor der Opposition abzuschließen. Dazu muss das Auto möglichst schnell fahren - wenn das Auto jedoch zu schnell unterwegs ist, wird es in den Kurven von seiner eigenen Trägheit aus der Bahn geworfen und kostbare Zeit geht verloren.

Abbildung 1 zeigt ein Rennauto auf einer Rennbahn mit zwei Schienenpaaren. Vor dem Auto liegt ein Handregler. Wird der Schlegel mit der gelben Kappe in Richtung des roten Knopfes gedrückt, so wird das Auto schneller. Seine Funktionalität ist vergleichbar mit dem Gaspedal in einem herkömmlichen Kraftfahrzeug. Eine Bremse gibt es nicht - um das Auto zu verzögern, muss vom Gas gegangen werden. Das Konstrukt zur Rechten der Bahn ist für die Bestromung der Schienen zuständig. Die Einspeisung erfolgt hier nur an einer Stelle, anders als beispielsweise bei einer U-Bahn, die entlang der Strecke mehrere Einspeisungsstellen hat, um die Effekte des Leitungswiderstands zu verringern.

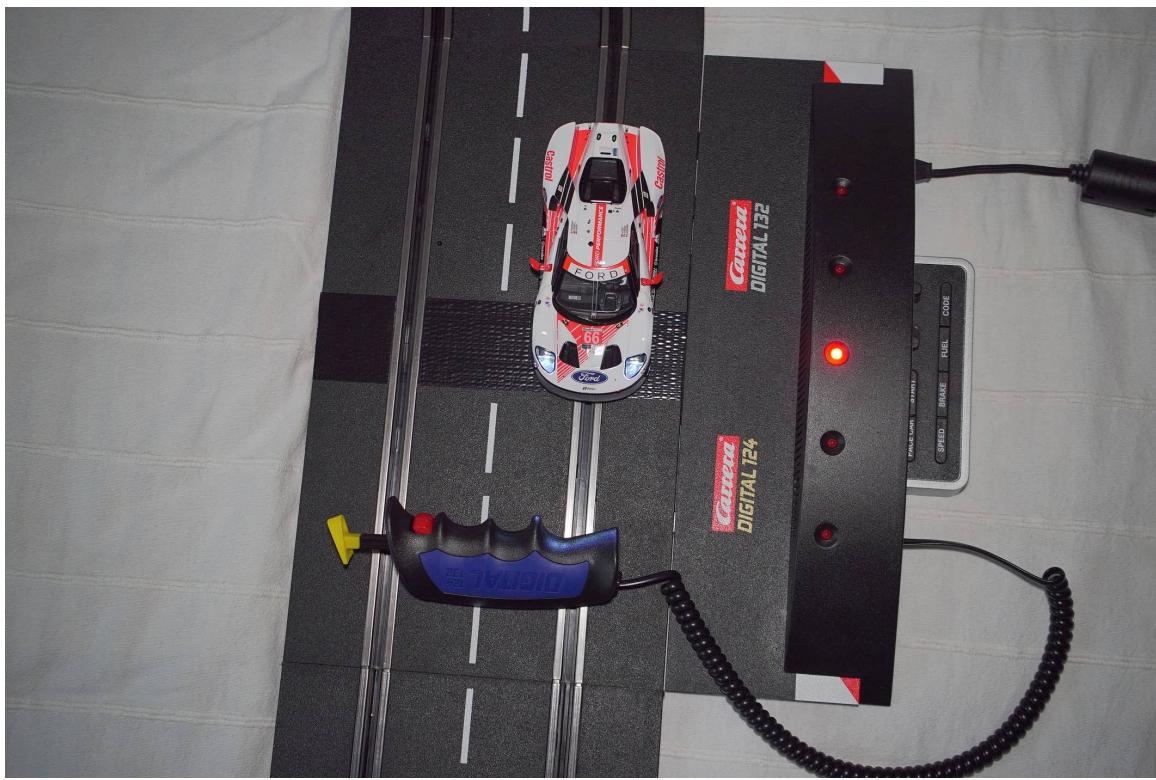


Abbildung 1: Rennauto, Spannungsversorgung und Handregler einer Carrera D132-Bahn

1.1 Projektziele

Wie bereits in Sektion 1 und im Abstract beschrieben, ist das Projektziel das Bereitstellen eines Mitspielenden.

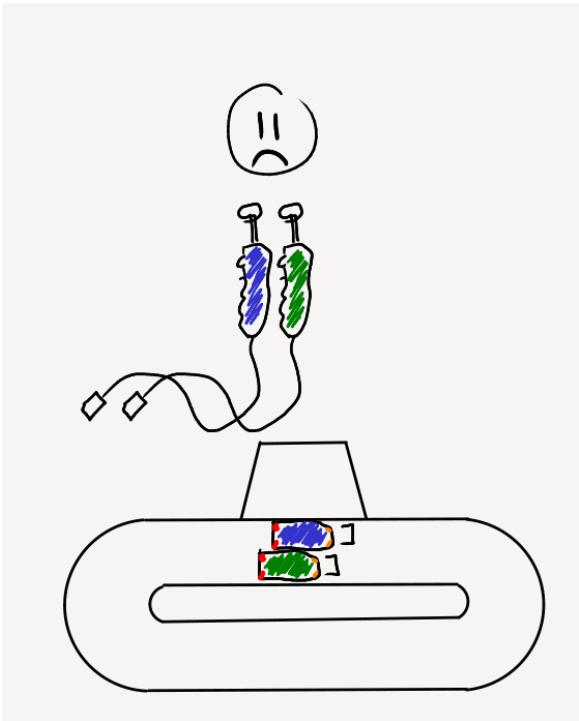


Abbildung 2: Mensch ohne Opposition

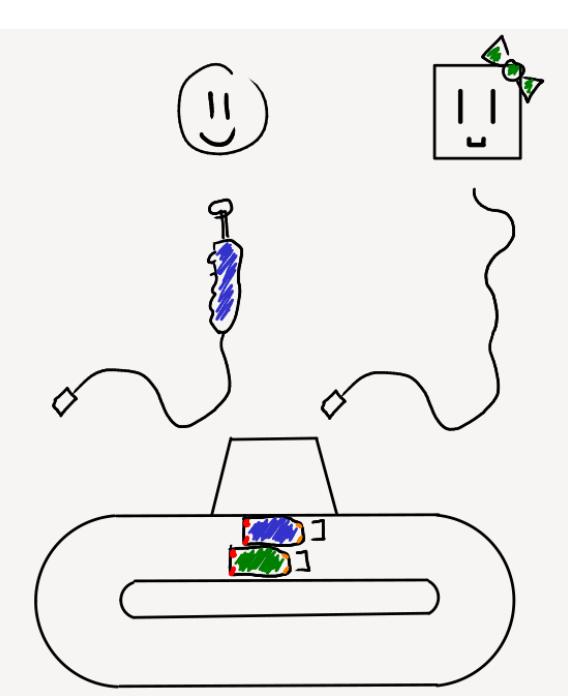


Abbildung 3: Mensch gegen EVA

EVA übernimmt also die Rolle der Konkurrenz. Dazu muss das Auto autonom gesteuert werden. Um die Gegenseite zu ersetzen, müssen die Ziele der Steuerung identisch zu denen eines Rennens zwischen zwei Menschen sein:

- Reduzieren der Rundenzeit
- Verhindern einer Entgleisung

Zusätzlich wird noch weitere Funktionalität benötigt - die Möglichkeit eines Rennens von Menschen gegen EVA und die Fähigkeit, sich auf eine Rundenzahl festzulegen.

1.2 Funktionskonzept und technische Ziele

Grundsätzlich ist für jede Steuerung oder Regelung wichtig:

- Was ist der aktuelle Zustand des zu regelnden Systems (Istwert)?
- Was soll der Zustand des Systems eigentlich sein (Sollwert)?

Um den aktuellen Zustand des Systems herauszufinden, müssen Sensoren eingesetzt werden. Um den Zielzustand zu bestimmen, muss ein Modell gebildet oder empirische Messungen durchgeführt werden. Um das System möglichst so

anzusteuern, dass der Zielzustand optimal erreicht wird, wird ein Algorithmus benötigt.

In Abbildung 4 ist ein grobes Blockschema augezeichnet, das diese Funktionalität aufweist. Das Modell ist hier im Entwurf des Algorithmus' bereits enthalten. Das zu regelnde System ist die Rennbahn. Von dieser können mithilfe von Sensoren physikalische Größen des Autos erhoben werden. Um das Auto zu steuern, muss EVA mit den Teilsystemen der Rennbahn interagieren.

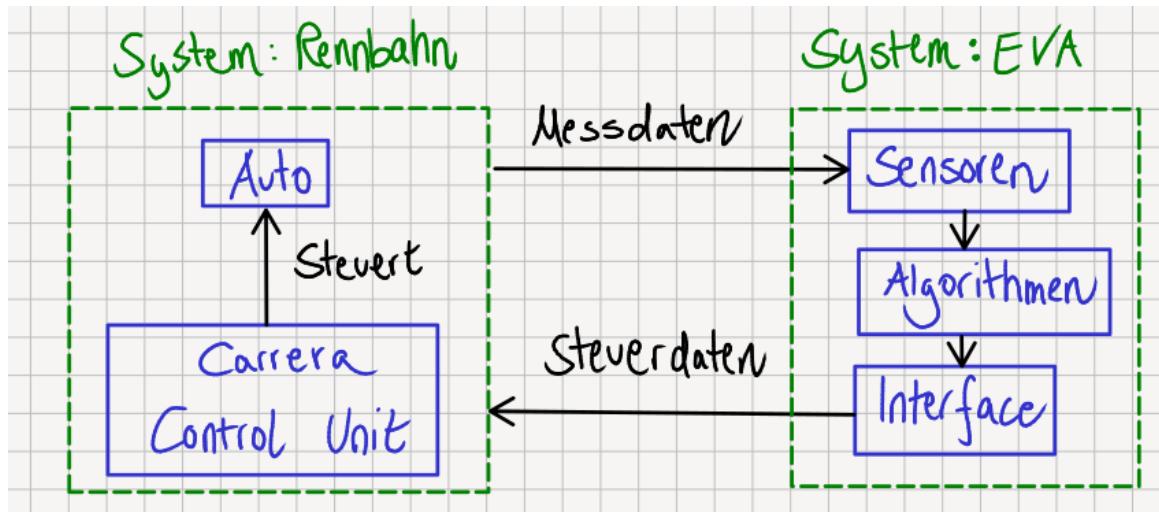


Abbildung 4: Grobe Systemarchitektur

2 Stand der Technik

Die Automatisierung einer Autorennbahn ist ein beliebtes Projekt für verschiedenste Felder der Forschung. Bei der Literaturrecherche finden sich häufig ähnliche Ansätze. Diese sind unter anderem

Methoden der Computer Vision

Es ist möglich, mit einer oder mehreren Kameras eine Rennstrecke in ihrer Gesamtheit zu überblicken. Verschiedene Typen von Kamera können genutzt werden, um wichtige Informationen zur Lage und Geschwindigkeit des Autos zu gewinnen. Der klassische Ansatz ist das Verwenden einer pixelbasierten Kamera. Aus dem Bildmaterial kann mit relativ großem algorithmischem Aufwand die relevante Information extrahiert werden. Zusätzlich kann der Streckenverlauf von den Pixeldaten abgeleitet werden. Die zu verarbeitende Datenmenge ist hier sehr hoch, obwohl jeweils nur eine kleine Untermenge von Pixeln zu jedem Auslesezeitpunkt relevant ist. Diesen Gegebenheit hat sich „STUDY OF AUTOMATIC CONTROL ALGORITHMS FOR A SLOT CAR“ im Jahr 2013 zunutze gemacht [2], indem

immer nur ein Ausschnitt - die sogenannte Region Of Interest (ROI) verarbeitet wird. In dieser Arbeit wurde abseits der Verwendung der Daten für eine Steuerung des Autos noch eine Anzeige für Nutzer implementiert. Diese erfasst Momentangeschwindigkeit, mittlere Geschwindigkeit, Spitzengeschwindigkeit, die Position des Autos sowie Rundenzeit.

Allerdings haben pixelbasierte Kameras häufig eine geringe Zeitauflösung. Eine gängige Bildwiederholungsfrequenz ist 60Hz , das entspricht einer Abtastdauer von $16.\bar{6}\text{ms}$. Manche Kameras können mehr Bilder pro Sekunde aufnehmen. Mit steigender Bildrate sinkt jedoch die Pixelauflösung und die Lichtmenge, die der Kamera zur Verfügung steht. Pixelbasierte Kameras können also durchaus für die Anwendung eingesetzt werden, jedoch gibt es eine geeigneter Form von Hardware, wenn es um bewegte Objekte geht.

Delbruck et al. [3] [4] arbeiten seit 2010 an der Entwicklung und Optimierung einer autonomen Steuerung, die einen DAVIS für die Erfassung der Daten verwendet. DAVIS steht für (event-based) Dynamic and Active pixel VIision Sensor. Die in [3] verwendete Kamera unterstützt Pixelausgabe und auch sogenannte Events - also Ereignisse der Pixelveränderung. Dadurch können die Vorteile der Pixelkamera genutzt werden (z.B. Streckenerkennung auf Pixelbasis), aber auch Bewegung innerhalb des Bildes mit niedriger Latenz und hoher Abtastrate verfolgt werden. Die Datenrate bleibt dabei vergleichsweise gering.

Anwendung von neuronalen Netzen

Das Fahren des Autos kann von neuronalen Netzen übernommen werden. Diese können mithilfe einer geeigneten Bewertungsfunktion die auf die Minimierung der Rundenzeit trainiert werden. Allerdings muss besondere Sorgfalt in die Trainingsdaten investiert werden - schließlich soll ein autonomes Auto auf jeder Strecke fahren können, ohne vorher lange trainiert werden zu müssen. In „The Neuro Slot Car Racer: Reinforcement Learning in a Real World Setting“ [5] wird ein neuroevolutionärer Ansatz gewählt. Information über die Position und Geschwindigkeit des Autos wird mit einer pixelbasierten Kamera erhoben. Der Rechenaufwand eines solchen Algorithmus' in Training und Anwendung ist aufgrund der hohen Datenmenge der Kamera beachtlich. Allerdings ist der Erfolg auf der Strecke respektabel: Auf der Trainingsstrecke sind nur 21% aller trainierten Netze entgleist. Von den erfolgreichen Netzen hatte das Beste eine Rundenzeit von 4.53s , im Mittel haben die erfolgreichen Netze eine Rundenzeit von 4.91s erreicht - der Menschenrekord auf dieser Strecke war 4.50s .

Markierung der Strecke

Um die Position des Autos entlang der Strecke zu bestimmen, kann diese auch mit Markierungen versehen werden. In „Autonomous Slot-car System“ [6] werden die Enden von Streckensegmenten mit Klebestreifen markiert. Diese Methode erlaubt bei Passieren eines Streifens die Aktualisierung der Autoposition zumindest segmentweise. Hier wurde die Strecke konstant gewählt und eine Steuerung des Autos in Abhängigkeit der Position entlang der Strecke realisiert. Dieser Ansatz hat den Nachteil, spezifisch für eine Strecke zu sein - die Steuerung für jede neue Strecke muss von Grund auf implementiert und optimiert werden. Vorteil ist der geringe Rechen- und Materialaufwand sowie die vergleichsweise gute Rundenzeit.

In „The Slot Car Stig: Performance and Consistency of a Slot Car Driven by a Heuristic Algorithm in an Embedded Microcontroller“ [7] wird jedes Streckensegment mit einem Barcode versehen. Dieser gibt abseits der Position des Autos noch Information über das Streckenteil - so können Kurven und Geraden ohne weitere Algorithmen voneinander unterschieden werden. Die Methodik teilt viele der Vorteile und Nachteile von „Autonomous Slot-car System“ [6].

Die markierungsbasierten Ansätze sind daher vielversprechend, allerdings besonders streckenabhängig. Eine automatische Streckenerkennung kombiniert mit einem allgemeinerem Algorithmus könnte eine Alternative zu den komplexen und rechenintensiven Ansätzen der Computer Vision und neuronalen Netzen darstellen. Daher nutzt EVA den erweiterten Ansatz der Streckenmarkierung - eine automatische Streckenerkennung kombiniert mit einem allgemeinen Algorithmus, der auf allen Strecken funktioniert.

3 Auswahl der Hardware

3.1 Wahl der Sensoren

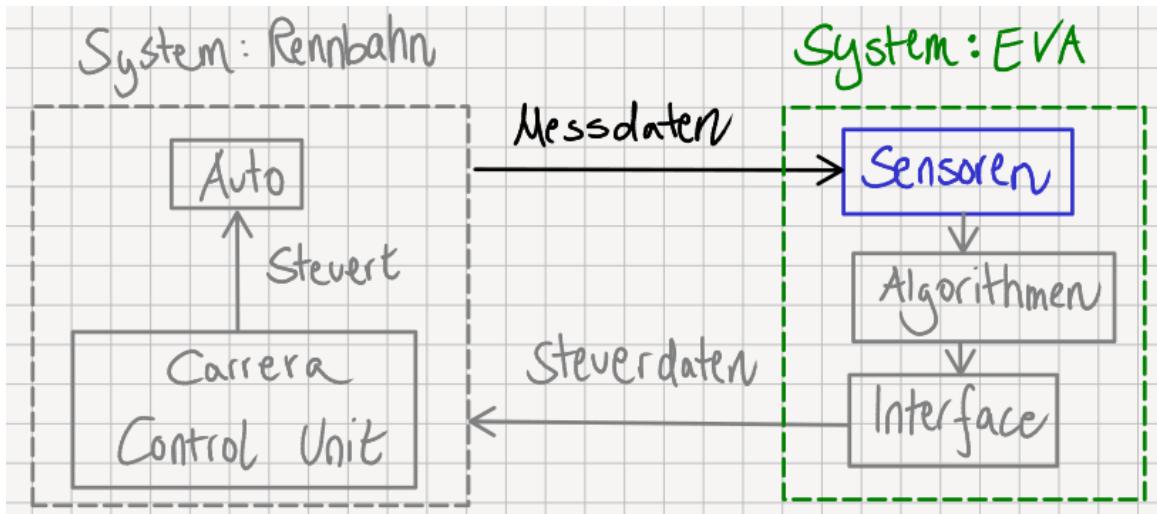


Abbildung 5: Grobe Systemarchitektur mit Fokus auf die Messdaten

Wie bereits in Sektion 1.2 erwähnt, ist für die Implementierung eines autonomen Autos die Verwendung von Sensoren zur Messung vom Istwert besonders relevant. Wichtig ist vor allem die Geschwindigkeit des Autos, da diese die zu regelnde Größe ist, denn die Geschwindigkeit hat einen Einfluss auf die Entgleisung sowie die Rundenzeit. Des Weiteren muss bereits vor einer Kurve bekannt sein, dass gleich eine Kurve auftritt - erst recht, weil das Auto ab Werk nicht über eine Bremsfunktion verfügt. Wie viele reale mechanische Systeme hat das Auto eine gewisse Trägheit. Um also eine Zielgeschwindigkeit früh genug erreichen zu können, muss rechtzeitig verzögert werden.

Bestimmung der Autogeschwindigkeit

Um die Geschwindigkeit eines Objekts zu messen, gibt es einige bewährte Methoden und Sensoren. Es gibt die Option, die Geschwindigkeit mit einem Sensor, der am Auto selbst befestigt ist, zu messen oder die Messung von einem externen Betrachtungspunkt durchzuführen.

Beispiele für Sensoren, die sich für externe Geschwindigkeitsbestimmung eignen, sind

- Radar und verwandte Technologien wie Lidar, die die Distanz zu einem Objekt zu zwei Zeitpunkten bestimmen können und mit $\vec{v} = \vec{ds} \div dt$ die Geschwindigkeit bezogen auf den Beobachtungspunkt bestimmen.

- Eine Kamera, die die Position der Pixel, die das Auto repräsentieren, über die Zeit erheben kann und so die Geschwindigkeit ermittelt.

Externe Betrachtung bedeutet leider meist auch, dass die bestimmten Werte in Relation zum Beobachtungspunkt stehen und daher erst auf das Bezugssystem (beispielsweise Messung innerhalb des Autos) umgerechnet werden müssen.

Sensorik, die die Geschwindigkeit innerhalb des Bezugssystems messen können, sind unter anderem

- Tachometer, optisch wie mechanisch, die Rotation in ein Signal übersetzen. Die Menge an Rotationen in einer Zeiteinheit, die ein Rad durchführt, ist proportional zur Geschwindigkeit eines Autos. Allerdings wird hierbei ein Ausrutschen der Reifen, der sogenannte Schlupf, nicht mitgemessen.
- Optische Flusssensoren [8] wie sie heutzutage gängig in optischen Computermausen sind, befestigt am Auto, die ähnlich wie Kameras funktionieren. Die Bewegung der Pixel im Bild wird hier als Signal aufgefasst. Ist der Sensor auf die Strecke gerichtet, so wird Schlupf berücksichtigt.
- Inertiale Messeinheiten (inertial measurement units, IMUs) die heute weit verbreitet sind und Kräfte entlang ihrer Achsen messen können. Zudem verfügen einige IMUs auch über Gyroskope und Magnetometer. Da $\vec{F} = m \cdot \vec{a}$, ist es gängig, mit ihnen Beschleunigungen zu messen. Die Geschwindigkeit kann dann als Integral der Beschleunigung nach der Zeit berechnet werden.

Bestimmung der Distanz zur nächsten Kurve

Um in Erfahrung zu bringen, wo die nächste Kurve platziert ist, sind die folgenden Schritte durchzuführen:

1. Streckenverlauf in Erfahrung bringen. Streckengeometrie (welcher Teil der Strecke ist eine Gerade, welcher eine Kurve) ist eine Funktion der Distanz auf der gefahrenen Strecke.
2. Momentanposition des Autos s_{Momentan} entlang der Strecke berechnen.
3. Position der Kurve, die nächsten zu s_{Momentan} ist, bestimmen. Diese Position ist s_{Kurve} .
4. Distanz zur nächsten Kurve ist die Differenz $s_{\text{Kurve}} - s_{\text{Momentan}}$.

Um die Position des Autos entlang der Strecke zu bestimmen, kann die Autogeschwindigkeit nach der Zeit integriert werden. Andernfalls kann die Strecke

markiert und die Markierungen gelesen werden. Dieser Ansatz bringt zwar eine gröbere zeitliche Auflösung mit sich, hat aber dafür den Vorteil, nicht sehr abhängig von absoluten Messfehlern (DC Offsets) zu sein.

Unabhängig davon, wie die Momentanposition bestimmt wird, ist es wichtig, die Streckengeometrie vorab zu kennen. Dafür muss die Strecke für eine Runde ab gefahren werden und die Streckenteile müssen klassifiziert werden oder ein Sensor wie eine externe Kamera muss die komplette Strecke überblicken können. Ansonsten kann die Kurve auch von einem Sensor detektiert werden, der in eine gewisse Distanz vor dem Auto tasten kann, wie eine Kamera oder ein Reflexionssensor, der die Krümmung der glänzenden Schiene im Vorraus erkennt.

Für das EVA-Projekt werden zwei Infrarot-Reflexionssensoren zur Erkennung der Streckengeometrie und segmentweise Positionsbestimmung eingesetzt. Die Markierung der Strecke erfolgt durch Bekleben der Enden von Streckensegmenten mit retroreflektivem Klebeband. Diese Markierungen sind beispielsweise auf der Titelseite dieses Dokuments und in Abbildung 6 zu sehen. Zur Messung der Geschwindigkeit kommen IMUs zum Einsatz. Beide Sensoren haben den Vorteil, nicht invasiv und Solid State zu sein.

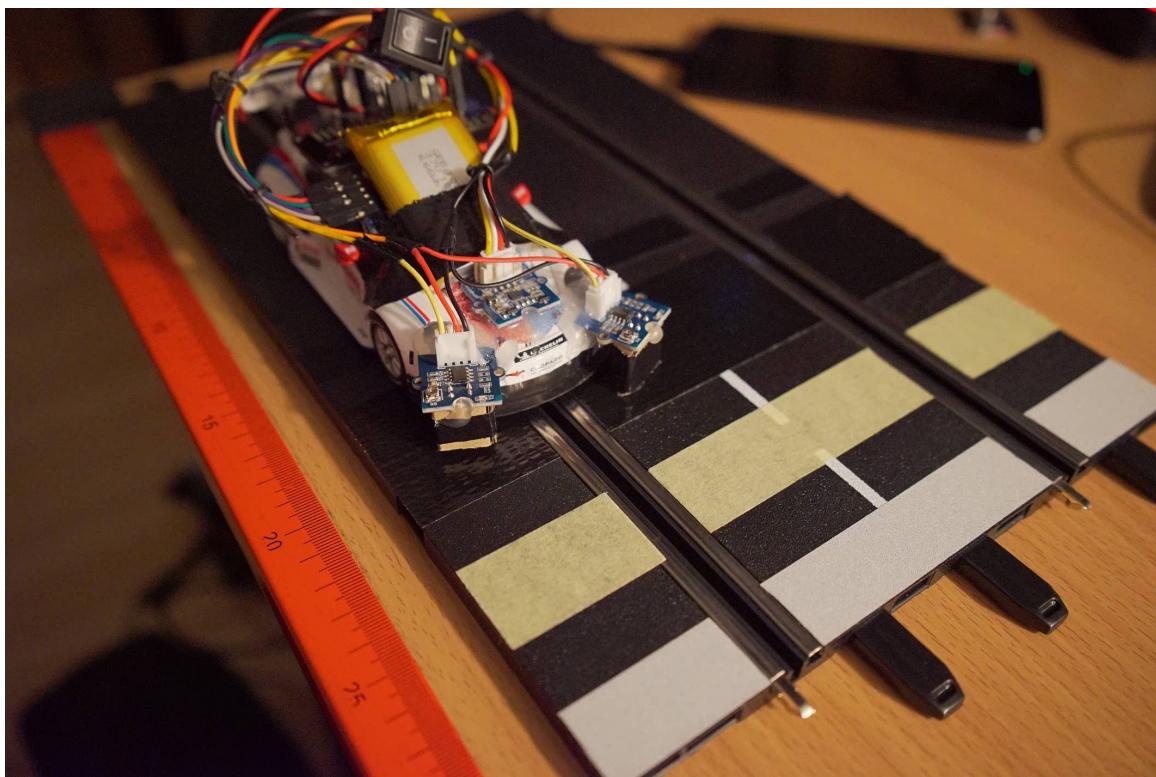


Abbildung 6: Messaufbau zum Vergleich der Reflektivität verschiedener Oberflächen. Von rechts nach links: Retroreflektives Klebeband, silber; Rennstrecke, schwarz; Kreppband, gelb; Panzerband, schwarz; Isolationsband, schwarz

3.2 Wahl der Rennbahn

Die Wahl der Rennbahn für das Projekt ist wichtig, da diese als Plattform von EVA gesteuert und vermessen werden muss. Einige Charakteristiken, die beim Kauf einer Rennbahn Relevanz aufweisen, sind:

- Digitale oder analoge Ansteuerung der Autos. Analoge Bahnen arbeiten wie in Sektion 1 beschrieben mit der Variation des Spannungspegels der Schienen, um das Auto zu beschleunigen oder zu verzögern. Digitale Bahnen haben in der Regel eine feste Versorgungsspannung, die schnell ein- und ausgeschaltet wird, um Informationen an das Auto zu übermitteln. Dieses dekodiert die empfangene Information und steuert den Motor mithilfe eines Motortreibers autointern.
- Verschiedene Schnittstellen an der Kontrolleinheit (Control Unit, CU). Einige Bahnen verfügen über standardisierte Schnittstellen wie UART.
- Verschiedene Maßstäbe der Modellautos. 1 : 48, 1 : 32 und 1 : 24 sind üblich im Hobbybereich.

Für das EVA-Projekt wurde die Carrera Digital 124/132 Plattform gewählt. Diese erlaubt die Verwendung von Autos im Maßstab 1 : 24 oder 1 : 32. Bei Autos in dieser Größe ist genug Platz für Sensorik und Microcontroller - zudem weisen die Autos ein größeres Gewicht auf. Damit ist die relative Gewichtszunahme durch EVA-Hardware nicht so gravierend wie bei kleineren und leichteren Autos. Das Schienensystem ist kompatibel mit analoger und digitaler Ansteuerung, für das Projekt wurde aber die digitale Ausführung gewählt. Die Literatur verwendet nahezu ausschließlich analoge Rennstrecken, weshalb ein Ansatz mit einer digitalen Strecke ein interessantes neues Segment erforscht. Allerdings sind digitale Strecken nicht ohne ihre Nachteile: die digitale Umsetzung fügt der Rennstrecke Latenzen hinzu, mit der eine analoge Strecke nicht zu kämpfen hat.

4 Funktionsweise und Features der digitalen Rennbahn

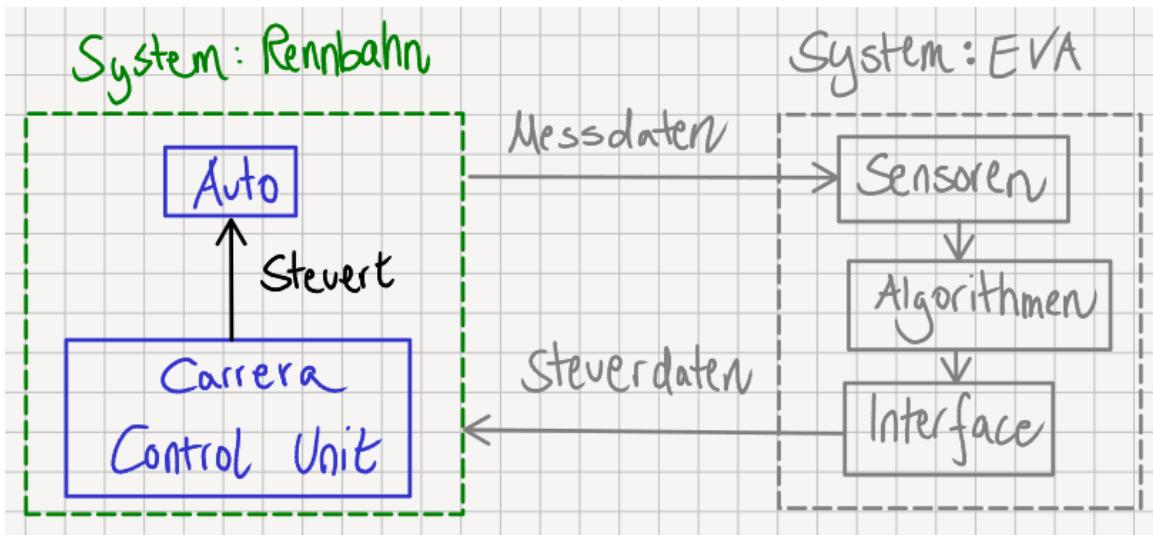


Abbildung 7: Grobe Systemarchitektur mit Fokus auf das System Rennbahn

4.1 Features

Die Carrera D124/132-Plattform verfügt über einige eingebaute Funktionen, die EVA nutzen kann oder umgehen muss. Einen kompletten Überblick über die Features gibt die Nutzungsanleitung vom Hersteller [9]. Zu den Funktionen gehören mitunter:

- Strom-Spar Funktion

Nach einer Nichtnutzung der Rennbahn für 20 Minuten schaltet sich die Rennbahn ab und ist nicht mehr zu wecken ohne sie für zwei bis drei Sekunden mit dem Netzschalter abzuschalten und wieder einzuschalten. Diese Funktion ist für den Dauerbetrieb an einem Tag der offenen Tür oder ähnlichen Veranstaltungen eher hinderlich. Daher hält EVA die Rennbahn wach, indem alle fünf Minuten bei Inaktivität eine Handregleraktion emuliert wird.

- Einstellung Geschwindigkeits- und Bremsverhalten

Die Bahn lässt den Nutzer in zehn Stufen einstellen, wie schnell das Auto beschleunigt und verzögert sowie was die maximale Geschwindigkeit sein soll. Hier ist Stufe 10 der „Normalzustand“, bei dem der Motor maximal genutzt wird. Alle niedrigeren Stufen führen zu einer künstlichen Limitierung der Fähigkeiten des Autos. Für EVA wird daher immer Stufe 10 für Geschwindigkeits- und Bremsverhalten verwendet.

- Rennen

Rennen können von Hand durch das Drücken einer Taste auf der Control Unit gestartet und gestoppt werden. Die Bahn verfügt über einen Countdown, der mit Tönen und Lichtern signalisiert, wann ein Rennen anfängt. Zusätzlich werden Frühstarts erkannt und angezeigt. Das Drücken der Taste sowie der Zustand des Startzählers und Erkennung von Frühstarts sind über die serielle Schnittstelle abrufbar.

- Rundenzeiten

Über die serielle Schnittstelle ist es auch realisierbar, Werte zu erhalten, von denen die Rundenzeiten ableitet werden können. Wie genau das im Detail funktioniert, ist in Sektion 4.2 ausgeführt.

- Programmieren von Autos auf diverse Steckplätze

Da die Strecke digital ist, ist es möglich, auf einer Schiene zwei Fahrzeuge zeitgleich zu betreiben. Jedes Auto ist seinem eigenen Steckplatz für Handregler zugewiesen - die Control Unit besitzt vier solcher Steckplätze. Es ist auch realisierbar, einem Steckplatz mehrere Vehikel zuzuweisen, allerdings gilt das nicht vice versa. Welchen Steckplatz ein Auto hat, ist für dessen Eingabelatenzen relevant und hat einen merklichen Einfluss auf die „Steuerbarkeit“ des Autos für Mensch und Maschine zugleich.

- Betrieb eines Pace Cars

Das Pace Car ist ein Auto, das eine konstante Geschwindigkeit fährt. Die Bahn gibt dem Nutzer die Möglichkeit, eine solche Geschwindigkeit festzulegen und ein Auto als Pace Car zu programmieren. So kann ein Rennen gegen ein Auto mit konstanter Geschwindigkeit ermöglicht werden.

4.2 Carrera D124/132 Protokolle

Die Protokollfunktionen und Realisierung in der Bahn wurden bereits von Stephan Heß [10] inoffiziell dokumentiert. Dennoch mussten einige Feinheiten, besonders die Dekodierung der Zeitstempel, durch eigenes Reverse Engineering herausgearbeitet werden.

4.2.1 Control Unit Datenprotokoll

In Abbildung 8 sind das Blockschaltbild und die Übertragung von der Control Unit an das Auto skizziert.

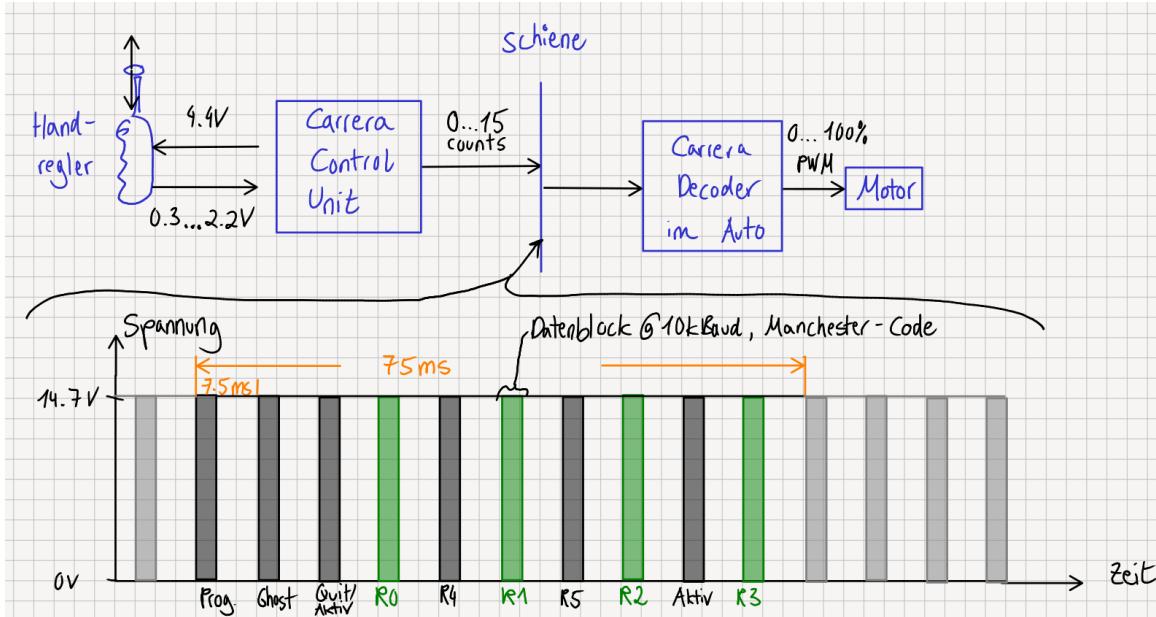


Abbildung 8: Blockschema der digitalen Rennbahn

Da für das Übertragen von Nachrichten nur die zwei Leiter für die Versorgungsspannung (oder die Luft) zur Verfügung stehen, greift Carrera auf eine Modulation der Versorgungsspannung zurück, um digitale Bitblöcke zu übertragen. Diese ist standardmäßig eine Gleichspannung von ca. 14.7V. Um die Versorgungsspannung nicht bei langen 0 oder 1 Folgen einbrechen zu lassen, werden die Bits im Manchestercode dargestellt. Hier ist die Information im Flankenwechsel enthalten. Eine fallende Flanke innerhalb des Auslesefensters wird als 1 bezeichnet, eine steigende Flanke als 0. Die Nachricht wird vom Auto empfangen und dekodiert. Je nach empfangenem Digitalwert wird die Versorgungsspannung des Gleichstrommotors verändert, um eine Geschwindigkeitsveränderung hervorzurufen.

Control Unit Datenprotokoll

Alle 75ms beginnt ein Übertragungszyklus. Dieser besteht aus zehn kleineren Übertragungen von der Control Unit zum Auto. Die kleineren Übertragungen finden alle in einem Abstand von 7.5ms statt, die Bitrate ist 10kb/s. Zwischen den letzten acht kleineren Übertragungen können andere Teilnehmer ihre eigenen Nachrichten senden. Grundsätzlich nutzt Carrera hier also eine Form von Time Division Multiple Access (TDMA). Die kleinen Datenblöcke erfüllen die folgenden Funktionen:

1. Datenblock: Programmierdatenwort

Wird hauptsächlich für Funktionen wie Geschwindigkeits- und Bremsverhalten modifizieren und Kommunikation mit anderen externen Geräten wie Rundenzählern verwendet.

2. Datenblock: Ghost

Dient der Steuerung des Pacecars.

3. Datenblock: Aktivdatenwort oder Quittungswort

Wurden im letzten $75ms$ -Übertragungszyklus Daten empfangen, so wird das Quittungswort gesendet, das den Fahrzeugen mitteilt, in welchen der acht möglichen Sendestellen eine Nachricht empfangen wurde.

Ist das nicht der Fall, so wird ein Aktivdatenwort gesendet. Dieses enthält die Information, ob der Schlegel eines Reglers weniger gedrückt ist als zuvor oder nicht. Vermutlich wird diese binäre Information von den Autos genutzt, um das Bremslicht ein- und auszuschalten.

4. Datenblock: Reglerdatenwort 0

Enthält den Geschwindigkeitswert für das Auto, das dem Steckplatz 0 zugewiesen ist. Das ist der Steckplatz, der am weitesten Links ist.

5. Datenblock: Reglerdatenwort 4

Enthält den Geschwindigkeitswert für das Auto, das dem Steckplatz 4 zugewiesen ist. Steckplätze 4 und 5 sind nur bei Rennbahnen mit drahtlosen Handreglern verfügbar.

6. Datenblock: Reglerdatenwort 1

7. Datenblock: Reglerdatenwort 5

8. Datenblock: Reglerdatenwort 2

9. Datenblock: Aktivdatenwort

10. Datenblock: Reglerdatenwort 3

Latenzen durch das Control Unit Datenprotokoll

Durch die Natur des Protokolls werden an zwei Stellen Totzeiten eingeführt. Die erste Verzögerung wird durch die Intervalldauer der $75ms$ -Übertragungszyklen herbeigeführt. Wird der Auslesezeitpunkt mit dem Reglerwert verpasst, so entsteht im schlimmsten Fall eine Latenz von $75ms$ zwischen gesendetem Wert und Übernahme des Werts. Dieser Zusammenhang ist in Abbildung 9 ausgeführt.

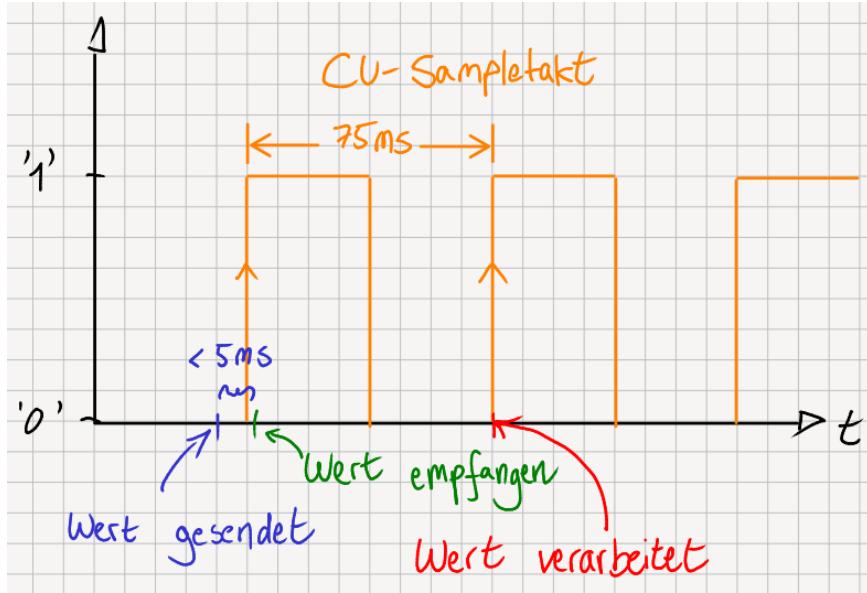


Abbildung 9: Latenzen durch den Sampletakt der Control Unit

Die zweite Verzögerung wird durch die fest programmierte TDMA-Methodik herbeigeführt. Dadurch sind die Latenzen immer gleich, unabhängig von der Anzahl aktiver Teilnehmer bei der Übertragung. Für das Auto in Steckplatz 0 besteht eine Mindestlatenz von 3 Blöcken ($7.5ms \cdot 3 = 22.5ms$), bis die Nachricht mit dem quantisierten Handreglerwert übertragen wird. Für das Auto in Steckplatz 3 besteht eine Mindestlatenz von 9 Blöcken ($7.5ms \cdot 9 = 67.5ms$), bis die Nachricht mit dem quantisierten Handreglerwert übertragen wird. Der Zusammenhang wird in Abbildung 8 klar. Er ist darauf zurückzuführen, dass eine Quantisierung des Handreglerwerts nur am Anfang des Übertragungszyklus stattfindet.

Die Kombination dieser zwei Totzeiten führt zu einer maximalen Latenz von ca. $100ms$ für ein Auto in Steckplatz 0 und ca. $140ms$ für ein Auto in Steckplatz 3 und ist auch für Menschen beim Fahren von Rennen spürbar.

Funktionsweise des Handreglers

Obwohl die Rennbahn digital ist, erfolgt die Steuerung immer noch mit einem analogen Handregler. Dieser enthält nur eine handvoll Komponenten: einen Widerstand, ein Potentiometer und einen Schalter. Das Innenleben eines Handreglers ist in den Abbildungen 10 und 11 dargestellt. Die individuellen Leiter im Kabel haben die folgende Funktion:

- Braun: Bezugspotential für den Handregler, GND (0V).
- Rot: 4.4V Versorgungsspannung für Spannungsteiler zwischen R1 und Potentiometer R2.

- Gelb: Ausgang des Spannungsteilers, Kontakt zwischen R1 und R2.
- Grün: 5V Spannung, die vom eingebauten Schalter im gedrückten Zustand auf GND gezogen wird. Der Pull-Up-Widerstand befindet sich in der Control Unit.

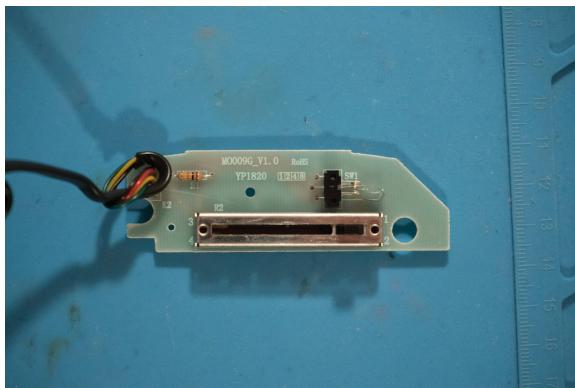


Abbildung 10: Oberseite des Handregler-PCBs

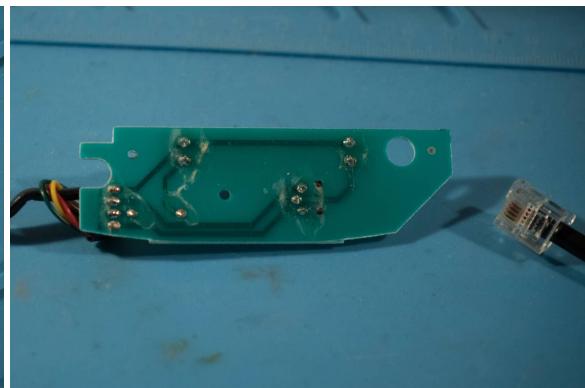


Abbildung 11: Unterseite des Handregler-PCBs

4.2.2 CU-Rundenzähler-Protokoll

Abseits der Kommunikation mit der Strecke weist die Carrera Control Unit noch eine serielle UART-Schnittstelle auf. Mit dieser können beispielsweise Rennen gestartet und gestoppt werden. Die Verbindungseinstellungen sind 19200 Baud, 8-N-1. Übertragen werden nur druckbare ASCII-Zeichen (20_H bis $7E_H$).

Soll eine Anfrage an die Control Unit gestartet werden, so muss die Nachricht mit einem "-Symbol (22_H , 34_D) beginnen. Je nach Kommando werden anschließend ein Befehlszeichen und manchmal auch zusätzliche Daten gesendet. Die Antworten der Control Unit beginnen mit dem Befehlszeichen als Bestätigung (ACK) und werden mit einer Prüfsumme und dem §-Symbol (24_H , 36_D) beendet.

Für EVA relevante Befehle sind die Abfrage der letzten Ziellurchfahrt mit Befehlszeichen ? ($3F_H$, 63_D) sowie der Tastendruck mit Befehlszeichen T (24_H , 84_D). Senden des Befehls "T1 ist äquivalent zum Drücken der Pace Car Taste. Senden des Befehls "T2 ist äquivalent zum Drücken der Starttaste.

Wenn eine Abfrage der letzten Ziellurchfahrt mit "?" gestartet wird, gibt es zwei mögliche Antworten unterschiedlicher Länge. Gab es keine Ziellurchfahrt von einem Auto seit der letzten Anfrage, so gibt die Control Unit unter anderem den ihren internen Zustand an. Die Zustände sind in Abbildung 12 zusammengefasst und werden von EVA genutzt, um interne Zustände daran zu synchronisieren.

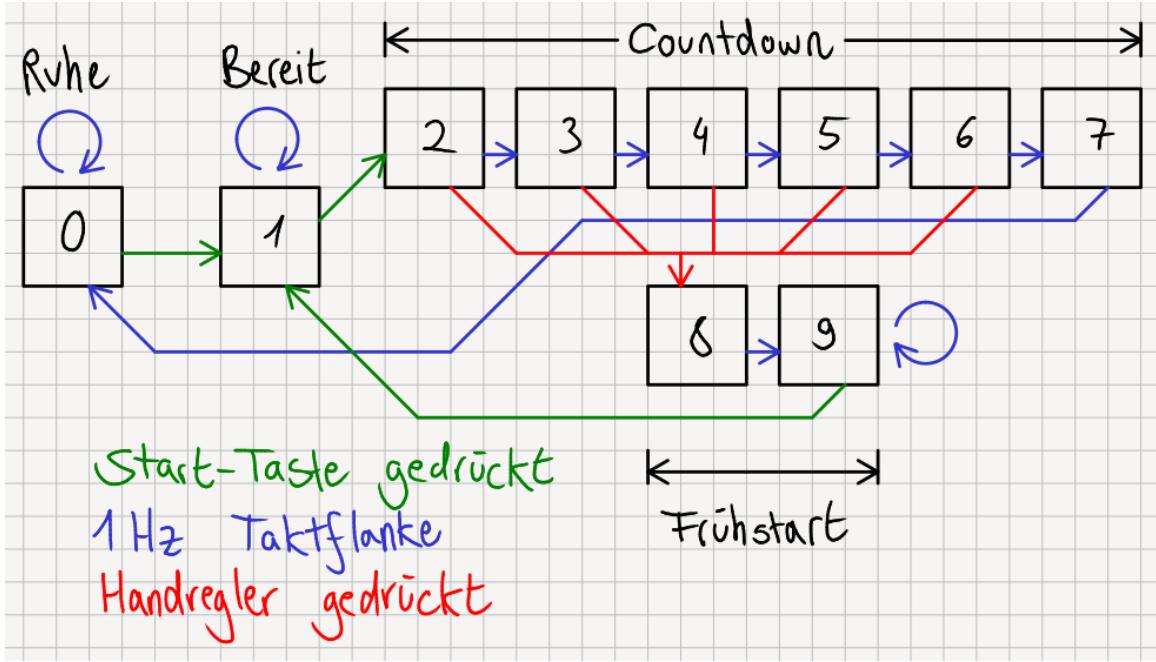


Abbildung 12: Zustandsfolgediagramm der Control Unit

Wird eine Abfrage der letzten Zieldurchfahrt gestartet und es gab eine Zieldurchfahrt von einem Auto seit der letzten Abfrage, so sendet die Control Unit eine Nachricht, die einen Zeitstempel sowie die Nummer des Autos, zu dem er gehört, enthält. Der Zeitstempel ist die Zeit, seit dem Control Unit zum Zeitpunkt der Zieldurchfahrt eingeschaltet ist und hat eine Auflösung von einer Millisekunde. Sobald ein Zeitstempel abgefragt wurde, ist er nicht mehr verfügbar und die CU überträgt bei weiteren Anfragen wieder ihren internen Zustand.

Dekodierung eines Zeitstempels

Die Dekodierung des Zeitstempels gestaltet sich als etwas komplexer als anfänglich angenommen, was hauptsächlich an der ungewöhnlichen Reihenfolge der Datenbits liegt. Abbildung 13 stellt eine Dekodierung graphisch dar. Für den Zeitstempel relevant sind die Bytes 1 bis 9. Byte 1 enthält die Nummer des Autos, das zuletzt die Ziellinie überquert hat, gültige Werte für EVA sind demnach 0...3. Die Nummer ist als ASCII-Buchstabe angegeben. Um den Zahlenwert zu erhalten, muss der Offset von 30_H bzw. 48_D subtrahiert werden. Die nächsten acht Bytes enthalten jeweils nur 4 Bit an Information, die im niedrigen Nibble enthalten ist. Das hohe Nibble ist in diesem Fall immer 3_H . Um das niedrige Nibble zu extrahieren, kann hier auch der Offset von 30_H bzw. 48_D subtrahiert werden. Jetzt sind also acht Nibbles mit Daten verfügbar. Diese repräsentieren vier Bytes, wobei hier das *niedrige Nibble zuerst* übertragen wird. Die resultierenden vier Bytes stehen für eine 32-Bit Dezimalzahl ohne Vorzeichenbehaftung, das *höchstwertige Byte* wird zuerst

übertragen.

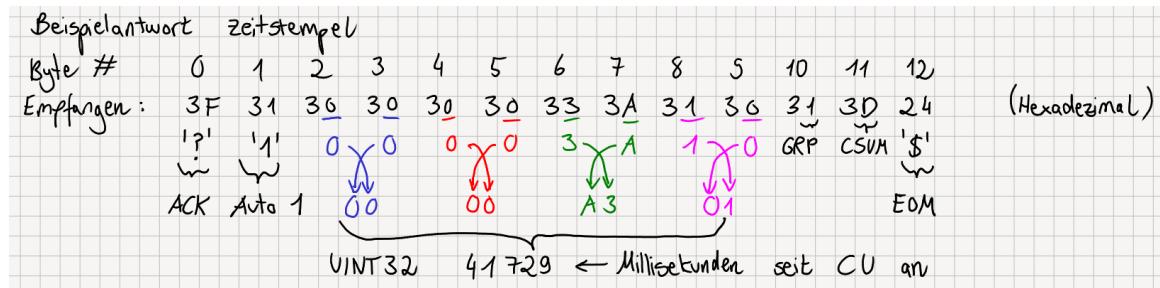


Abbildung 13: Schema zur Dekodierung eines Zeitstempels

5 Schnittstelle mit der Strecke

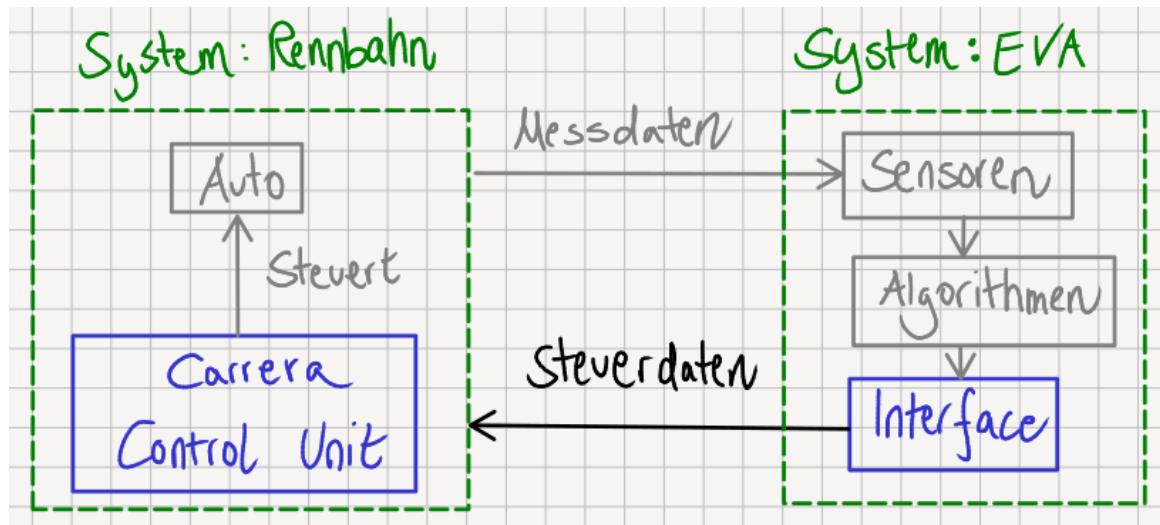


Abbildung 14: Grobe Systemarchitektur mit Fokus auf die Schnittstelle mit der Rennbahn

Um ein Auto auf einer solchen digitalen Strecke steuern zu können, gibt es prinzipiell zwei Ansätze:

1. Die Steuerung treibt mit eigener Elektronik den Motor direkt an, EVA klinkt sich im Blockschema nach Abbildung 8 also an einem Teil *hinter* dem Decoder im Auto ein.
2. Der digitale Teil wird genutzt, EVA klinkt sich im Blockschema nach Abbildung 8 also an einem Teil *vor* dem Decoder im Auto ein.

Direktantrieb des Motors im Auto bringt die folgenden Vorteile:

- Geringe Totzeit

- Hoher Grad an Kontrolle über Motorstrom und -spannung (feine Amplitudenauflösung erreichbar)
- Möglichkeit, Motorbremsung durchzuführen
- Stellgrößenfrequenz ist flexibel (feine zeitliche Auflösung erzielbar)

Allerdings ist die Implementierung eines Direktantriebes auch an einige Herausforderungen gekoppelt:

- Prozess ist invasiv, in Elektronik des Autos muss eingegriffen werden
- Sinnvolle Realisierung benötigt eine Platine
→ Wissen zum Platinendesign notwendig
- Wahl der Komponenten und Entwurf der Systemarchitektur erfordert Wissen in den Feldern der Leistungselektronik und elektrischer Maschinen
→ Zeitaufwändig
- Bei solch hoher Integration kann Spannungsversorgung direkt von Schiene erfolgen
→ Kein Akku benötigt → Gewichtersparnis

Die Alternative ist eine Nutzung der bestehenden digitalen Infrastruktur. Hier kann an verschiedenen Stellen eingegriffen werden:

1. Manuelles Pulsen der Schiene, Überbrückung der Control Unit
2. Einspeisen des Signals direkt vor den Decoder im Auto
3. Emulation eines Handreglers

Wenn das Protokoll und dessen festgelegte TDMA-Methodik nicht verletzt werden soll, bestehen zwischen den genannten Ansätzen im Endergebnis keine Unterschiede. Allerdings ist die Emulation eines Handreglers in der Realisierung mit Abstand die einfachste Methode - Generation einer analogen Spannung im Bereich von ca. $0.3V$ bis $2.2V$, die nicht stark belastet wird, ist mit geringem Aufwand verbunden. Viele moderne Microcontroller besitzen on-board Digital-zu-Analog-Wandler, die diese Aufgabe auch ohne externe Verstärker erfüllen können. Alternativ kann immer noch eine Pulsbreitenmodulation mit einem digitalen Pin realisiert werden. Durch hinzufügen eines einstufigen RC-Tiefpassfilters der richtigen Grenzfrequenz kann auch so eine analoge Spannung zwischen $0V$ und VCC , meist $3.3V$ oder $5V$, erzeugt werden.

Aus diesen Gründen wurde die Emulation eines Handreglers als Interface zur Rennstrecke gewählt. Abseits des bereits erwähnten geringen Implementierungsaufwands weist die Handregleremulation noch weitere Vorteile auf:

- Unterliegt gleichen Voraussetzungen wie Mensch
- Keine Modifizierung der Rennbahn erforderlich
- Geringe Menge an bestehender Literatur zum Thema Automatisierung einer *digitalen* Rennstrecke
→ Erweiterung des allgemeinen Wissens durch Beitrag von Projekt EVA

Jedoch ist die Emulation eines Handreglers nicht ohne Limitierungen:

- Hohe Protokolllatenzen, wie bereits in Sektion 4.2 angesprochen
- Nur 16 diskrete Spannungspegel, um den Motor anzutreiben (geringe Amplitudenauflösung)
- Maximal ein Stellwert alle $75ms$ (geringe zeitliche Auflösung)

6 Systemarchitektur

6.1 Aufbau des EVA-Systems

Um nun also ein System zu realisieren, das die relevante Sensorik und das Interface bereit stellen kann, wurde ein Ansatz mit zwei Komponenten gewählt. Ein Blockschema der Komponenten und Schnittstellen des EVA-Systems ist in Abbildung 15 dargestellt.

Ein Teil, der sogenannte Controller Emulator, oder auch CE, ist im Wesentlichen für die Kommunikation mit der Carrera Control Unit zuständig. Über einen der zwei integrierten DACs wird ein Handregler emuliert. Da der Microcontroller über mehrere Hardware-UARTs verfügt, kann die serielle Schnittstelle der Control Unit direkt mit ihm verbunden werden. In grau ist zusätzliche Peripherie dargestellt, die EVA um Funktionen für das User Interface (UI) erweitert.

Der zweite Teil ist das Sensorauto, auch manchmal als Sensorcar bezeichnet. Dieser besitzt auch einen Microcontroller, an den die Sensoren angebunden sind. Das Sensorcar ist abseits der Kommunikation mit den Sensoren auch für die Verarbeitung der Werte und Berechnung einer optimalen Stellgröße verantwortlich. Die berechnete Stellgröße wird dann an den Controller Emulator weitergereicht, der die Strecke ansteuert. In grau ist hier noch eine Schnittstelle zu externem Speicher

eingezeichnet. Dieser dient zum Speichern von Messdaten während der Fahrt, die zu einem späteren Zeitpunkt gelesen und verarbeitet werden können.

Einen Überblick über die grobe Verteilung der Funktionen gibt Abbildung 16.

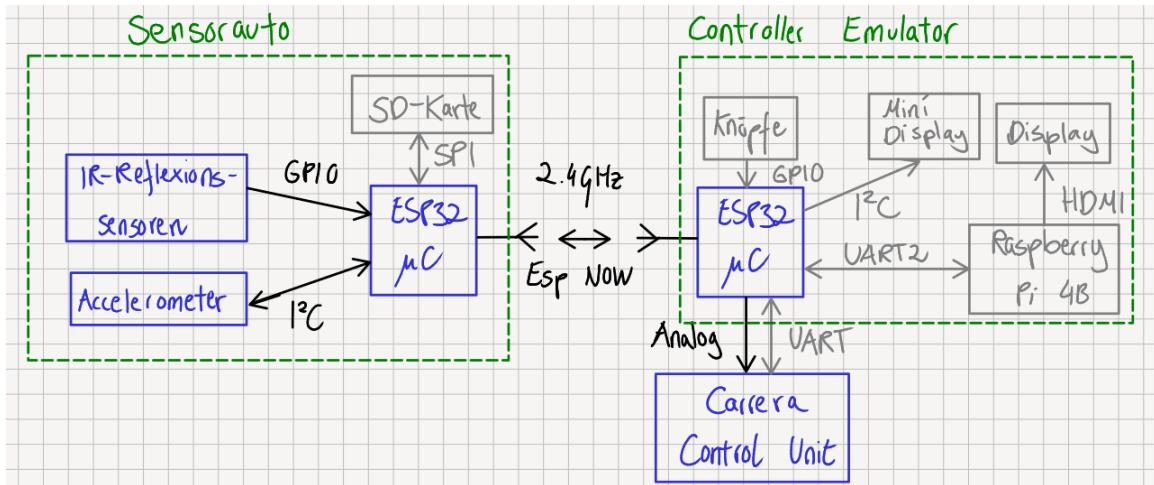


Abbildung 15: Blockschema der Schnittstellen des EVA-Systems

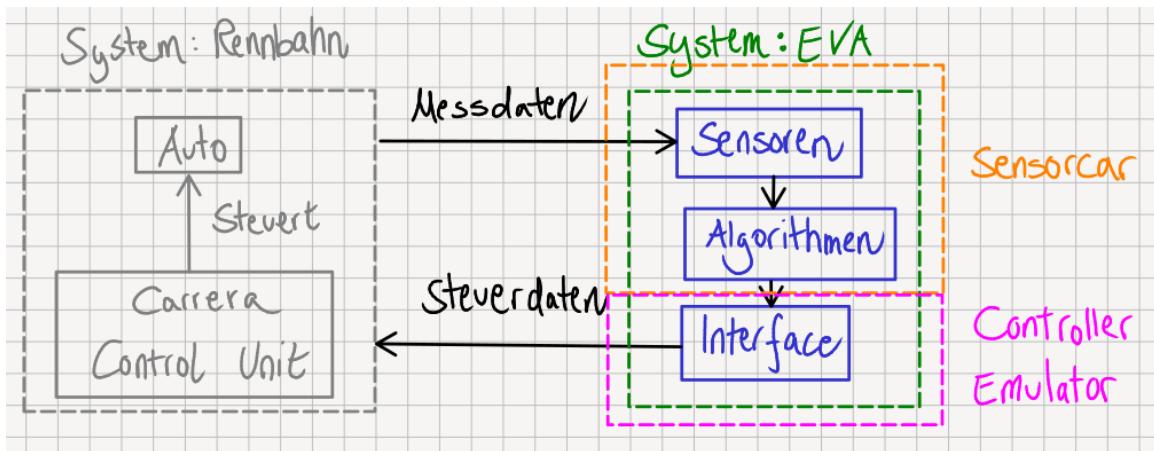


Abbildung 16: Aufteilung der Funktionen

Physischer Aufbau

Abbildungen 17 und 18 zeigen die Hardware, wie sie letztendlich aufgebaut wurde.

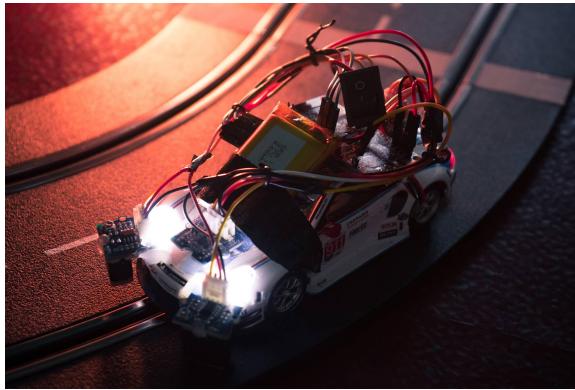


Abbildung 17: Physischer Aufbau des Sensorautos

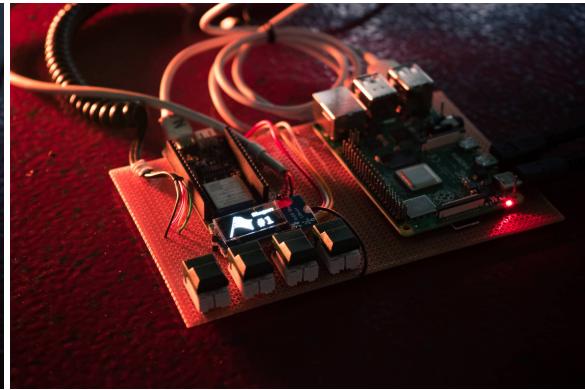


Abbildung 18: Physischer Aufbau des Handregler-Emulators

Das Sensorcar besitzt zwei Reflexionssensoren vor den Scheinwerfern. Stromversorgung des Sensorcars erfolgt über einen Lithium-Akku, der Ladeschaltkreis ist bereits auf der Platine des Microcontrollers implementiert. Die Stromversorgung kann mit einem Kippschalter getrennt werden. Auf der Motorhaube und am Heck sind die IMUs befestigt. Microcontroller und Kartenleser sind auf dem Dach angebracht. Für die Befestigung wurde Heißkleber verwendet, da er mit etwas Hitze und Isopropylalkohol leicht wieder entfernt werden kann. Das Auto wiegt 186g mit Akku und 147g ohne. Ab Werk hat das Auto ein Eigengewicht von 105g.

Der Controller Emulator wurde auf einer Lochrasterplatine angelötet und mit externen Knöpfen sowie einem kleinen OLED-Display verbunden. Die Funktion der Knöpfe ist (von Links nach Rechts):

- Handreglertaster → hat die gleiche Funktion wie der Taster eines normalen Handreglers (Licht an/aus, Zuweisen von Autos zu Ports, Schiene wechseln) und ist nicht mit dem Microcontroller verbunden.
- Rennstart/-stop → sendet den Start/Stop-Befehl an die Control Unit um ein Rennen wie in Sektionen 4.2 und Abbildung 12 beschrieben zu starten / stoppen.
- Rundenzahl erhöhen → erhöht die Anzahl Runden, aus denen ein Rennen besteht. Standard ist 10 Runden, Maximalwert ist 255 Runden.
- Rundenzahl verringern → verringert die Anzahl Runden, aus denen ein Rennen besteht. Minimum ist 1 Runde.

Das Display zeigt Informationen über Rundenzeiten, maximale Anzahl Runden in einem Rennen, sowie den Sieger eines Rennens an. Da das Display sehr klein ist, wird ein Raspberry Pi genutzt, um Ausgabe von solchen Informationen

über die serielle Schnittstelle auf ein zweites, großes Display zu übertragen. Abbildung 19 zeigt eine Sammlung von Beispielausgaben, die der Controller Emulator im Laufe eines Rennens über die serielle Schnittstelle erzeugen kann.

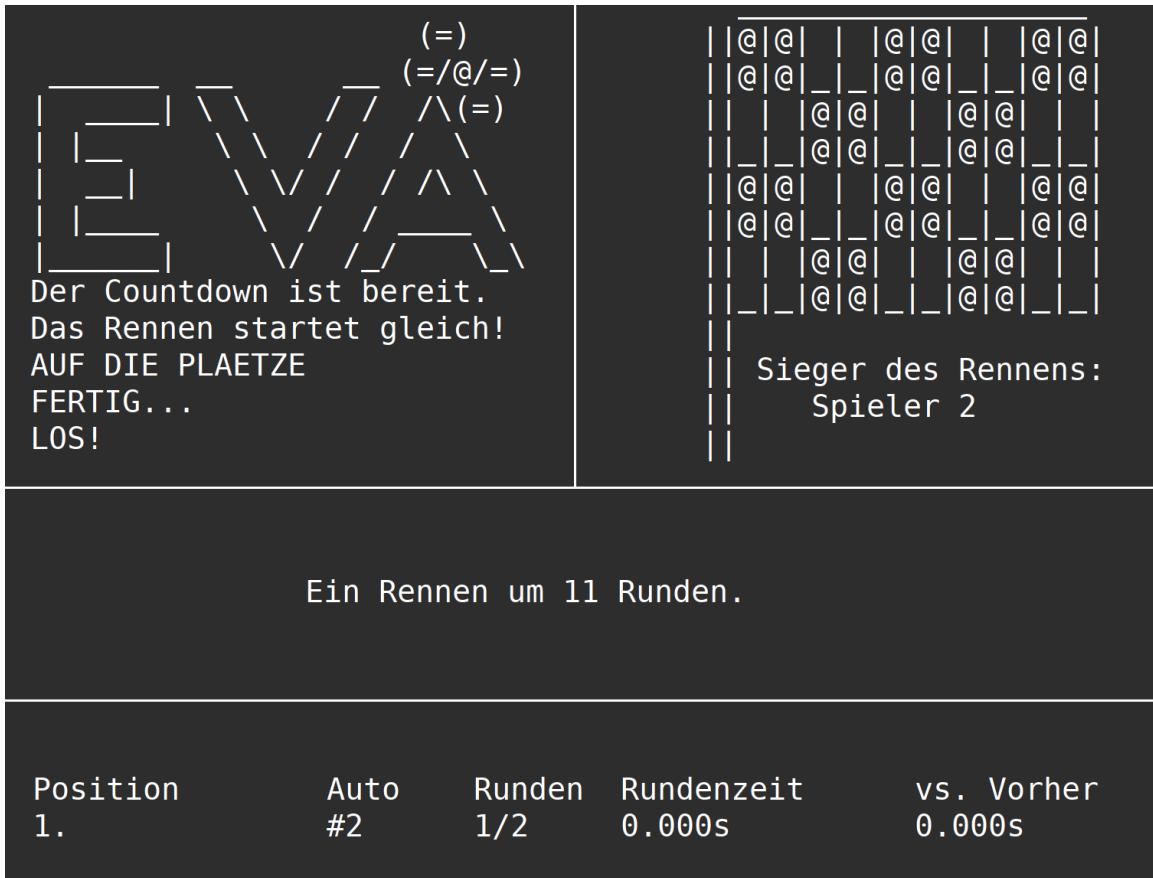


Abbildung 19: Ausgaben des Controller Emulators über die serielle Schnittstelle

Drahtlose Kommunikation

Sensorcar und Controller Emulator kommunizieren über eine drahtlose Verbindung, damit das Sensorcar sich selbst steuern kann. Das verwendete Protokoll ist ESP Now, ein proprietäres Protokoll vom Hersteller des Microcontrollers. Dokumentation zur Funktionsweise von ESP Now ist nicht öffentlich zugänglich. Aufgrund der zeitlichen Beschränkungen war es nicht praktikabel, die übertragenen Signale mit geeigneter Hardware zu empfangen und zu demodulieren. Allerdings konnten durch Messungen und Überlegungen einige Eigenschaften des Protokolls abgeleitet werden.

- Round-Trip-Time (RTT) beträgt in einem Raum mit guter SNR ca. 4ms...9ms bei Nutzdaten (Payload) von einem Byte
 - RTT steigt mit schlechter SNR, aber Protocol Data Units gehen nicht verloren

- Das Protokoll verfügt über Automatic Repeat reQuest (ARQ) Funktionalität
- Für Übertragung müssen MAC-Addressen der involvierten Boards bekannt sein
- Betriebsfrequenz ist 2.4GHz und in API können 13 verschiedene Kanäle gewählt werden, ESP32 Datenblatt [11] bestätigt Support von IEEE 802.11b/g/n
→ Hohe Wahrscheinlichkeit, dass IEEE 802.11 wireless LAN als Layer 2 Protokoll genutzt wird

Vermutlich werden also für Layer 1 und 2 Standardverfahren verwendet. Protokolloverhead bei einem Ethernet-Frame ist mindestens 7 Byte Präambel, 1 Byte SOF, 14 Byte Ethernet-Header sowie 4 Byte Prüfsumme. Das macht 26 Bytes Overhead. Ein Ethernet-Frame mitsamt Präambel und SOF muss mindestens 72 Byte groß sein - ist er kleiner, so wird der Rest des Layer 2 Payloads mit dem Ethernet-Pad am Ende der Nachricht aufgefüllt. Somit ist das effizienteste mögliche Layer 2 Payload bis zu 38 Bytes groß. Da ESP Now seinen eigenen Overhead mitbringt, ist die Menge an Nutzdaten zusätzlich beschränkt. Unabhängig von der Größe des ESP Now Overheads wird eine der kleinstmöglichen Protocol Data Units mit einem Payload von einem Byte realisiert. Abbildung 20 zeigt eine Protocol Data Unit (PDU).

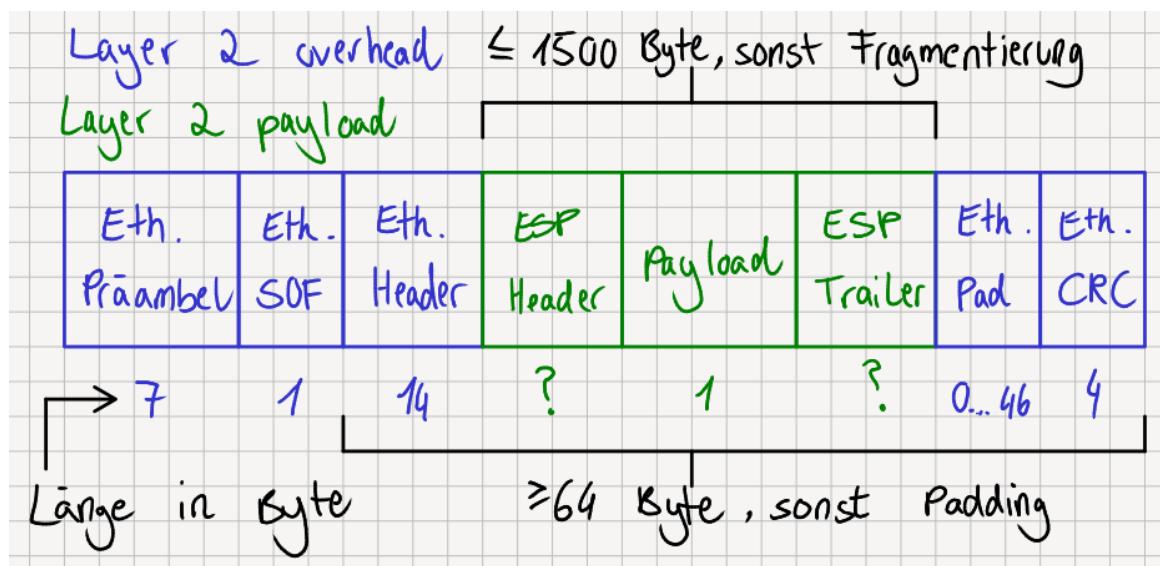


Abbildung 20: Größen einer PDU

Die Protocol Data Units (PDUs) können auch größer als die Mindestlänge sein, allerdings kommt das dann auf Kosten von erhöhter Latenz. Bei paketbasierten digitalen Funkübertragungen sind die folgenden Zeiten wichtig:

1. Die Paketisierungsdauer - das ist die Zeit, die vergeht, bis eine komplette PDU in den Kanal gesendet wurde.

$$T_p = \frac{N}{R}$$

Mit T_p als Paketisierungsdauer, N als Menge der zu übertragenden Bits und R als Bitrate der Übertragung. Die komplette Übertragungszeit enthält immer zwei Paketisierungsdauern T_p , eine beim Senden in den Kanal und eine beim Empfangen, da nur vollständige PDUs verarbeitet werden können.

Um T_p klein zu halten, muss die Bitrate der Übertragung R möglichst groß sein. Das kann erzielt werden, indem die SNR maximiert wird. Bessere SNR bedeutet die Möglichkeit, in Layer 1 mehr Konstellationspunkte zu verwenden. Diese können längere Bitfolgen repräsentieren und damit können in einer Zeiteinheit mehr Daten übertragen werden. Darüber hinaus müssen weniger PDUs wiederholt werden, wenn weniger Übertragungsfehler stattfinden. Gerade IEEE 802.11 wireless LAN nutzt zur Behebung von Übertragungsfehlern Send-And-Wait als ARQ-Protokoll, eines der ineffizientesten Verfahren für automatic repeat requests.

Für die Menge der zu übertragenden Bits N muss allerdings ein Kompromiss gefunden werden. Aufgrund des Overheads ist das Senden von kleinen PDUs ineffizient: die effektive Datenrate sinkt, da ein erhöhter Prozentsatz der gesendeten Daten Protokolloverhead ist. Zudem steigt die Prozessorlast, da mehr PDUs verarbeitet werden müssen. Aufgrund der Paketisierungsdauer ist das Senden von großen PDUs jedoch mit einer Steigerung der Latenz verbunden.

2. Die Verarbeitungsdauer. Besonders bei den verwendeten Microcontrollern, die abseits der Datenübertragung noch andere Tasks zu bearbeiten haben, ist diese relevant. Sender und Empfänger haben jeweils ihre eigene Verarbeitungsdauer.
3. Die Übertragungszeit der elektromagnetischen Wellen in Luft. Diese ist aufgrund der hohen Übertragungsgeschwindigkeit bei diesen kurzen Distanzen allerdings vernachlässigbar klein (bei einer Distanz von 30m: $\tau = \frac{s}{c} \approx 100\text{ns}$). Zudem geht der Zeitwert bei einer unidirektonalen Übertragung nur einmalig ein.

6.2 Integration der Sensoren

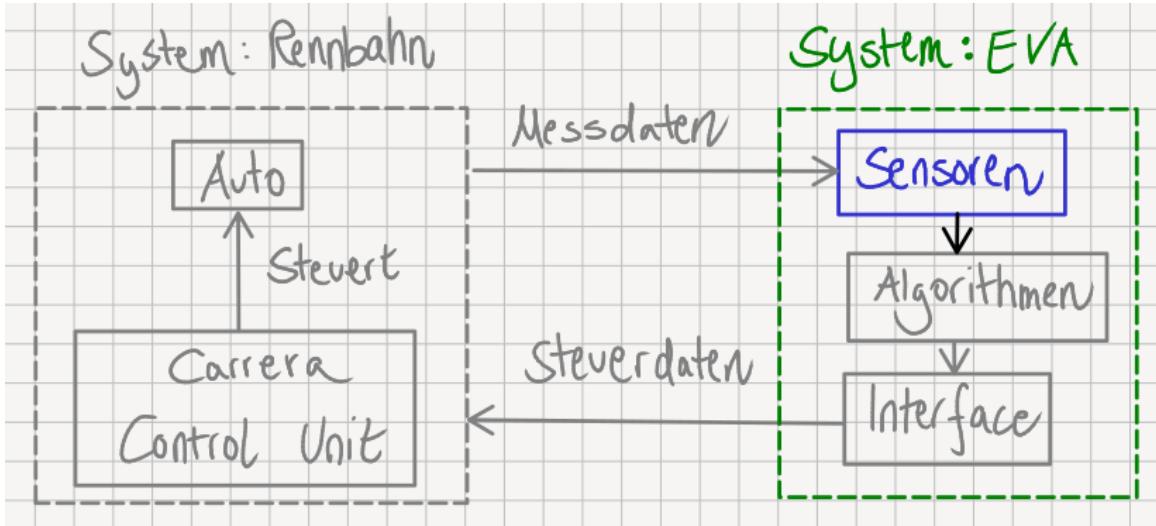


Abbildung 21: Grobe Systemarchitektur mit Fokus auf die Schnittstelle mit den Sensoren

Intertiale Messeinheit (IMU) LSM6DS3

Die IMU LSM6DS3 wurde aufgrund ihrer guten Rauschcharakteristiken und hoher Output Data Rate (ODR) von $1kHz$ gewählt. Mögliche Schnittstellen sind I²C und SPI. Das einzige erhältliche Entwicklerboard mit dem IC verfügte leider nur über die I²C-Schnittstelle. Das Datenblatt [12] gibt keine Auskunft über die maximal unterstützte I²C Clockfrequenz. Durch Messungen mit einem Oszilloskop wurde die maximal unterstützte Geschwindigkeit als $10kHz$ erfasst. Obwohl eine eigene, optimierte C-Library für das Interface mit dem Sensor geschrieben wurde, ist die maximale ODR mit dieser Frequenz nur ca. $800Hz$ - allerdings beschränkt das Loggen auf die SD-Karte aufgrund der hohen anfallenden Datenmenge diese ohnehin auf nur $100Hz$.

Im I²C-Bus sind die Teilnehmer parallel geschaltet, Nachrichten werden mit Addressierung voneinander unterschieden. So verfügt jeder Busteilnehmer über seine eigene 8-Bit I²C-Adresse. Um mehrere Sensoren gleichen Typs am gleichen Bus verwenden zu können, brauchen diese eine unterschiedliche Adresse. Deshalb haben die meisten ICs mit I²C-Schnittstellen einen Pin SA0, mit dem das LSB der Adresse verändert werden kann. Bei mehr als zwei ICs mit gleicher Adresse kann dieser Pin bei jedem IC individuell vom lesenden Microcontroller manipuliert werden, um nacheinander von allen ICs zu lesen. Bei zwei ICs reicht es aus, einen Adresspin HIGH und den anderen als LOW zu wählen. Der Hersteller der Entwicklerplatine hat diesen Pin allerdings auf dem PCB mit einer Leiterbahn mit GND verbunden (siehe Abbildungen 22 und 23). Aus diesem Grund wurde

bei einem der zwei Sensoren die Leiterbahn mit einem Skalpell durchtrennt (siehe Abbildung 24). Der interne Pullup-Widerstand in der IMU zieht den Pin dann auf HIGH-Potential.

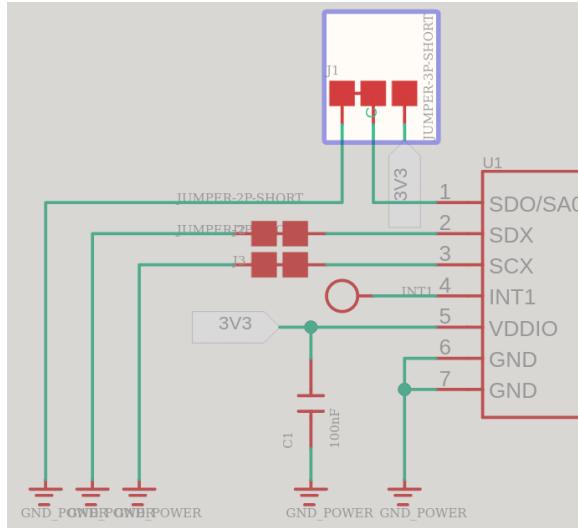


Abbildung 22: Schematic der IMU-Platine

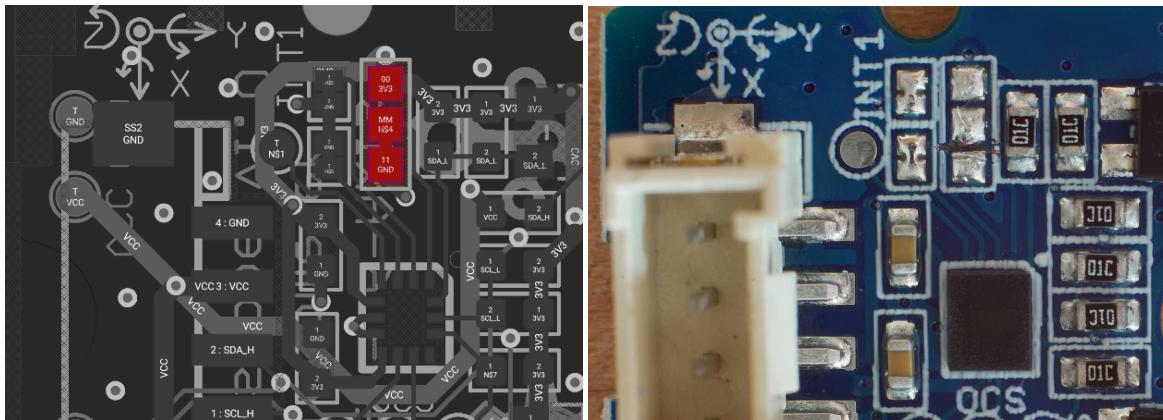


Abbildung 23: Verbindung zwischen SA0 und GND im Boardview

Abbildung 24: Durchtrennte Leiterbahn auf der Platine

Entwicklerboard Reflexionssensor

Der Reflexionssensor verfügt nicht über ein digitales Interface wie I²C oder SPI. Stattdessen hat er einen digitalen Ausgabepin. Die Funktionsweise des Sensorboards ist in Abbildung 25 schematisch dargestellt. Ein Infrarotdioden-Fototransistorpaar strahlt Licht im nahen Infrarotbereich auf eine Fläche. Ein Teil des Lichts wird zurückreflektiert. Je mehr Licht den Fototransistor erreicht, desto größer der Strom durch ihn. Dieser Strom wird mit einem Widerstand in eine Spannung gewandelt. Ein Rail-to-Rail Operationsverstärker dient als invertierender 1-Bit-DAC

(Komparator). Er vergleicht den Spannungswert, der durch den Fototransistor erzeugt wird, mit einer Referenz, die mit einem Potentiometer zwischen GND und V_{CC} (in diesem Fall 3.3V) einstellt werden kann.

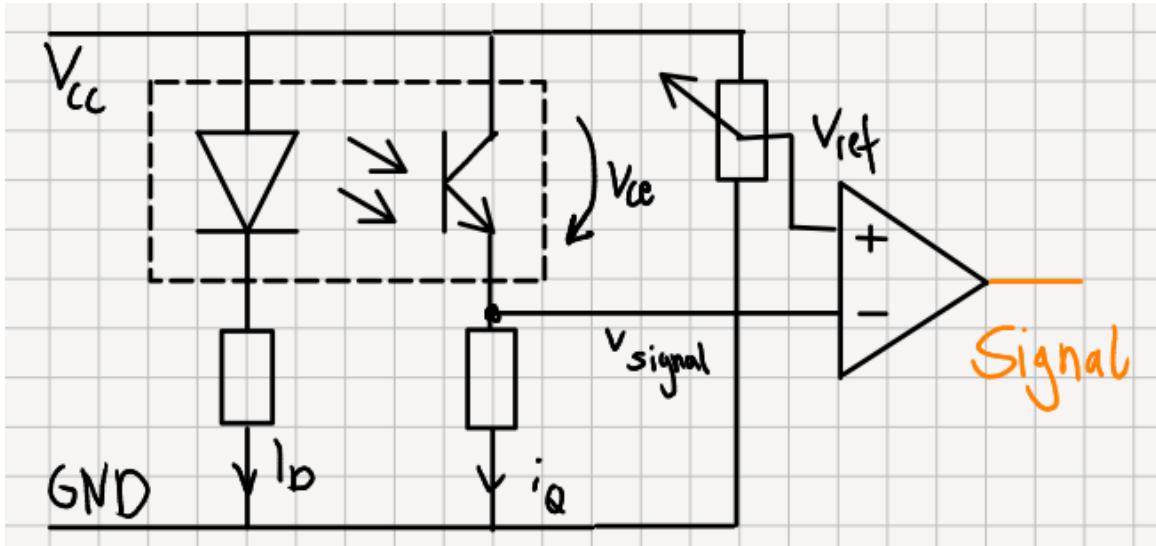


Abbildung 25: Schematische Darstellung der Funktionsweise des Boards

Die Reflexionssensoren sind in der Lage, ein digitales Signal in Abhängigkeit der Reflexion der Fläche unter ihnen zu generieren. Um verschiedene Typen von Oberflächen unterscheiden zu können, wurden einige Tests mit unterschiedlichen Materialien und Distanzen der Sensoren zur Oberfläche durchgeführt. Abbildungen 26 und 27 zeigen die Messaufbauten. Dazu wurde der Spannungspegel v_{Signal} am Fototransistor-Widerstand mit einem Multimeter gemessen. Abbildung 28 zeigt die Messergebnisse. Das schwarze Scotch Tape ist der beste umsetzbare Absorber, retroreflektives Klebeband 3M 610C in silber ist der beste Reflektor. Luft, also nichts in der Nähe zum Reflektieren des Signals, soll als Referenz dienen und zeigt die interne Reflexion.

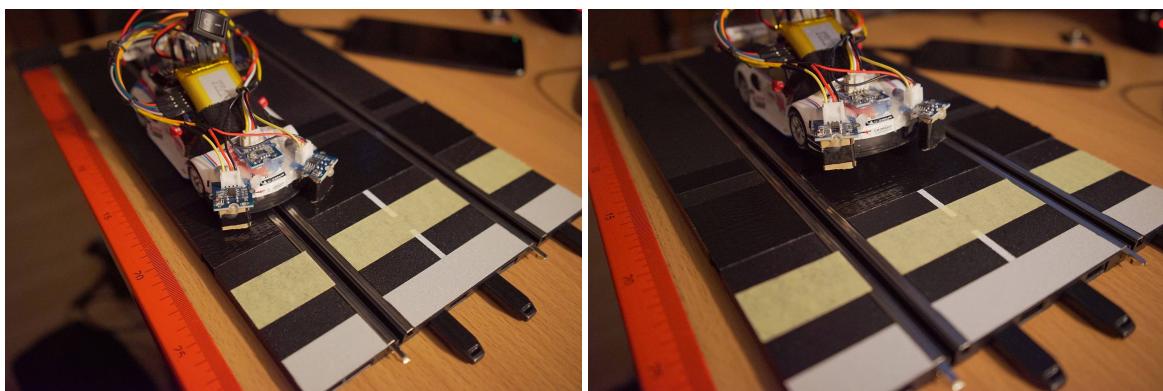


Abbildung 26: Sensoren direkt über der Bahn

Abbildung 27: Sensoren ca. 5mm über der Bahn

Linker Sensor auf der Bahn						
Oberfläche →	Luft	Isoband	Scotch	Krepp	Retroband	D132 Bahn
Signal in V SNR gegen Bahn in dBV	0.50 −6.61	0.98 −0.68	0.90 −1.42	3.08 9.26	3.13 9.40	1.06 0.00
Linker Sensor über der Bahn						
Oberfläche →	Luft	Isoband	Scotch	Krepp	Retroband	D132 Bahn
Signal in V SNR gegen Bahn in dBV	0.50 −1.81	0.60 −0.14	0.59 −0.36	1.33 6.77	3.06 14.01	0.61 0.00

Abbildung 28: Messergebnisse

Da der verbaute Operationsverstärker nicht über eine Hysterese verfügt, sollte das Signal vorgefiltert werden, um hochfrequente Signalanteile zu entfernen. Diese Aufgabe wurde mit einem keramischen 100pF -Kondensator gelöst, der zwischen GND und Signal gelötet ist. Keramische Kondensatoren eignen sich gut zur Entstörung aufgrund ihres geringen Innenwiderstands.

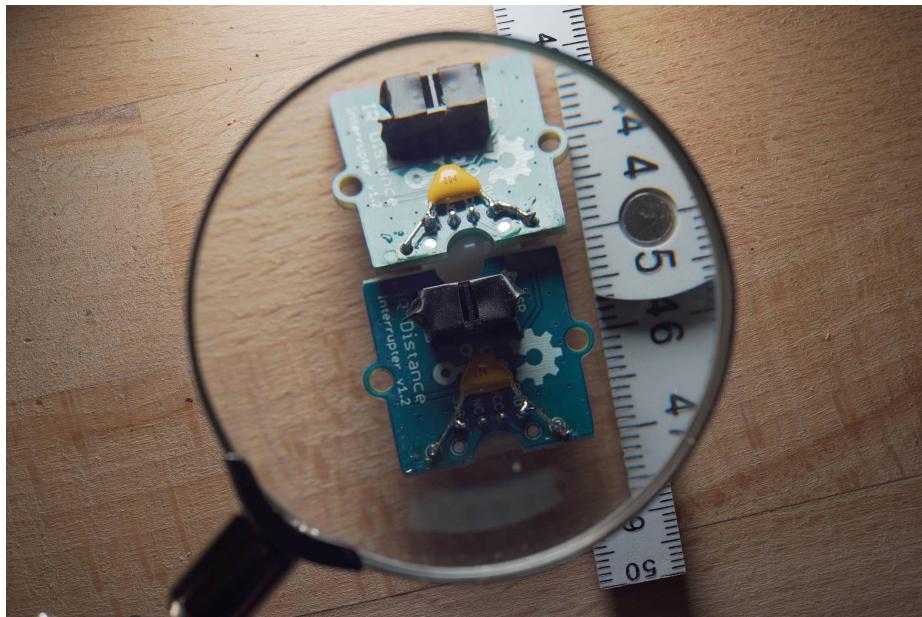


Abbildung 29: Kondensator zwischen Signal und GND

Eine weitere Schwäche dieser Platine ist die geringe Menge an Licht, die bereits ausreicht, um ein Signal zu erzeugen. In der eingesetzten Distanz reicht auch die schwarze, rauhe Fahrbahn, um ein Signal zu erzeugen. So kann nicht zwischen Strecke und Markierung unterschieden werden. Daher wurde ein großer Teil des Fototranistors abgeklebt, um die Menge an einfallendem Licht zu verringern (siehe

Abbildung 30, grün). Um Resistenz gegen Infrarot-Störquellen wie Fernbedienungen, Glühlampen und die Sonne zu aufzubauen, ist eine schwarze Blende aus Karton rund um den Sensor installiert. Abbildung 30 zeigt eine solche Blende (grau) und ihre Effekte. Abseits der direkten internen Reflektionen, die eine solche Konstruktion mit sich bringt, werden auch Reflektionen von der Oberfläche von der Blende „gefangen“, die normalerweise nicht ihren Weg zum Fototransistor finden würden.

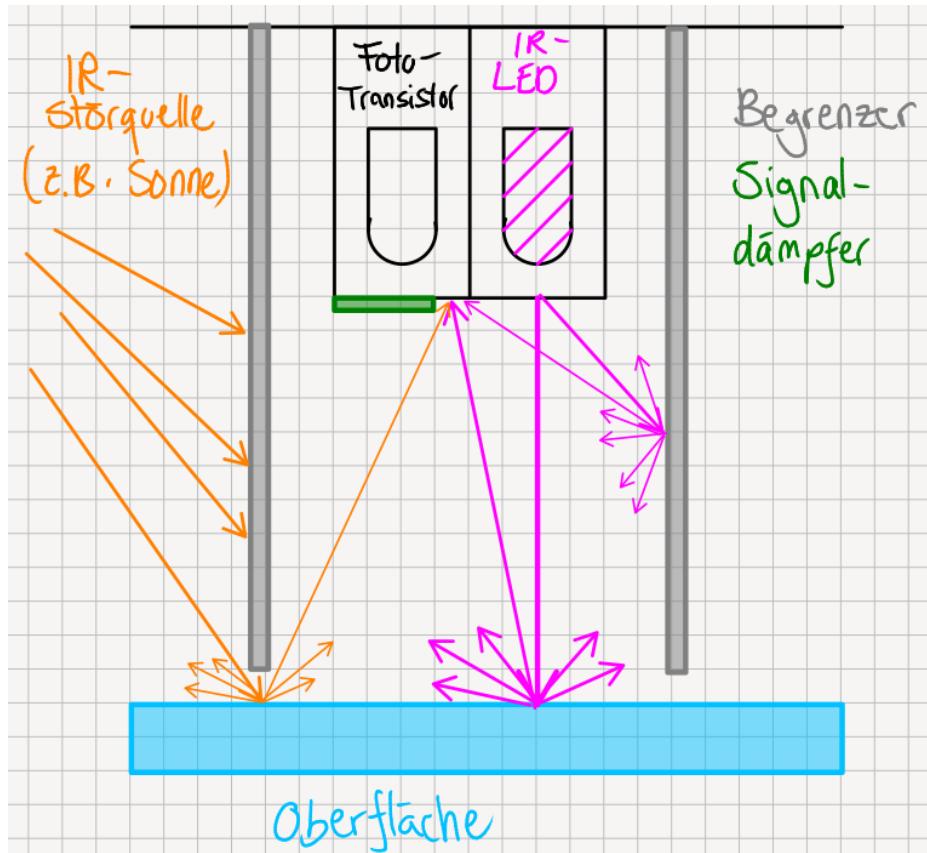


Abbildung 30: Zusätzliche Geometrie zur Verbesserung des Verhaltens

Obwohl das Signal vorgefiltert ist, müssen trotzdem weitere Schritte unternommen werden, um eine möglichst fehlerfreie Funktion zu gewährleisten.

Das Datenblatt des verwendeten Microcontrollers ESP32 [13] sieht vor, dass Pegel von $2.475V \dots 3.6V$ als HIGH bzw. logische 1 interpretiert werden (High-level input voltage). Pegel von $-0.3V \dots 0.825V$ werden als LOW bzw. logische 0 interpretiert (Low-level input voltage). Pegel von $0.825 \dots 2.475V$ sind in der nicht definierten Zone und ändern den Zustand nicht. Dadurch entsteht eine Hysterese, die ein Prellen durch Signalschwankungen um eine Schaltschwelle mit kleiner Amplitude verhindert. Allerdings ist die Implementierung der Flankentrigger bei der Verwendung des ESP32 unintuitiv. Werden Trigger genutzt, die bei Positiver und Negativer Flanke auslösen, so werden auch in der undefinierten Zone Events ausgelöst.

Messungen geben Anlass zur Vermutung, dass für diese Funktion ein Schwellwert bei $\approx V_{CC}/2$ ohne Hysterese verwendet wird. Abbildung 31 zeigt solches Verhalten. Die lila Trace ist ein Signal, das bei jeder erkannten Flanke invertiert. Die gelbe Trace ist das Ausgabesignal des IR-Boards. Die Zeitbasis ist $20\mu s/\text{Division}$, Amplitude ist $1V/\text{Division}$, Nullpunkte der Signale sind mit Markern an der Linken Seite gegeben. Überschwinger und Spitzen in der gelben Trace sind dem Messaufbau geschuldet.

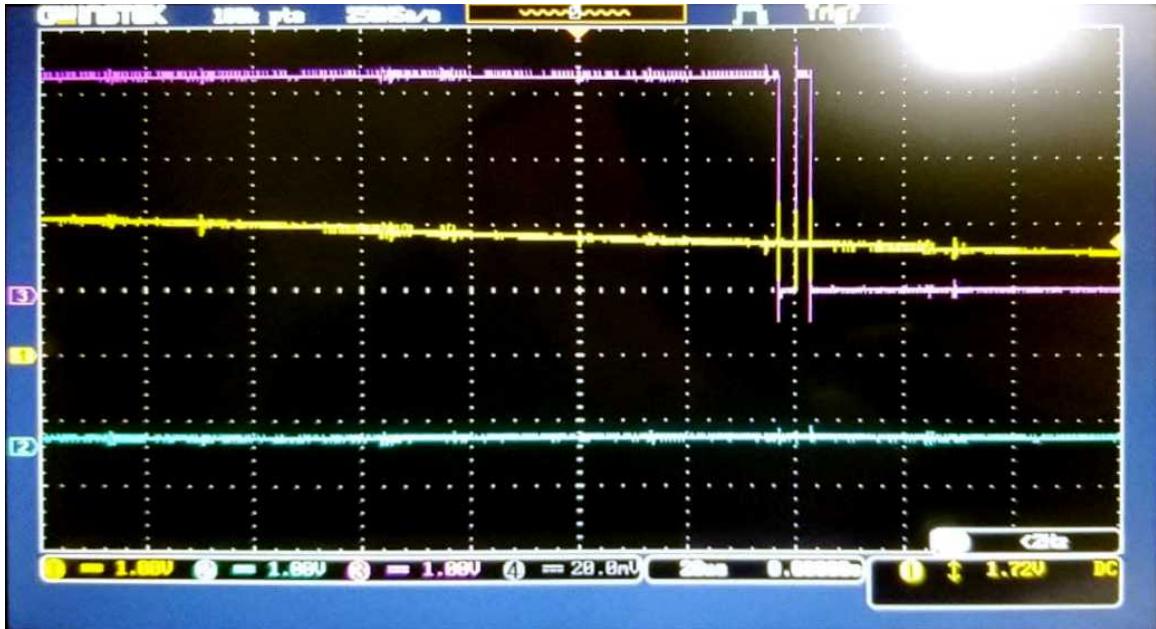


Abbildung 31: Fehlverhalten des ESP32 Flankentrigger

Um dieses Fehlverhalten zu unterbinden, muss entweder in Software oder mit einem Schmitt-Trigger entprellt werden. Entprellung in Software ist in diesem Fall vorzuziehen, um die Komplexität des Systems gering zu halten. Um die Latenz durch Entprellung gering zu halten, wurde der Algorithmus aus Abbildung 32 implementiert. Anstatt bei der ersten Flanke einen Timer zu starten und den Wert nach Ablauf der Zeit auszulesen, wird der Timer bei jeder steigenden Flanke neu gestartet. So muss die Wartezeit t_w nur länger als die längste Zeit zwischen zwei steigenden Flanken während einer Prellung sein. Kurze Prellvorgänge können damit schneller verarbeitet werden.

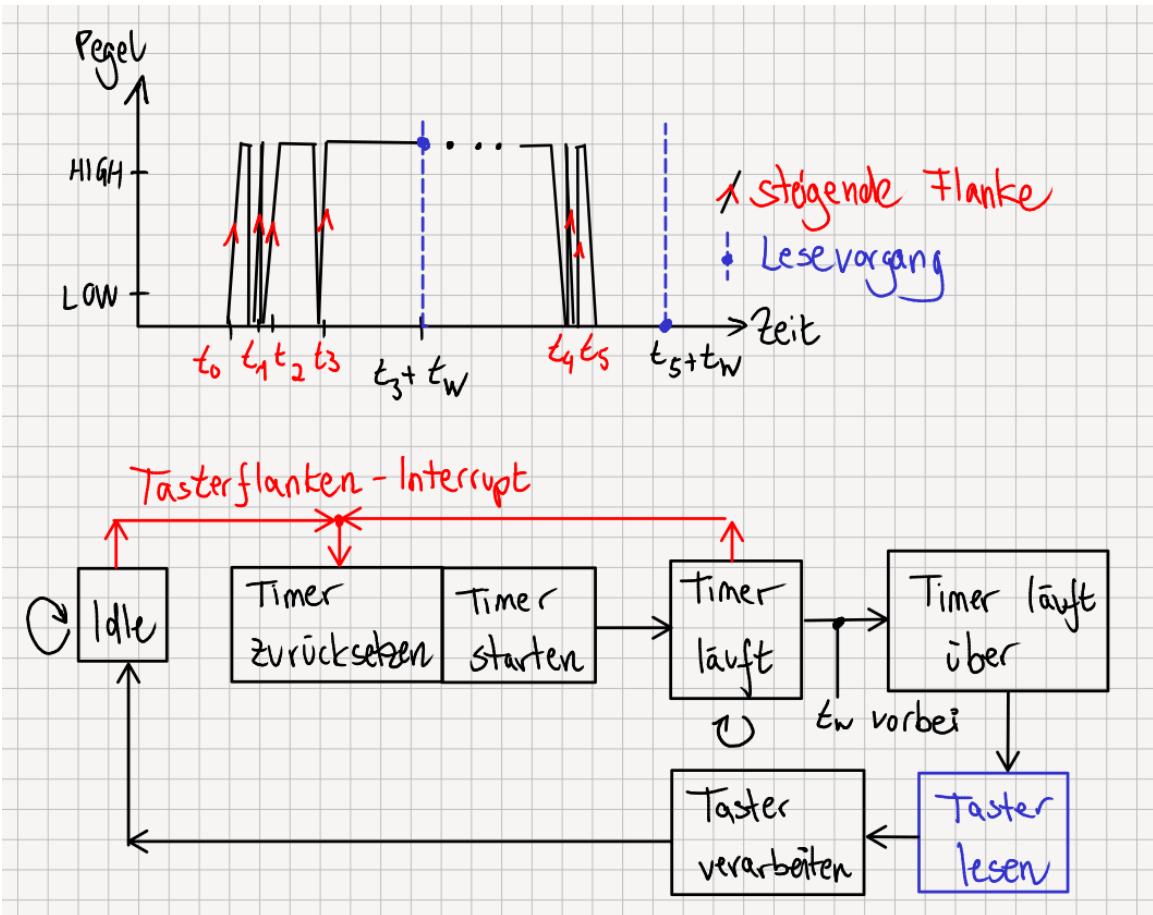


Abbildung 32: Schema des Entprellungsalgorithmus'

6.3 Algorithmus: Wie muss die Rennbahn angesteuert werden?

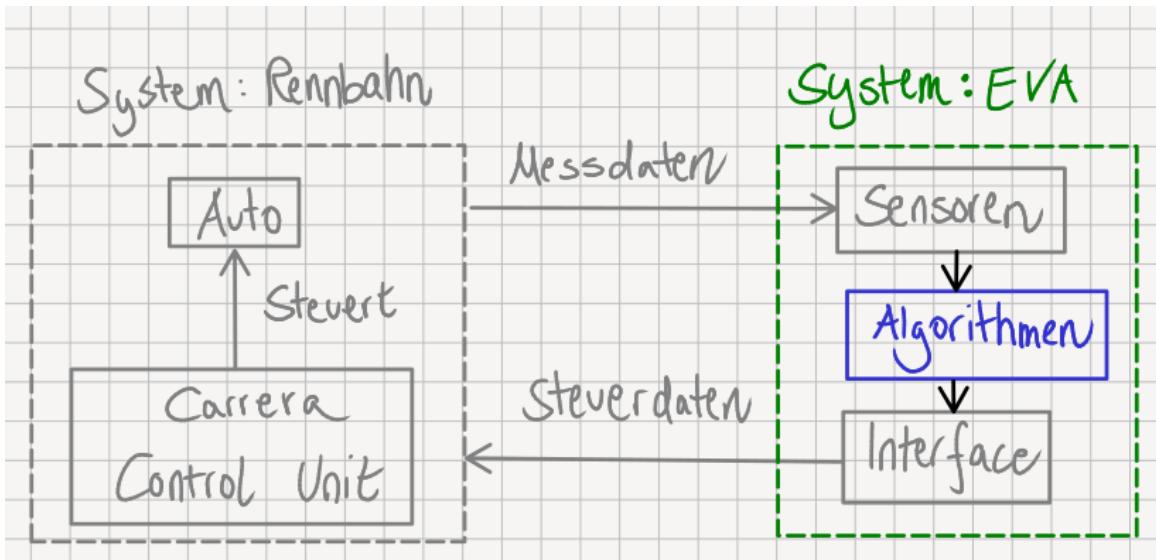


Abbildung 33: Grobe Systemarchitektur mit Fokus auf die Algorithmen zur Steuerung des Autos

Zum Entwurf des Algorithmus' sollten die Zieldefinitionen klar sein. In Sektion 1.1 wurden diese bereits angesprochen. Die Ziele sind

1. Reduzieren der Rundenzeit
 - Fahren maximal möglicher Geschwindigkeit
2. Verhindern von Entgleisung
 - Verringerung der Geschwindigkeit vor Kurven
 - Fahren minimal nötiger Geschwindigkeit
 - Kurven müssen im Vorraus bekannt sein

Die Kurven zwingen das Auto in eine niedrige Geschwindigkeit, um nicht zu entgleisen, die Anforderung an geringe Rundenzeit bedingt einer möglichst hohen Geschwindigkeit. Diese entgegengesetzte Ziele führen zu einem Optimierungsproblem.

Für die Optimierung ist es demnach wichtig, die Geschwindigkeit, das Systemverhalten und die Position des Autos sowie den Streckenverlauf zu kennen. Darüber hinaus stellt sich noch die Frage, wie eine optimale Ansteuerung als Funktion dieser Variablen erfolgen kann.

6.3.1 Streckenverlauf bestimmen

Wie bereits in Sektion 6.2 beschrieben, sind die Reflexionssensoren in der Lage, Flächen unterschiedlicher Reflexionsgrade voneinander zu unterscheiden. Wenn Streckensegmente an ihren Enden mit einer reflektierenden Markierung versehen werden, so kann das Ende eines Streckensegments erkannt werden. In Abhängigkeit des Winkels der Sensoren zur Reflektionsfläche entsteht eine Zeitverschiebung zwischen den Sensorwerten. Diese kann erfasst werden. Wenn die Geschwindigkeit konstant gewählt wird, so ist der Zeitabstand proportional zum Ausfahrtswinkel. Bei Kurven ist der Ausfahrtswinkel größer, je enger die Kurve wird. So können Rechts- und Linkskurven, Innen- und Außenkurven sowie Geraden klassifiziert werden. Abbildung 34 zeigt das Verfahren und gängige Werte.

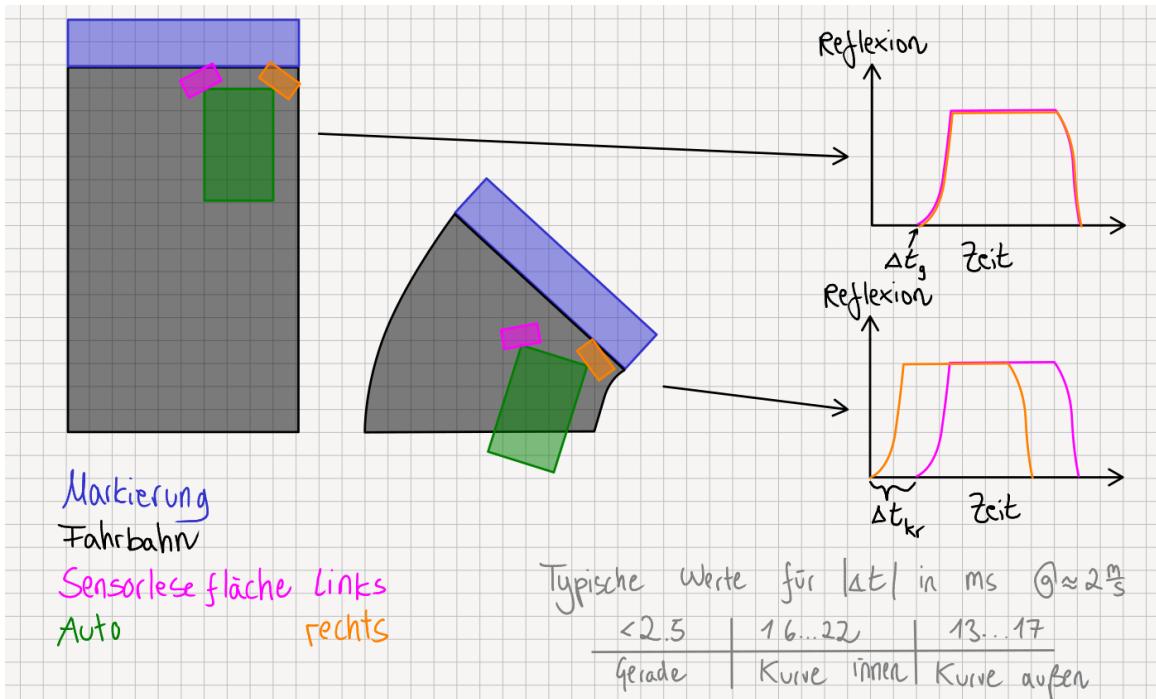


Abbildung 34: Methodik zur Klassifizierung von Streckenteilen

Die Streuung der Werte liegt an der Schwankung der Geschwindigkeit und macht eine Klassifizierung zwischen Innen- und Außenkurven nicht eindeutig. Besonders bei niedrigen Geschwindigkeiten hat die erhöhte Reibung in den Kurven - besonders Innenkurven - einen Einfluss auf die Geschwindigkeit des Autos trotz konstanter Motorspannung. Dazu kommt, dass die Platzierung der Sensoren mit Heißkleber nicht exakt erfolgen kann. Des Weiteren befindet sich bei einem Sensor die LED vorn und der Transistor hinten, bei dem anderen Sensor ist es gerade invers. Deshalb messen die Sensoren auf allen Teilen einen geringen Versatz, der aber durch Kalibrierung behoben werden kann. Dabei wird der Versatz auf einer Geraden gemessen - hier sollte die Zeitdifferenz 0s betragen. Ist die Grundzeitdifferenz ermittelt und die Geschwindigkeit konstant, kann die Grundzeitdifferenz vom gemessenen Wert subtrahiert werden.

Alternativ kann das in den IMUs verbaute Dreiachsengyroskop zum Einsatz kommen. Allerdings ist die Kalibrierung von diesem auf das Koordinatensystem des Autos für akkurate Messungen nicht trivial. Eine weitere Möglichkeit wäre die Erfassung der Zentrifugalkraft, die die Accelerometer messen. Diese ist eine Funktion von Kurvenradius R , Masse des Autos m und Quadrat der Geschwindigkeit v .

$$F_Z(R, m, v) = m \cdot \frac{v^2}{R}$$

Milan Brejl und Jaroslav Němcany haben bereits im Jahr 2008 auf Seitenbeschleu-

nigungen zur Erkennung der Streckengeometrie zurückgegriffen [14].

6.3.2 Segmentweise Position bestimmen

Die Position eines Autos kann mit den Reflexionssensoren zumindest auf einer Segment-zu-Segment-Basis bestimmt werden. Abbildung 35 zeigt das Blockschema für die Segmenterkennung.

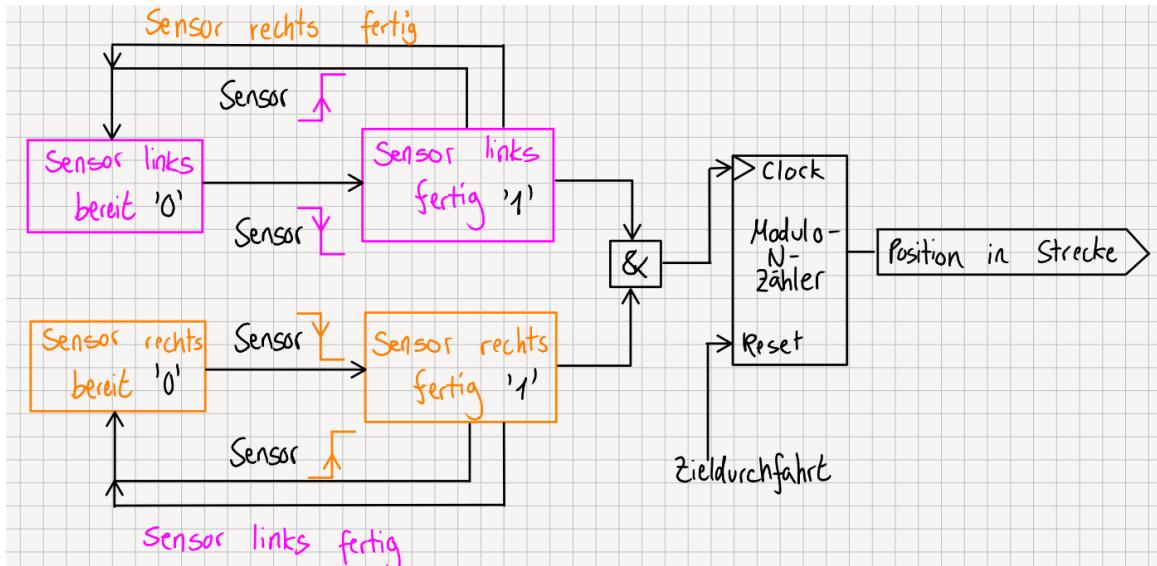


Abbildung 35: Blockschema für Segmenterkennung

Wird eine steigende Flanke gefolgt von einer fallenden Flanke erkannt, so hat ein Sensor die Markierung detektiert. Detektieren beide Sensoren eine Markierung, so wird der Segmentzähler um 1 inkrementiert. Der Segmentzähler besitzt N Zustände, von 0 bis $N - 1$, wobei N die Anzahl Segmente ist, aus denen die Strecke besteht. Es wurden mehrere Methoden zur Unterdrückung von Fehlern implementiert.

1. Der Segmentzähler wird nur inkrementiert, wenn *beide* Sensoren eine Markierung erkannt haben.
2. Eine Erkennung wird zurückgesetzt sobald eine neue steigende Flanke erkannt wird.
3. Wenn das Ziel durchfahren wird, sendet der Controller Emulator eine Nachricht an das Sensorcar, damit der Segmentzähler zurückgesetzt wird. Dieser Reset verhindert kumulative Fehler. Da die Funkübertragung an die in Sektion 6.1 beschriebenen Latenzen gekoppelt ist, kann es passieren, dass der Segmentzähler zu spät zurückgesetzt wird. Zudem basiert die Rundenerkennung auf Polling der Control Unit über deren langsame serielle Schnittstelle.

Sollte die Rundenerkennung akkurater funktionieren, so kann der Klinkenstecker an der Control Unit genutzt werden. Dieser gibt ein direktes digitales Signal bei einer Zieldurchfahrt auf das der Microcontroller mit einem Flankenttrigger reagieren kann. Im Fall einer zu großen Latenz und/oder Autogeschwindigkeit entsteht dann ein absoluter Segmentfehler für eine Runde. Im normalen Betrieb ist die Latenz niedrig genug (ca. 2ms...5ms) und das Auto erreicht keine Geschwindigkeiten, die einen solchen Fehler hervorrufen könnten (Autogeschwindigkeit im Betrieb: maximal ca. $2.5 \frac{m}{s}$), wenn der Klinkenstecker verwendet wird.

6.3.3 Geschwindigkeit mit Accelerometern bestimmen

Bevor die Daten von den Accelerometern genutzt werden können, müssen die Rohwerte normiert und an das Koordinatensystem des Autos angepasst werden. Abbildung 36 zeigt beispielhaft, dass eine Verdrehung der Koordinatensysteme um die z-Achse der Sensoren nur zu einer anteiligen Messung des tatsächlich gesuchten Wertes führt.

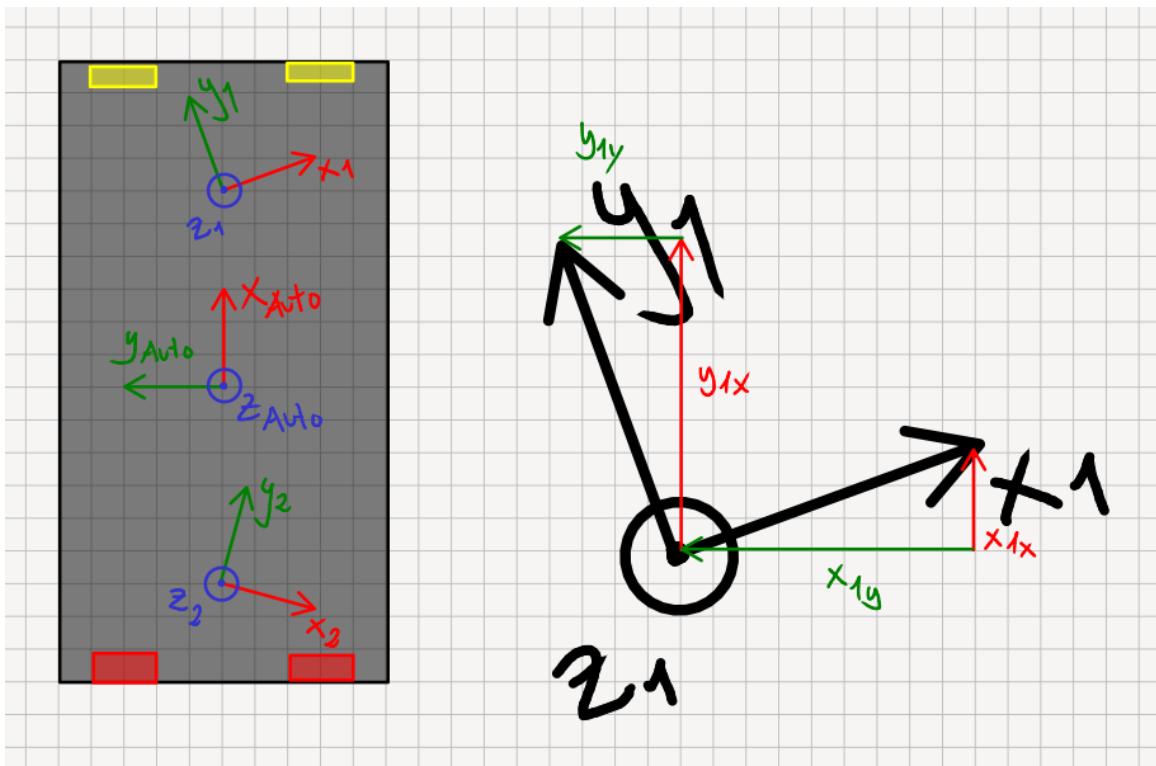


Abbildung 36: Zerlegung der Messachsen in ihre Komponenten

Jede Achse kann als Linearkombination der Messwerte und einem Offset berechnet werden. Um die gemessenen Werte anzupassen, ist die Berechnung von

zwölf Kalibrierungsparametern notwendig.

$$x_{\text{kalibriert}} = x_{\text{gemessen}} \cdot A_{11} + y_{\text{gemessen}} \cdot A_{21} + z_{\text{gemessen}} \cdot A_{31} + A_{41}$$

$$y_{\text{kalibriert}} = x_{\text{gemessen}} \cdot A_{12} + y_{\text{gemessen}} \cdot A_{22} + z_{\text{gemessen}} \cdot A_{32} + A_{42}$$

$$z_{\text{kalibriert}} = x_{\text{gemessen}} \cdot A_{13} + y_{\text{gemessen}} \cdot A_{23} + z_{\text{gemessen}} \cdot A_{33} + A_{43}$$

$$\vec{x}_{\text{kalibriert}} = \underline{A} \cdot \begin{bmatrix} x_{\text{gemessen}} & y_{\text{gemessen}} & z_{\text{gemessen}} & 1 \end{bmatrix}$$

Um die Parameter der Matrix \underline{A} zu bestimmen, müssen zuerst sechs oder mehr Messungen mit bekanntem Zielwert durchgeführt werden. Um die Ausrichtung zu vereinfachen, wird sie so gewählt, dass das *Zielkoordinatensystem* jeweils nur auf einer Achse $+1g$ bzw. $-1g$ messen würde, während die anderen Achsen $0g$ messen sollten. Die Zielwerte sind damit

$$P_1 = \begin{bmatrix} 1g \\ 0g \\ 0g \end{bmatrix}, P_2 = \begin{bmatrix} -1g \\ 0g \\ 0g \end{bmatrix}, P_3 = \begin{bmatrix} 0g \\ 1g \\ 0g \end{bmatrix}, P_4 = \begin{bmatrix} 0g \\ -1g \\ 0g \end{bmatrix}, P_5 = \begin{bmatrix} 0g \\ 0g \\ 1g \end{bmatrix}, P_6 = \begin{bmatrix} 0g \\ 0g \\ -1g \end{bmatrix}$$

Das entspricht einer Ausrichtung in die Richtungen

$$+x, -x, +y, -y, +z, -z$$

des Koordinatensystems auf dem Auto gemäß Abbildung 36. Um die Messungen genau durchzuführen, wurde ein Kalibierungsblock entworfen und gefertigt (siehe Abbildungen 37 und 38). Seine Oberflächen weisen Winkel von 90° zueinander auf. Um das Auto eben am Block befestigen zu können, hat er eine Kerbe, in der die Führung platziert wird.

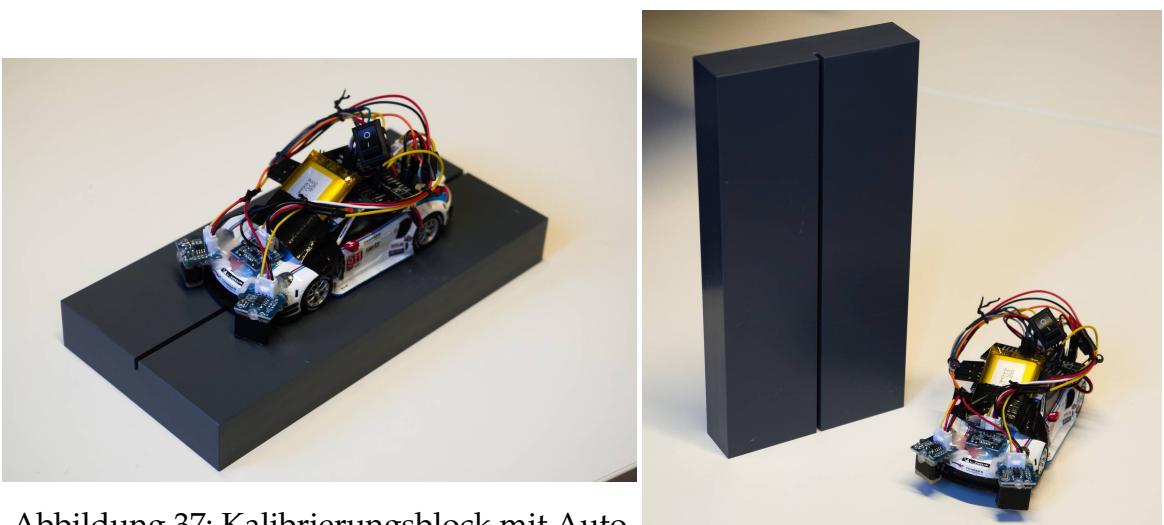


Abbildung 37: Kalibrierungsblock mit Auto darauf in $+z$ -Ausrichtung

Abbildung 38: Kalibrierungsblock in $+x$ - oder $-x$ -Ausrichtung

Das Auto wird also wie in Abbildung 37 gezeigt am Block befestigt. Dann wird der Block nacheinander auf alle sechs seiner Flächen gestellt (die Referenzfläche, auf der er steht, muss parallel zur Erdoberfläche sein) und die Messung wird durchgeführt. Anschließend kann das entstehende Gleichungssystem mit der Methode der geringsten Fehlerquadrate gelöst werden. Der Prozess ist in einer Application Note von ST [15] im Detail ausgeführt.

Während die Koordinatensysteme der IMUs an das Referenzkoordinatensystem des Autos angepasst werden können, besteht dennoch ein gewisser Restfehler.

Zudem besteht ein weiteres Problem mit IMUs: das integrierte Accelerometer misst nicht die Beschleunigung. In Sektion 3 wurde bereits angesprochen, dass Accelerometer tatsächlich die Kraft messen, die auf sie wirkt. Wenn also ein Accelerometer auf einem Tisch liegt, drückt die Erdanziehungskraft es gegen den Tisch. Diese Kraft wird gemessen. Allerdings wird sich der Sensor nicht bewegen, da der Tisch mit der exakt gleichen Kraft gegen den Sensor drückt. Diese Kraft kann allerdings nicht gemessen werden.

Die Fahrbahn auf der das Auto fährt ist niemals exakt parallel zur Erdoberfläche. Aufgrund dieser Tatsache wird immer ein Teil der Erdbeschleunigung mitgemessen. Wenn diese gemessenen Kräfte also ohne Berücksichtigung der Rahmenbedingungen (Fahrbahn, Führung in Schiene) integriert werden, entsteht ein permanenter Fehler. Abbildung 39 zeigt einen möglichen Fehler bei der Mitmessung von einem *konstanten* Offset. Dieser kann entfernt werden. Beispielsweise kann im Ruhezustand eine Messung durchgeführt werden und der gemessene Offset dann von allen weiteren Messwerten subtrahiert werden. Allerdings ändert

sich über die Dauer eines Rennens die Ausrichtung des Autos im Raum und so auch der Offset durch Erdbeschleunigung und Fliehkräfte. Auch eine Filterung im Frequenzbereich (Hochpassfilter) ist ungeeignet, da das Signal und der Offset beide viel zu niederfrequent sind, um effektiv getrennt zu werden. Daher kann der Offset nicht ohne externe Referenz entfernt werden - Accelerometer ohne externe Referenzen sind also für die Bestimmung der Geschwindigkeit eines schienenbasierten Rennautos ungeeignet. Externe Referenzen geben eine Möglichkeit zu einer genaueren Schätzung der Geschwindigkeit, beispielsweise mit Kalman-Filters.

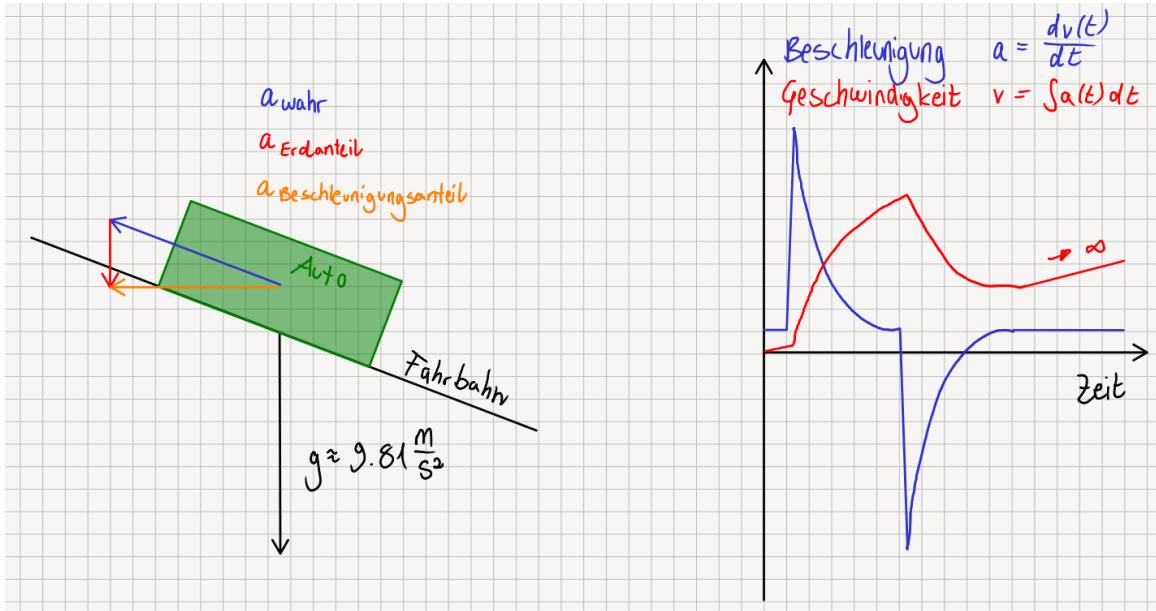


Abbildung 39: Ausrichtungsfehler eines Accelerometers und resultierender Integrationsfehler

6.3.4 Geschwindigkeit mit Reflexionssensoren bestimmen

In Sektion 6.3.3 wurde bereits beschrieben, warum Accelerometer ohne externe Referenzen ungeeignet für die Anwendung als Geschwindigkeitssensor sind. Die Geschwindigkeit ist allerdings ein wichtiger Parameter für alle weiteren Schritte. Daher werden im Folgenden alternative Methoden zur Bestimmung der Geschwindigkeit mit der existierenden Sensorik erläutert.

Die einzigen weiteren Sensoren, über die das Sensorcar verfügt, sind die Reflexionssensoren. Je nach Geschwindigkeit des Autos weist das Signalplateau eine unterschiedliche zeitliche Länge auf. Das liegt daran, dass die Markierung eine feste Breite aufweist (20mm). Es ist bekannt, dass

$$\bar{v} = \frac{\Delta s}{\Delta t}$$

mit \bar{v} als mittlere Geschwindigkeit, Δs als Breite der Markierung und Δt als Passierungsduer. Daher dauert es bei einer niedrigen Geschwindigkeit länger, die Markierung zu passieren. Mit diesem Verfahren könnte zumindest segmentweise die mittlere Geschwindigkeit über der Markierung bestimmt werden.

Allerdings misst der Reflexionssensor nicht die Reflexion in einem Punkt, sondern in einer Fläche. Das hat zur Folge, dass die Markierung bereits erkannt wird, bevor die Sensorfläche ganz von Markierung bedeckt ist. Zudem wird immer noch Markierung erkannt, wenn bereits ein Teil der Sensorfläche wieder von Fahrbahn bedeckt ist. Die erfassten Zeitwerte sind damit zu groß, die berechneten Geschwindigkeiten zu klein. Erschwerend kommt dazu, dass der Winkel von der Infrarot-LED und Fototransistor zu der Oberfläche einen Einfluss auf gemessene Menge Licht hat, da die LED kein paralleles Licht erzeugen kann (siehe Datenblatt des IR-Transceivers IR9909 [16]) und das retroreflektive Klebeband, das als Markierung dient, stark Winkelabhängig reflektiert. Der Sensorwert ist folglich eine Funktion von

- Schnittfläche von Bahn und Markierung mit der Sensorfläche
- Reflektivität von Bahn und Markierung als Funktion des Winkels von IR-LED zur ihnen

Wie in Abbildung 40 gezeigt, hat auch der Eintrittswinkel der Sensorfläche einen Einfluss auf das gemessene Signal, da die Schnittflächen sich verändern. Abbildung 40 zeigt eine Skizze einer Ausgabe der Reflexionssensoren bei Passieren einer Markierung auf der Fahrbahn unter verschiedenen Winkeln.

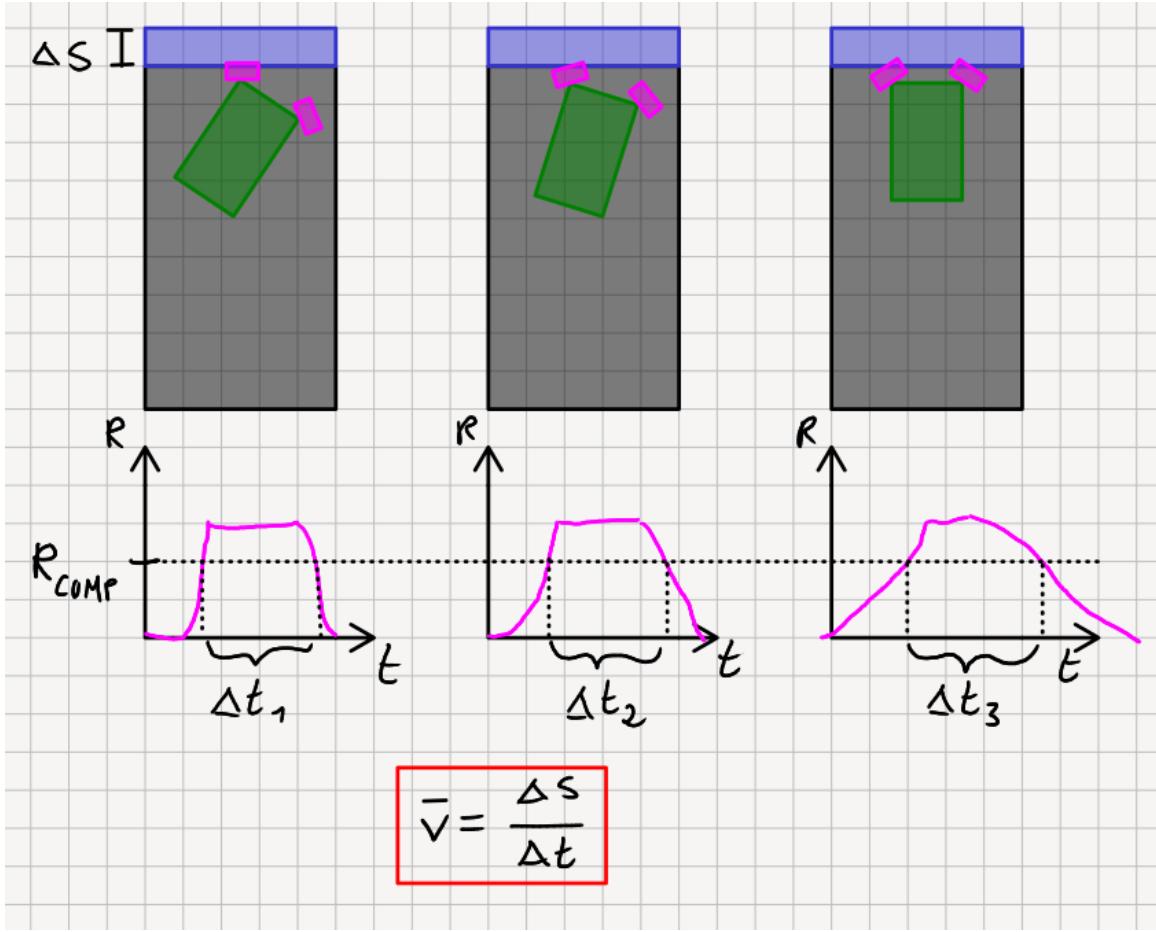


Abbildung 40: Passierung einer Markierung unter diversen Winkeln mit korrespondierenden Wellenformen

Dieser Einfluss könnte auch durch eine kreisförmige Lesefläche verringert werden. Um Lesefehler durch die Lesefläche zu minimieren, muss diese gegen $0mm^2$ gehen. Allerdings verschwindet das Signal auch für kleine Flächen. Es entsteht also ein Optimierungsproblem für die Lesefläche, das diese gegen den minimal tolerierbaren Signalpegel stellt.

Allerdings werden in Kurven immer noch andere Werte gemessen als auf Geraden, da die Markierung nicht an die Kurvenkrümmung angepasst ist. Der Sensor, der die größere Distanz zurücklegt (in der Kurve außen liegt), wird eine höhere Geschwindigkeit besitzen. Um dafür zu kompensieren, müsste die Markierung für Kurven wie in Abbildung 41 an die Kurvenkrümmung angepasst werden.

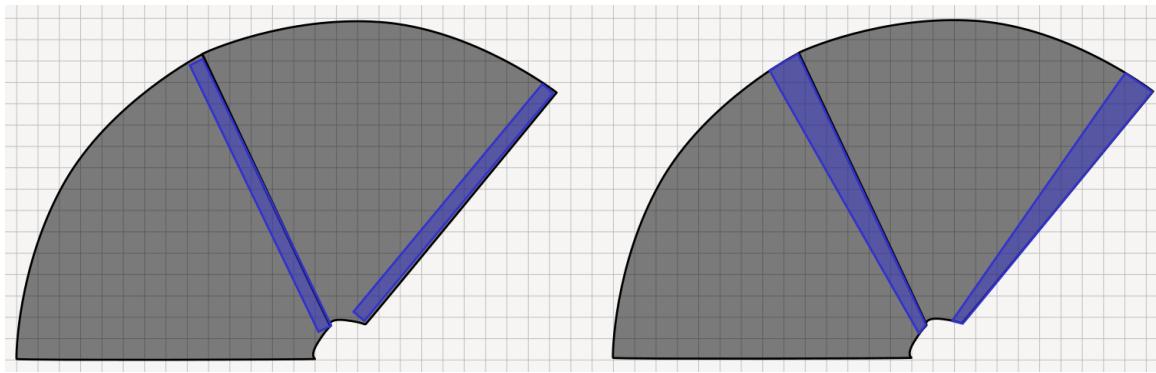


Abbildung 41: Kompensierung für Kurvenkrümmung

Aufgrund dieser Limitierungen ist die Bestimmung der Geschwindigkeit mit den Reflexionssensoren mit der existierenden Implementierung im laufenden Betrieb nicht möglich.

Nichtsdestotrotz gibt es ein Verfahren, um eine akkurate Messung der mittleren Geschwindigkeit zu erhalten: Wird die Auswahl der Streckensegmente auf Geraden beschränkt, so können Fehler durch die Lesefläche ausgeglichen werden. Zudem ist der Eintrittswinkel immer gleich. Es muss auch nicht für Kurvenkrümmung kompensiert werden. Abbildung 42 zeigt die zweite Herangehensweise.

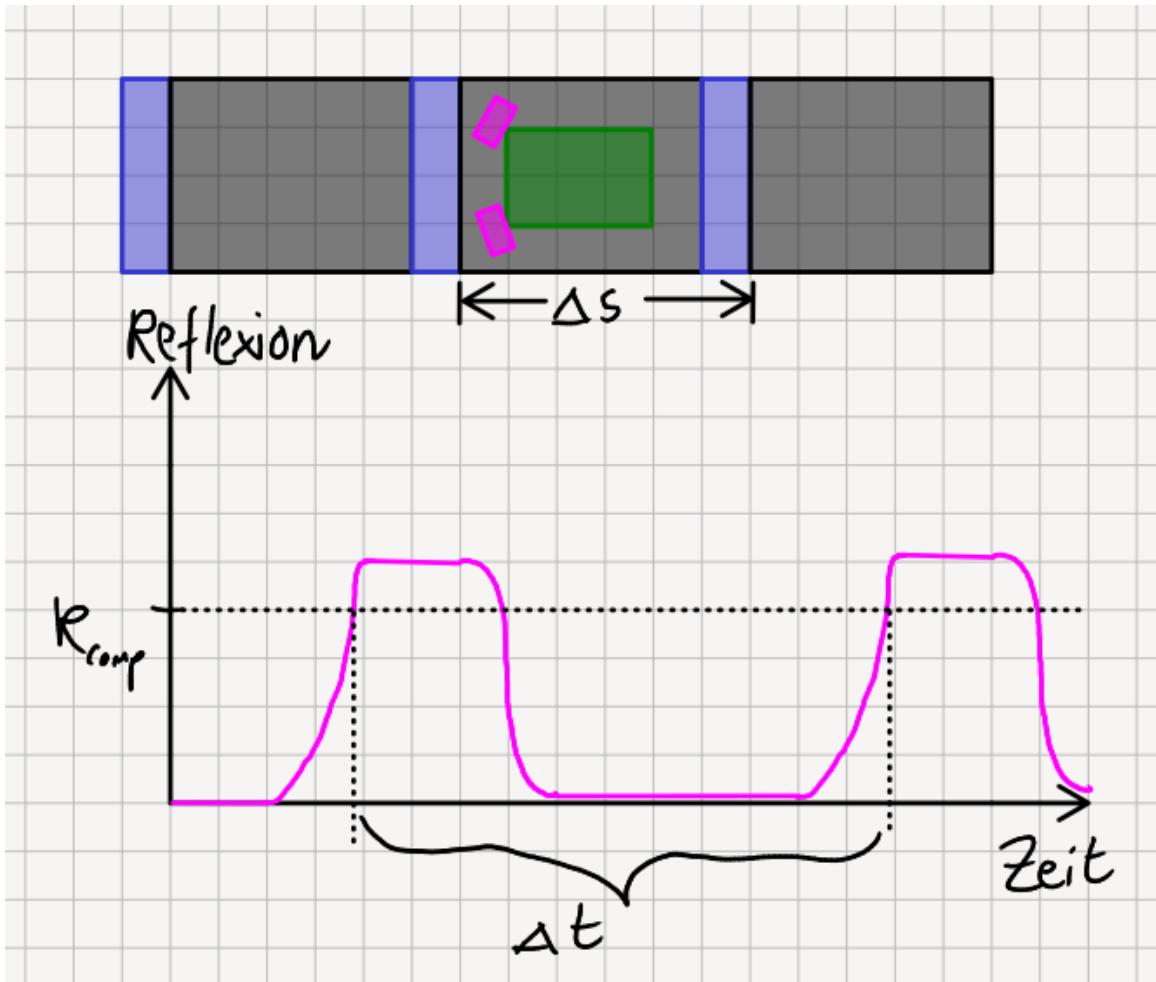


Abbildung 42: Messung der mittleren Geradengeschwindigkeit

Anstatt die Zeit zu bestimmen, die benötigt wird, um die Markierung zu passieren, wird die Zeit zwischen zwei Segmenten gemessen. Dieses Prozedere führt dazu, dass der Fehler durch zu frühe Detektion kompensiert wird, da er zwischen zwei Geradensegmenten immer gleich ist. Hierdurch wird es ermöglicht, die mittlere Geradengeschwindigkeit zu bestimmen. Ein Nachteil, der dadurch entsteht, ist die Mittelung der Geschwindigkeit über einen ausgedehnten Zeitraum. Dadurch ist die Abweichung zur Momentangeschwindigkeit erheblich signifikanter, denn der Differenzenquotient geht nur für kleine Δt in den Differentialquotienten über:

$$\bar{v}(t) = \frac{s(t) - s(t - \Delta t)}{t - (t - \Delta t)} = \frac{\Delta s}{\Delta t}$$

$$v(t) = \lim_{\Delta t \rightarrow 0} \frac{\Delta s}{\Delta t} = \frac{ds}{dt}$$

$$\bar{v}(t) \rightarrow v(t) \text{ für } \Delta t \rightarrow 0$$

6.3.5 Systemverhalten bestimmen: mathematische Modellbildung

Sei es für die Bestimmung von Parametern für Algorithmen oder die Simulation eines Systems zum Algorithmusentwurf - die Modellbildung ist unverzichtbar.

Es gibt verschiedene Verfahren zur Modellgewinnung. Eine analytische Modellgewinnung erfolgt durch Aufstellen einer oder mehreren (Differential-) Gleichungen auf Basis der physikalischen Gegebenheiten und Formeln. Werte für Parameter oder Konstanten, die prozessspezifisch sind, können einem Datenblatt entnommen werden oder müssen experimentell bestimmt werden. Da für viele wichtige Zusammenhänge rund um das System Rennbahn keine Datenblätter vorliegen, führt kein Weg um die experimentelle Bestimmung von Parametern herum. Da die Parameter ohnehin experimentell bestimmt werden müssen, macht es Sinn, die komplette Modellgewinnung experimentell zu realisieren.

Das System Rennbahn kann als dynamisches SISO-Modell (Single Input, Single Output) modelliert werden. Dabei entspricht der Eingang einem 8-Bit Digitalwert und der Ausgang der Bahngeschwindigkeit des Autos (Geschwindigkeit in Fahrtrichtung gemessen tangential zur Bahn). Aufgrund der Wertdiskretisierung für die Motorspannung ist das System stark nichtlinear. Um das System trotzdem linear modellieren zu können, wurde der nichtlineare Zusammenhang zwischen Digitalwert und Geschwindigkeit durch Messungen bestimmt. Durch Messung der Sprungantwort kann PT_1T_t -ähnliches Verhalten beobachtet werden. Die Modellierung mit einem nichtlinearen Teil, einem Totzeitglied und einem Verzögerungsglied erster Ordnung in Serie liegt also nahe. Abbildung 43 zeigt diese mögliche Modellstruktur.

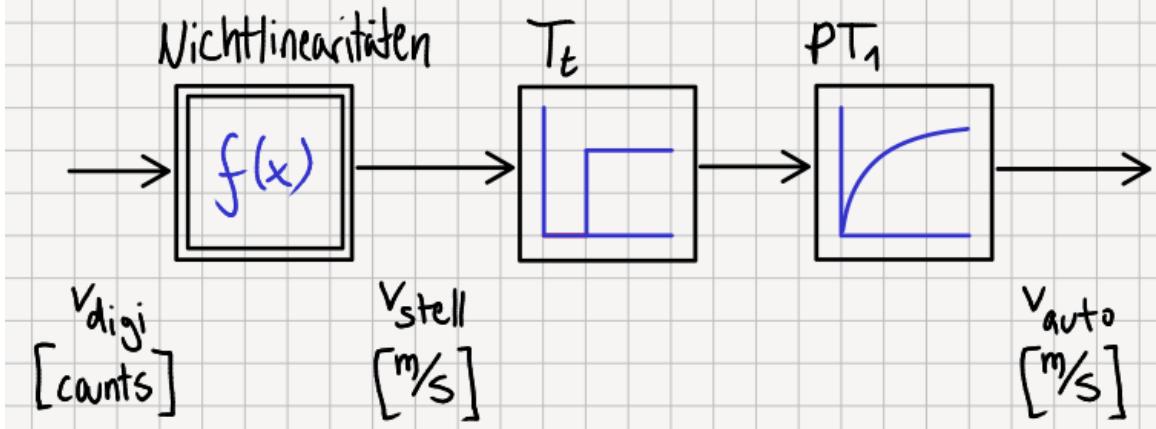
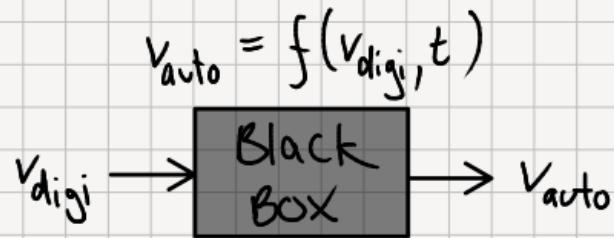


Abbildung 43: Mögliche Modellstruktur für das System Rennbahn

Um die Modellparameter oder -struktur für ein System zu schätzen, haben sich einige Testsignale als hilfreich erwiesen [17]:

- Sprungfunktion $\sigma(t)$ bzw. $\sigma(kT)$
- Impulsfunktion $\delta(t)$ bzw. $\delta(kT)$
- Pulsfolge mit pseudozufälliger Pulsbreite
- Weißes Rauschen

Das weiße Rauschen ist besonders geeignet, um das Frequenzverhalten eines Systems (Übertragungsfunktion) zu bestimmen, allerdings weniger, um dynamisches (=zeitabhängiges) Verhalten zu erfassen. Der Ansatz der Pulsfolge mit pseudozufälliger Pulsbreite ist ähnlich dem mit weißen Rauschen, jedoch ist die Erfassung des dynamischen Verhaltens hier ausgeprägter. Impuls- und Sprungfunktion führen zur Impuls- und Sprungantwort eines Systems, anhand derer gut das dynamische Verhalten ermittelt werden kann. Von allen Testsignalen ist im Rahmen der Steuerung der Rennbahn nach Sektion 5 leider nur die Sprungfunktion realisierbar. Das liegt vor allem an der geringen zeitlichen Auflösung der Stellgröße, aber

auch an der begrenzten Teststreckenlänge. Die Teststreckenlänge limitiert auch die Genauigkeit der Messung von der Sprungantwort.

Der Messaufbau für die Sprungantwort und die Nichtlinearitäten ist nahezu identisch. Die Teststrecke ist in jedem Fall eine Bahn, die nur aus Geraden besteht (siehe Abbildung 44). Um Konsistenz zwischen Testdurchläufen sicherzustellen, ist ein Stopper auf der Bahn installiert, der die Position des Autos zu Beginn des Tests gleich hält (siehe Abbildung 45).

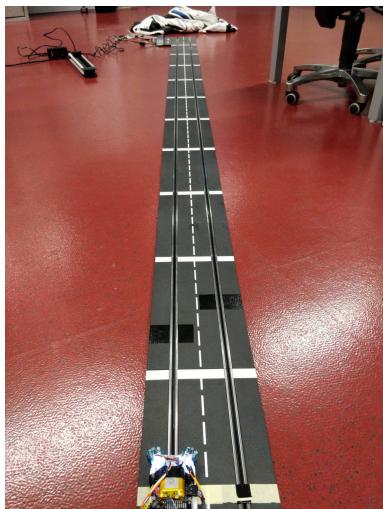


Abbildung 44: Testbahn mit Sensorauto

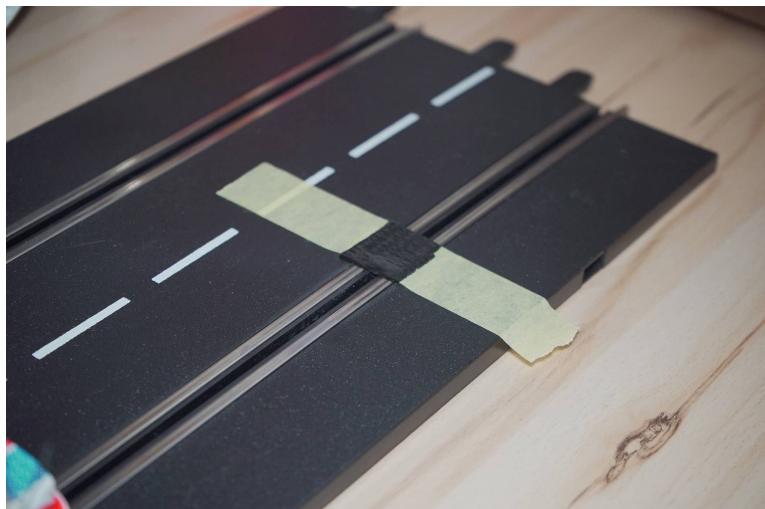


Abbildung 45: Stopper aus Tape, Startpunkt der Messung für Konsistenz

Für die Bestimmung des nichtlinearen Teils wird ein 8-Bit Digitalwert angelegt. Das Auto reagiert auf diesen Pegel: Es passiert die Streckenmarkierungen. Dabei wird die mittlere Geradengeschwindigkeit wie in Sektion 6.3.4 beschrieben gemessen. Die erreichte Endgeschwindigkeit wird als $h(t \rightarrow \infty)$ interpretiert, also als Endwert der Sprungantwort. Dieser Ansatz wird genauer, je länger die Teststrecke ist. Abbildung 46 zeigt die Ergebnisse dieser Tests. Es wurden Werte im Abstand von 3 counts angelegt, nach 110 counts wurde der Test beendet. Es sind die diskreten Spannungsspeicher, die am Motor anliegen, im Verlauf erkennbar: die Endgeschwindigkeit weist auch diskrete Pegel auf. Nach Sektion 4 werden 16 Stufen erwartet. Allerdings reichen die unteren 3 Pegel nicht, um die Haftreibung zu überwinden.

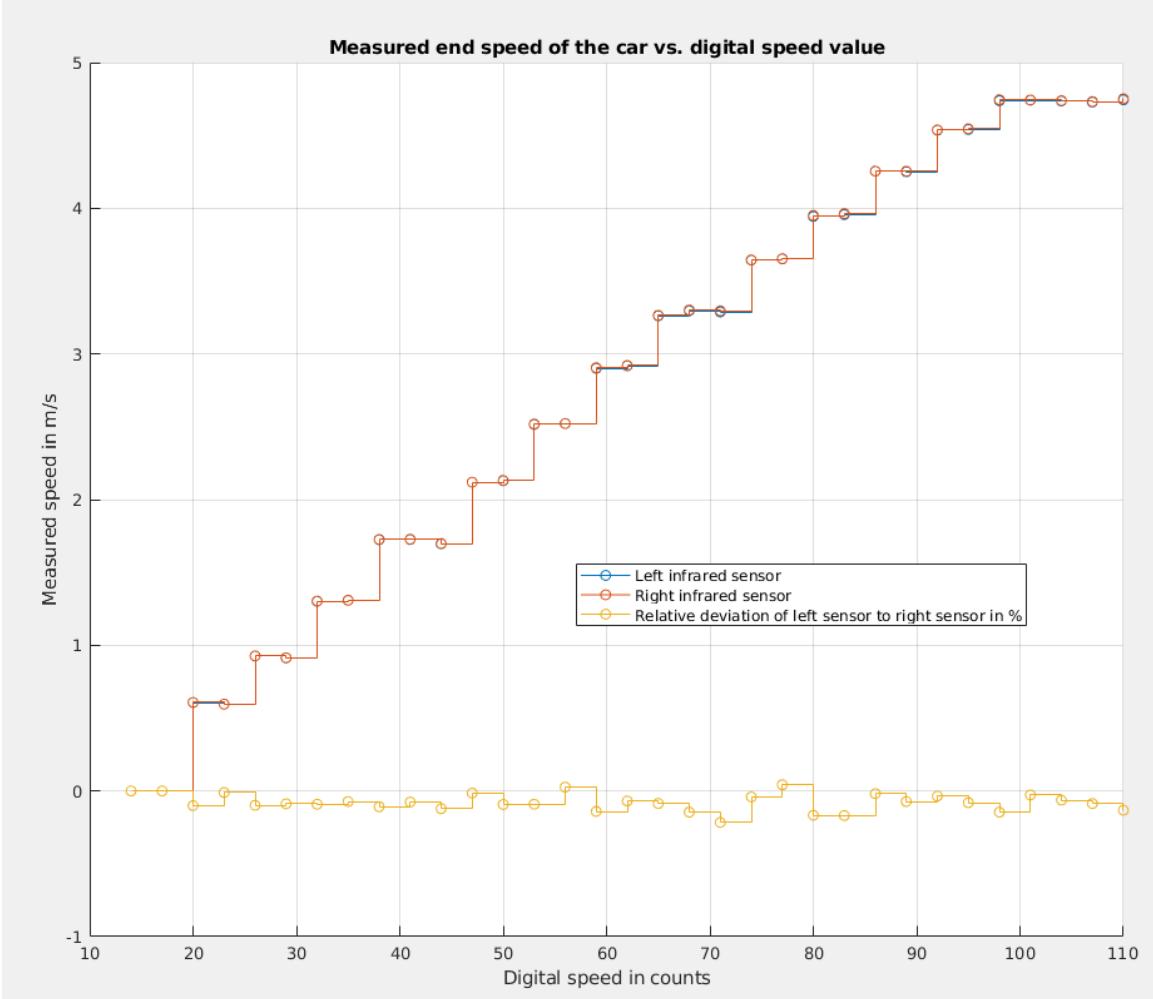


Abbildung 46: Gemessener Zusammenhang zwischen v_{digi} und gemessener Geschwindigkeit \bar{v}

Für die Bestimmung der Totzeit T_t wird ein hoher Digitalwert für $75ms$ angelegt und dann wieder entfernt. Die Zeit von Senden des Digitalwerts bis zu einer Messung von Beschleunigung mit den Accelerometern kann dann bestimmt werden. Messwerte zeigen Ergebnisse von $30ms \dots 150ms$ bei guter SNR. Die Totzeit ist eine Konstante, sobald CU und Sensorcar eingeschaltet sind, da die Desynchronisierung der $75ms$ -Takte gleich bleibt. Dieser Zusammenhang gilt aber auch nur, wenn die SNR gut genug ist, um niedrige Latenzen zu erzielen. Ist die Drahtloslatenz zu groß, so steigt die Wahrscheinlichkeit, eine Taktflanke zu verfehlten.

Um die Zeitkonstante T_1 zu schätzen, werden die Messwerte an jedem Geradenstück relevant. Mit manueller Parametermanipulation wurde eine Zeitkonstante von $400ms$ ermittelt.

Durch Limitierungen in Sensorik, Testaufbau und der gewählten Modellierung werden viele relevante Parameter nicht erfasst oder fehlerhaft gemessen:

- Die gemessene Geschwindigkeit wird als Momentangeschwindigkeit inter-

pretiert, ist aber die mittlere Geradengeschwindigkeit.

- Der Motor weist eine starke Temperaturabhängigkeit auf, die nicht gemessen oder berücksichtigt wird.
- Die Teststrecke enthält nur Geraden. Die Reibung in den Kurven ist erheblich höher als auf Geraden, was zu einer merklichen Verzögerung des Autos führt.
- Die Tests werden aus dem Stand durchgeführt, das Auto ist im Betrieb aber immer schon in Bewegung.
- Je weiter das Auto von der CU entfernt ist und je größer der Motorstrom, desto mehr Spannungsabfall erfährt es durch den steigenden Leitungswiderstand.
- Der Abrieb und die Geometrie der Kontaktschleifer hat auch einen Einfluss auf die Stromversorgung und Reibung des Autos.

6.3.6 Algorithmen für die Ansteuerung der Rennstrecke

PID-Regelung

Üblicherweise werden für die autonome Steuerung mechanischer Systeme in einen Arbeitspunkt Regler verwendet. Ein Regelalgorithmus arbeitet in erster Linie mit einem Soll-Istwert-Vergleich, wobei der Sollwert vom Nutzer vorgegeben und der Istwert von Sensorik gemessen wird. Abbildung 47 zeigt den Standard-Regelkreis, der die Funktionsweise eines Reglers gut beschreibt.

Einer Regelstrecke, das ist im Fall von Projekt EVA das System der Rennbahn, wird ein Regler vorgeschaltet. Dieser erhält als Eingang die Differenz von Soll- und Istwert, das Fehlersignal. Je nach Typ Regler wird dieses Fehlersignal unterschiedlich verarbeitet, um eine Stellgröße zu erzeugen. Stellgrößen sind Eingänge vom zu regelnden System, auf die mit einem Stellglied (z.B. einem Handregler) Einfluss genommen werden kann.

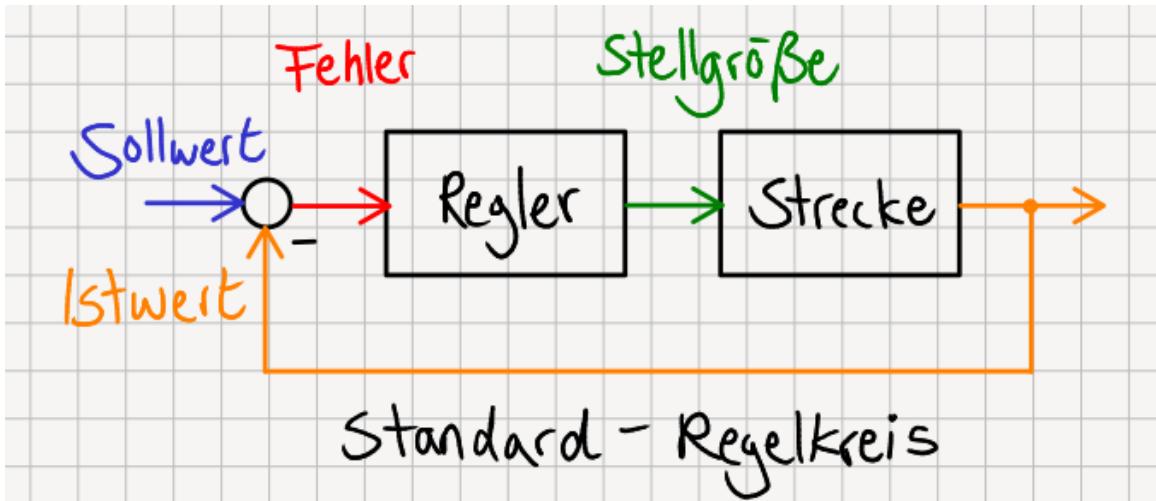


Abbildung 47: Standard-Regelkreis

Die parallele Form des Proportional-Integral-Differential-Reglers (PID-Regler) berechnet die Stellgröße wie folgt:

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau) d\tau + T_d \cdot \frac{de(t)}{dt}$$

Mit Stellgröße $u(t)$, Fehlersignal $e(t)$ sowie Proportional-, Integral- und Differentiationskonstanten K_p , K_i und K_d . Der Proportionalanteil gewährt Sprungfähigkeit, der Differentialanteil beschleunigt die Antwort und der Integralanteil hat die Aufgabe, die Regeldifferenz für zunehmende Zeit zu echt Null zu bringen. Die drei Konstanten sind die Parameter des PID-Algorithmus und werden nach verschiedenen Verfahren gewählt. Relevant hierbei ist zum einen, wie das System auf Eingabe reagiert und zum anderen, welches Ziel verfolgt werden soll. Der Regler wird dann nach Rahmenbedingungen wie der maximalen Überschwingweite oder Dämpfung eingestellt. Abbildung 48 zeigt eine mögliche Wellenform der Sprungantwort eines Systems mit vorgeschaltetem PID-Regler. Dieser ist so eingestellt, dass Überschwinger zugelassen werden. Der Regler kann aber auch so eingestellt werden, dass ein solcher Sprung nicht zu einer Schwingung führt, allerdings dann auf Kosten der Reaktionsgeschwindigkeit des Gesamtsystems.

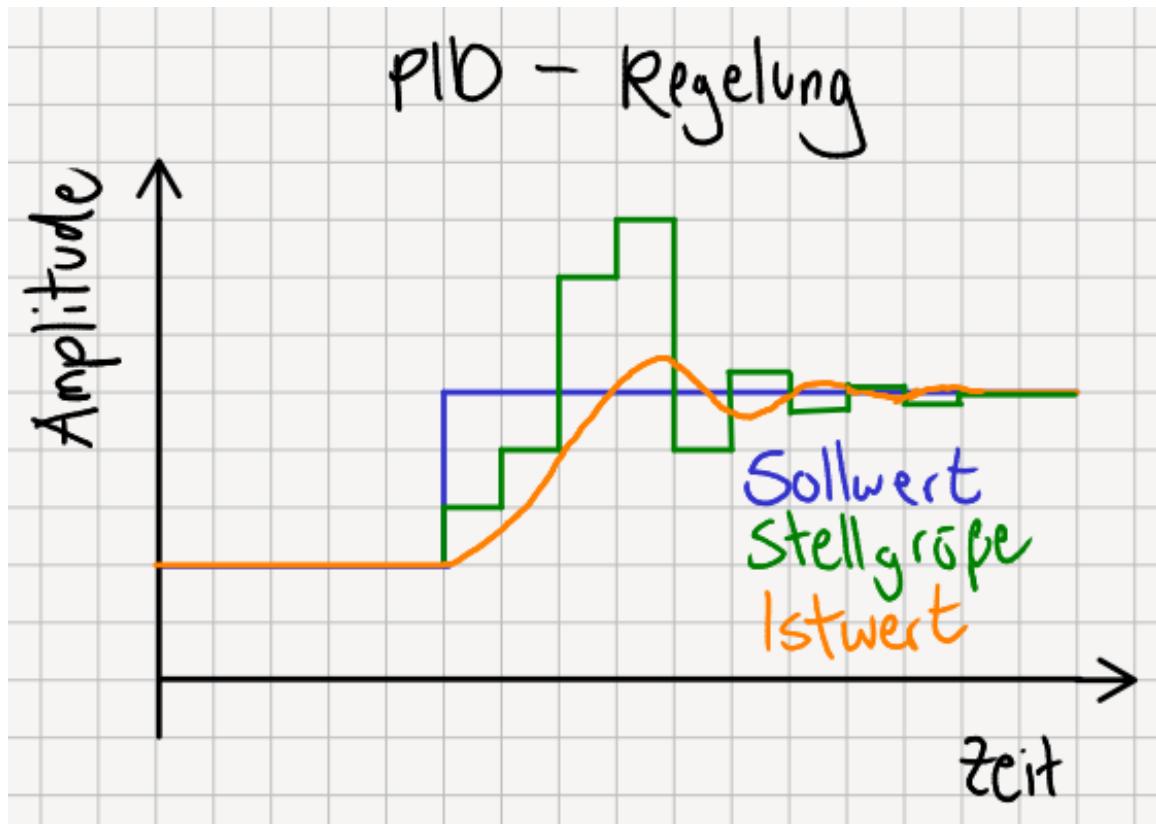


Abbildung 48: Beispiel einer Wellenform bei PID-Regelung

Die optimale Wahl der Parameter ist ein komplexes Problem. Abseits der gängigen Methoden und Faustregeln behandelt „Evolutionary Algorithms and Reinforcement Learning in Experiments with Slot Cars“ [18] daher das Finden optimaler Parameter mithilfe neuroevolutionärer Algorithmen.

Prädiktive Regelung

Wie Abbildung 48 zu sehen ist, greift ein PID-Regler erst ein, wenn bereits ein Fehler vorliegt, also der Istwert nicht dem Sollwert entspricht. In manchen Fällen ist der Fehler dann aber schon zu groß, um mit dem verfügbaren Stellglied korrigiert werden zu können. Hierbei muss also der Fehler bereits verhindert werden, bevor er passiert - es muss ein Blick in die Zukunft geworfen werden. Der modellprädiktive Regler (Model Predictive Controller, MPC) ist ein sinnvoller Ansatz, um dieses Problem zu lösen. Ist bekannt, welchen Ausgang das System in der Zukunft haben soll, so kann bestimmt werden, welche Folge von Stellgrößen diesen Ausgang möglichst gut erreicht. Um dies zu realisieren, nutzt der MPC ein internes Modell der Regelstrecke. Um die Effizienz zu erhöhen, ist diesem meist noch ein PID-Regler vorgeschaltet, das Modell beschreibt dann das Verhalten des bereits vorgeregelten Systems. Abbildung 49 zeigt eine mögliche Wellenform der Sprungantwort eines Systems mit modellprädiktiver Regelung.

Modellprädiktive Regelung (MPC)

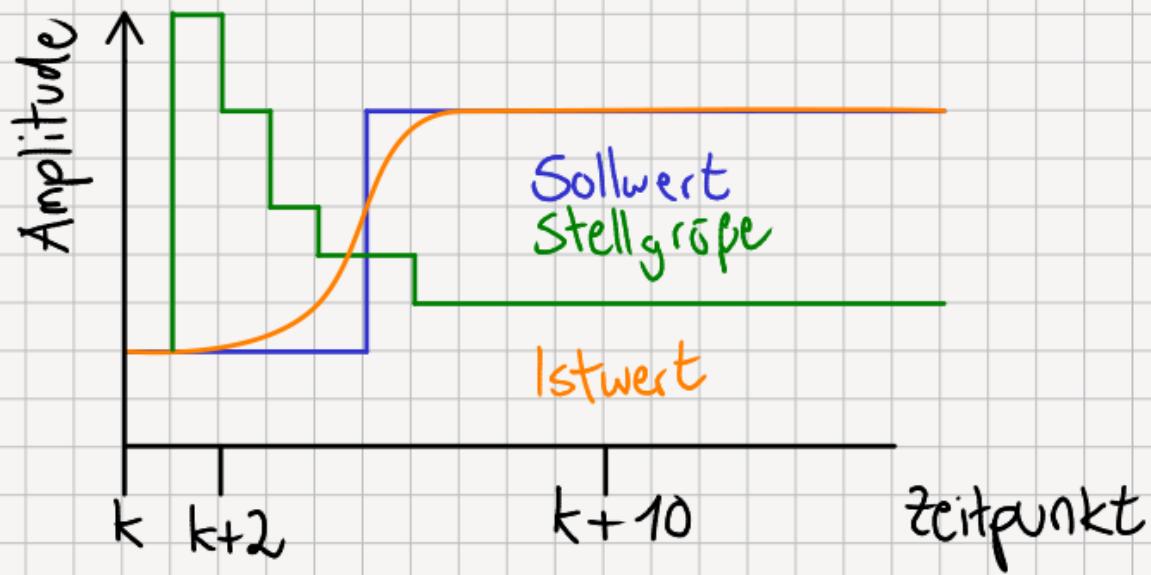


Abbildung 49: Beispiel einer Wellenform bei modellprädiktiver Regelung

Je nach System ist es nötig, unterschiedlich weit in die Zukunft zu schauen. Mit jedem vorhergesagten Schritt erhöht sich auch die Menge möglichen Stellgrößen und damit der Rechenaufwand erheblich. Daher macht es mitunter Sinn, die Stellgrößen nicht „online“, also im laufenden Betrieb, zu berechnen, sondern offline – also im Vorraus. Die Ergebnisse sind allerdings erheblich abhängiger von der Qualität des verwendeten Modells.

Abseits der modellprädiktiven Regelung gibt es noch weitere Verfahren, optimale Kontrolle zu realisieren. „Optimal control of a slot car racer“ [19] sucht die Lösung für das Problem unter Verwendung von analytischen Modellen offline.

Der Bedarf von genauem Istwert und Modell sowie eine ausreichende Stellgrößenrate kann von der gegebenen Hardware allerdings nicht erfüllt werden. Es muss also ein Algorithmus entworfen werden, der mit den verfügbaren Sensorwerten arbeiten kann. Die einzigen verlässlichen Sensorwerte, die zur Verfügung stehen, sind

- Die segmentweise Position des Autos.
- Der Verlauf der Strecke.

Darum ist es sinnig, segmentbasierte Algorithmen zu implementieren. Jeder Segmentgeometrie kann eine Zielgeschwindigkeit zugeordnet werden (Zielgeschwindigkeit für Kurve, Zielgeschwindigkeit für Gerade). Dann kann in Abhängigkeit

der nächsten N Segmente eine der 16 möglichen Stellgrößen gewählt werden. Um die Anzahl Parameter gering zu halten, kann die Gewichtung für alle N Segmente gleich gewählt werden. Die freien Parameter sind damit nur noch

- Zielgeschwindigkeit für Kurven
- Zielgeschwindigkeit für Geraden
- N : Anzahl der nächsten Segmente, die berücksichtigt werden sollen

Die Idee hinter diesem Ansatz ist, dass die Geschwindigkeit auf langen Geraden maximiert und bei langen Kurven minimiert wird, während eine Mischung von Geraden und Kurven in den nächsten N Segmenten zu einer Verringerung der Geschwindigkeit führt. So wird vor Kurven verzögert und aus Kurven heraus beschleunigt.

Aber was sind die optimalen Werte für eine beliebige Strecke? Um den optimalen Parametern näher zu kommen, kann eine Simulation durchgeführt werden. Dieses Vorgehen schränkt den Suchbereich von Parametern am realen System ein und spart so sehr viel Zeit.

6.3.7 Simulation des Systems aus EVA und der Rennstrecke

Der gefundene Algorithmus muss auf allen Strecken funktionieren. Um die Simulation zu beschleunigen, wurden drei Streckenverläufe gewählt, die repräsentativ für verschiedene Strecken stehen und auch mit den verfügbaren Teilen als reale Strecke aufgebaut werden können. Diese Testtracks sind in Abbildung 50 dargestellt. Sie bestehen alle aus sechs 60° -Kurven und sechs Geraden in unterschiedlicher Anordnung. Die Gesamtlänge aller Testtracks ist damit gleich, lediglich der Streckenverlauf ist unterschiedlich.

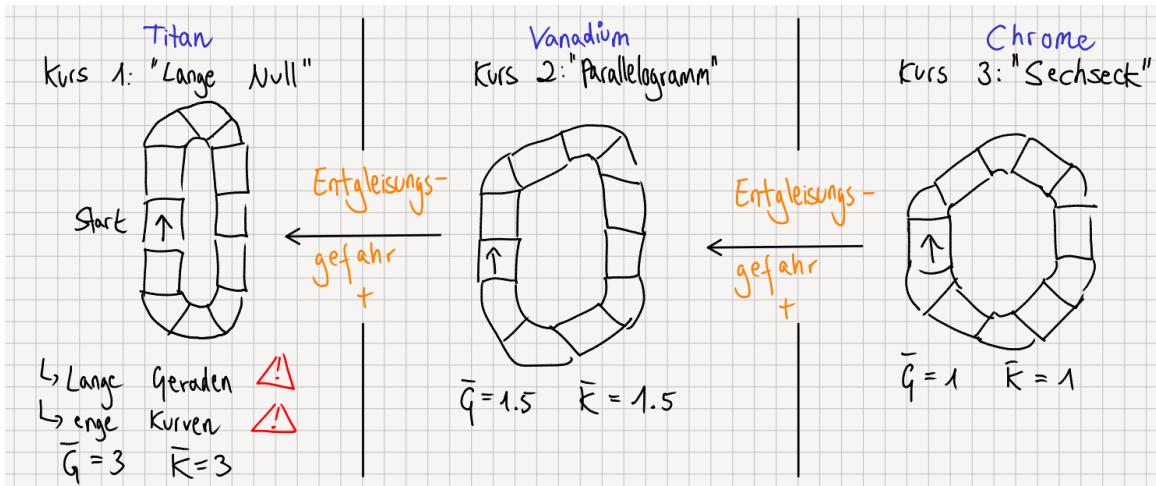


Abbildung 50: Testtracks „Titan“ , „Vanadium“ und „Chrome“ mit mittlerer Geradenlänge \bar{G} und mittlerer Kurvenlänge \bar{K}

Um die zulässige maximale Kurvengeschwindigkeit zu bestimmen, wurde ein Testaufbau wie in Abbildung 51 realisiert. Das Auto wird auf einer langen Gerade in verschiedenen Durchläufen immer mehr beschleunigt, die Kurveneintrittsgeschwindigkeit wird gemessen. Die Messung wird beendet, sobald das Auto entgleist. Die letzte gemessene Kurveneintrittsgeschwindigkeit ist somit die maximale Kurvengeschwindigkeit.



Abbildung 51: Messaufbau für maximale Eintrittsgeschwindigkeit in Kurve

Die Simulation erfolgt nach dem in Abbildung 52 skizzierten Schema.

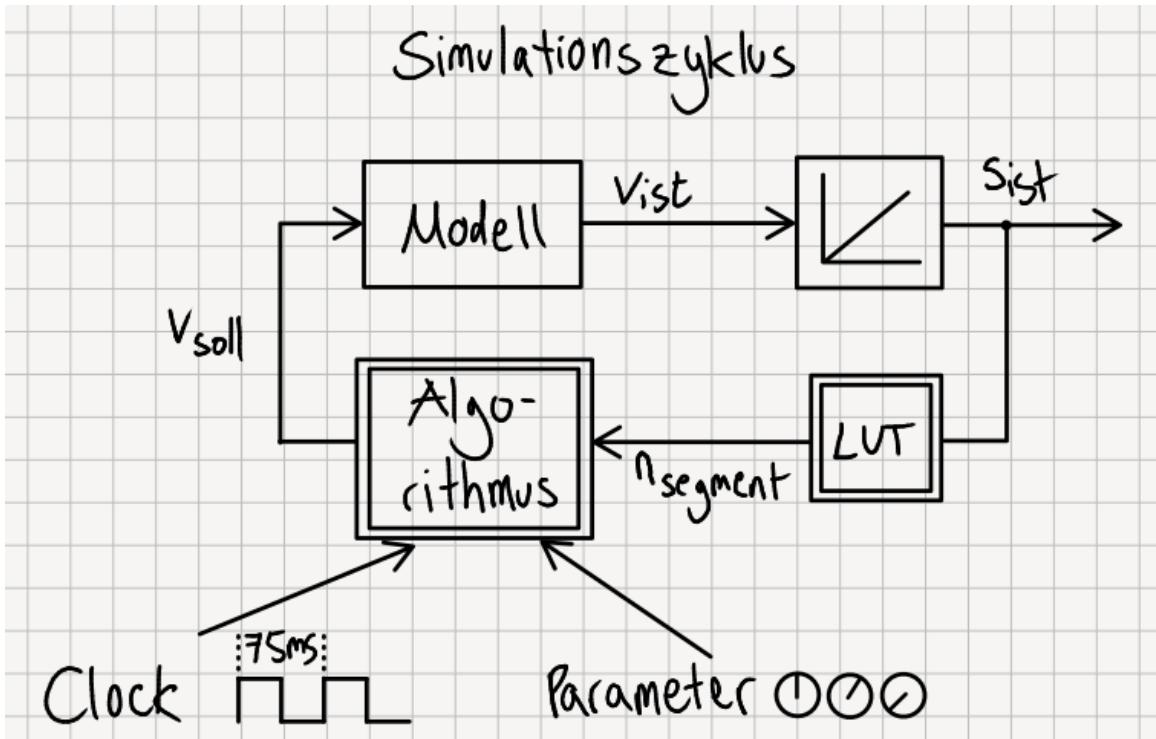


Abbildung 52: Simulationszyklus

Wird die maximale Kurvengeschwindigkeit um ein relevantes Maß überschritten oder die vorgegebene Anzahl an Runden wurde zurückgelegt, so wird die Simulation beendet. Die Simulation erfolgt zeitdiskret in einstellbaren Zeitschritten. Die in diesem Dokument aufgezeigten Ergebnisse wurden mit einer Simulation mit Zeitschritt $1ms$ erreicht. Für das Modell der Strecke wurde das in Sektion 6.3.5 angesprochene grobe PT_1T_t -Modell gewählt. Es wird der zeitliche Verlauf der Autogeschwindigkeit v_{ist} und der Autoposition $n_{Segment}$ unter den Begrenzungen des Systems simuliert (siehe Abbildung 53). Zusätzlich sind die Stellgröße, Stellgrößentakt und die maximale Geschwindigkeit zu sehen. Abseits dieser Werte werden die mittlere Rundenzeit μ_R und Standardabweichung σ_R aller fliegenden Runden erfasst. Die erste Runde wird nicht erfasst, da das Auto hier aus dem Stand beschleunigen muss. Die Simulation verfügt über die Möglichkeit, eine Rastersuche der Parameter durchzuführen. Ist das Optimum gefunden, so werden der zeitliche Verlauf und die statistischen Daten für diese Parameter angezeigt.

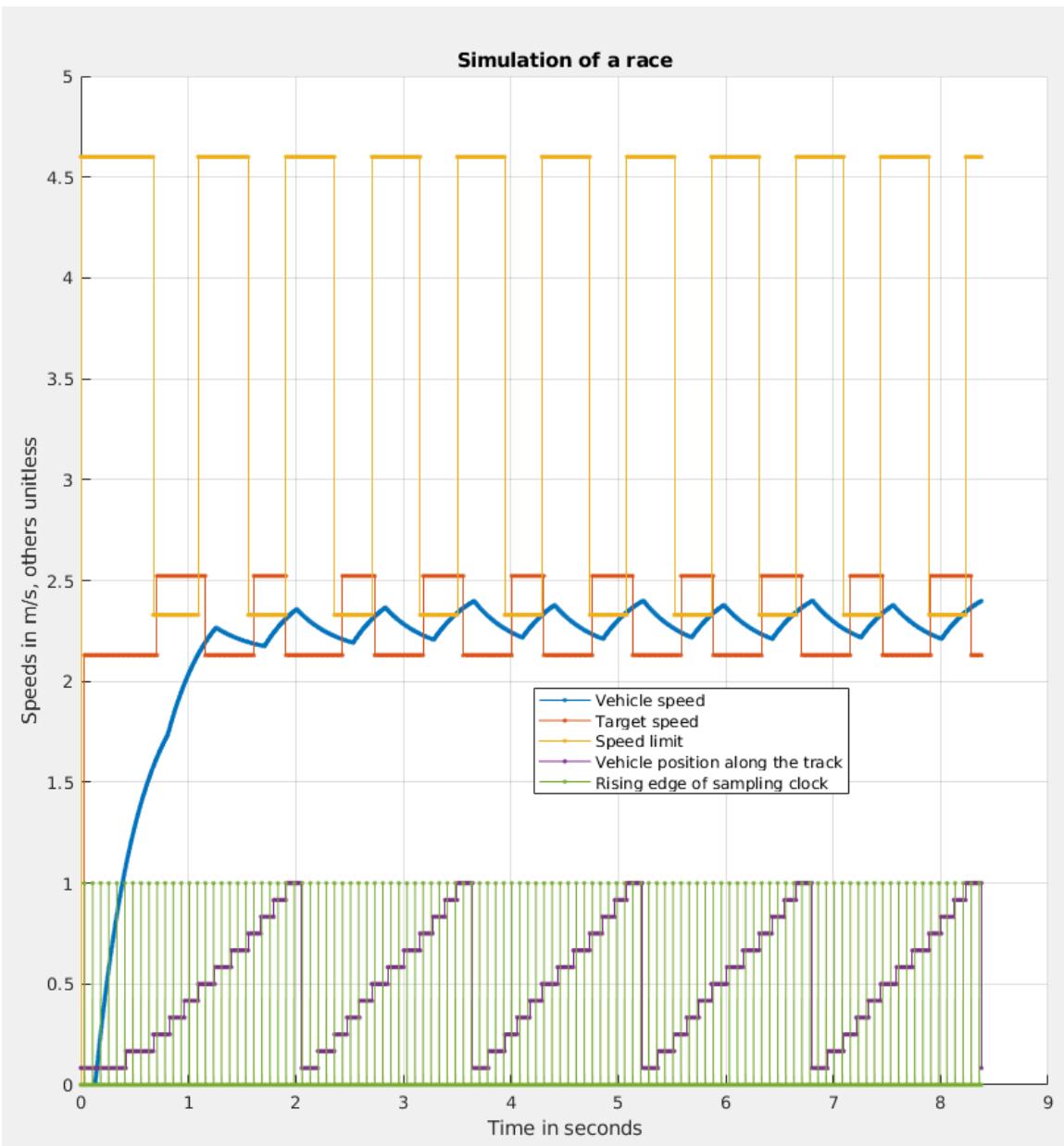


Abbildung 53: Simulation eines Rennens mit fünf Runden

```
Command Window
lap_time_vector =
    2.0560    1.5910    1.5780    1.5770    1.5780

lap_time_mean =
    1.5810

lap_time_standard =
    0.0058

fx >>
```

Abbildung 54: Rundenzeiten und Standardabweichung des Rennens

6.3.8 Ergebnisse

In Abbildung 53 sind die Ergebnisse einer Rastersuche zu sehen. Die gelbe Wellenform beschreibt die maximale Geschwindigkeit: $4.6 \frac{m}{s}$ auf Geraden und $2.3 \frac{m}{s}$ in Kurven, ohne zu entgleisen. Die orange Wellenform beschreibt die Stellgröße, also einer von 16 Spannungspegeln, die auf den Motor gegeben werden können und die damit korrespondierende Endgeschwindigkeit (auf Geraden, ohne Berücksichtigung der Motortemperatur). In blau dargestellt ist die simulierte Momentangeschwindigkeit des Autos. Lila zeigt das Passieren von Streckensegmenten auf. Jeder Stufensprung ist das Passieren eines Segments. Die grünen Peaks sind $75ms$ -Taktflanken, die vorgeben, wann eine Stellgröße an die Bahn gesendet werden kann.

Die Simulation zeigt, dass unter den Limitierungen der D132-Bahn Kurven- und Geradengeschwindigkeiten nahe der maximalen Kurvengeschwindigkeit optimal sind. Die Anzahl der nächsten Segmente, die berücksichtigt werden sollen, ist im optimalen Fall 5. Idealerweise würde das Auto immer exakt die maximale Rundengeschwindigkeit fahren. Allerdings funktioniert das nicht, da der Motor nur in seinen 16 diskreten Spannungspegeln gesteuert werden kann. Der Algorithmus moduliert also zwischen möglichen Spannungspegeln, um die maximale Kurvengeschwindigkeit im Mittel möglichst gut zu erreichen. Am realen System mussten die Zielgeschwindigkeiten höher gewählt werden, um für die erhöhte Reibung in den Kurven zu kompensieren. Da Innenkurven nicht gut von Außenkurven unterschieden werden können, ist der Algorithmus auf Innenkurven optimiert und fährt daher auf Außenkurven langsamer als nötig.

Ein weiterer limitierender Faktor ist die Erwärmung des Motors im laufenden Betrieb. Dieses Problem wird durch die erhöhte Last wegen Gewichtszunahme des Autos noch verschärft. Die Erwärmung des Motors führt dazu, dass der Motor bei gleicher Spannung weniger Drehmoment liefert. Die in Sektion 6.3.5 angesprochenen Modelle werden also noch ungenauer. Da Motorstrom und -temperatur unbekannt sind, kann nicht für sie kompensiert werden. Der Effekt ist sehr dramatisch: Die Rundenzeit verlängert sich beim Fahren von „Titan“ von $2s$ im kalten Zustand auf $2.442s$ nach 50 Runden am Stück - eine Zunahme von mehr als 20%. Menschen können nach zwei bis drei Rennen an Training eine Rundenzeit von weniger als $1.8s$ erreichen.

7 Ausblick

Die konkrete Implementierung, die gewählt wurde, weist einige Limitierungen auf, die mit anderen Ansätzen nicht auftreten.

Interface mit der Strecke

Ein großes Beispiel ist die Nutzung der bahninternen Ansteuerung des Motors. Sektion 5 weist noch auf den Direktantrieb des Motors im Auto als Alternative zur Emulation eines Handreglers hin. Dieses Verfahren umgeht die Diskretisierungsprobleme in Zeit und Wert, die die Handregleremulation mit sich bringt. Abseits davon weist der Entwurf einer Platine einige weitere Vorteile auf. Beispielsweise kann die Spannungsversorgung über die Schiene erfolgen. So kann auch auf einen Akku samt Ladeschaltkreis verzichtet werden, was wiederum zu Gewichtseinsparungen führt. Hierbei macht es durchaus Sinn, eine digitale Bahn zu verwenden, da diese die Schienen mit einer konstanten Spannung versorgt.

Wenn das Auto jedoch mit einem eigenen Motortreiber gesteuert wird, entsteht den Menschen, die gegen das autonome System fahren wollen, möglicherweise ein Nachteil. Um dieses Problem zu umgehen, sollten alle Autos, die am Rennen teilnehmen, über die gleiche Hardware (und Software) verfügen - besonders wenn eine Bremsfunktion implementiert wird, die Mensch und Algorithmus zugleich einen breiteren Fächer an möglichen Strategien zur Verfügung stellt.

Ansteuerung des Motors

Eine Vorregelung des Motors mit einem PID-Regler macht durchaus Sinn, um die Reaktionszeit des Autos zu verkürzen - auch dieser Vorteil soll dem Menschen nicht vorenthalten bleiben. Zudem ist das Gewicht der Autos auch gleich, wenn beide Teilnehmer die gleiche Hardware zur Verfügung gestellt bekommen. Der einzige Unterschied sollte die steuernde Person sein.

Verwendete Sensoren

Die verwendete Sensorik weist viele Probleme, hauptsächlich in Genauigkeit, auf. Der vielversprechendste extern betrachtende Sensor scheint DAVIS mit seiner niedrigen Datenrate aber hohen Abtastfrequenz zu sein. Externe Sensoren geben die Möglichkeit, die Rechenplattform zu ändern, da sie nicht den gleichen Gewichts-, Größen- und Stromversorgungsbegrenzungen unterliegen. So können komplexe Algorithmen implementiert werden. Da die Sensordaten in einem leistungsstarken Rechner direkt lokal zur Verfügung stehen, können diese auch lokal visualisiert und geloggt werden, sei es zu Entwicklungszwecken oder zur Informationserfas-

sung während Rennen. Zudem führt die Entfernung von Sensoren aus dem Auto zu einer Gewichtsverringerung, was sich wiederum positiv auf das Fahrverhalten auswirkt. Lohnensvolle interne Sensoren wären beispielsweise optische Flusssensoren, vielleicht auch in Kombination mit einem Sensor für die Messung der Deflektion der Führung in der Schiene. Interne Sensorik hat den Vorteil niedriger Latenz, da die Sensorwerte direkt vor Ort verarbeitet werden können.

Drahtlosinterface

Die Verwendung von ESP32 Microcontrollern ist sehr empfehlenswert: Funktionen der Plattform und verfügbare Bibliotheken machen das Programmieren unkompliziert und umfangreich. Im Projekt EVA wurde die $2.4GHz$ -Drahtlosschnittstelle allerdings bereits zur Kommunikation der zwei Teilsysteme miteinander verwendet. Die Schnittstelle kann aber vor allem für die Entwicklung viel effizienter genutzt werden: Messdaten können nahezu live übertragen werden und die Programmierung mit Over-The-Air Updates (OTA-Updates) ist erheblich schneller als über die serielle Schnittstelle mit 115200 Baud. Zudem ist das $2.4GHz$ -Band bereits gefüllt mit allerlei Störquellen: WLAN und Bluetooth-Geräte sowie Eingabegeräte mit $2.4GHz$ -Empfänger sind weit verbreitet. Für eine Übertragung mit niedriger Latenz bietet sich ein geringer besetztes Band mit niedrigeren Verlusten an die Freiraumausbreitung an, beispielsweise $900MHz$, das häufig bei Fernbedienungen von ferngesteuerten Spielzeugen zum Einsatz kommt. Hier kann dann auch auf Großteile des Protokolloverheads verzichtet werden, den gängige $2.4GHz$ -Implementierungen mit sich bringen.

Abbildungen 55 und 56 zeigen mögliche Systemarchitekturen, die für die Steuerung einer digitalen Strecke nach Erwägung des neu gewonnenen Wissens geeignet scheinen.

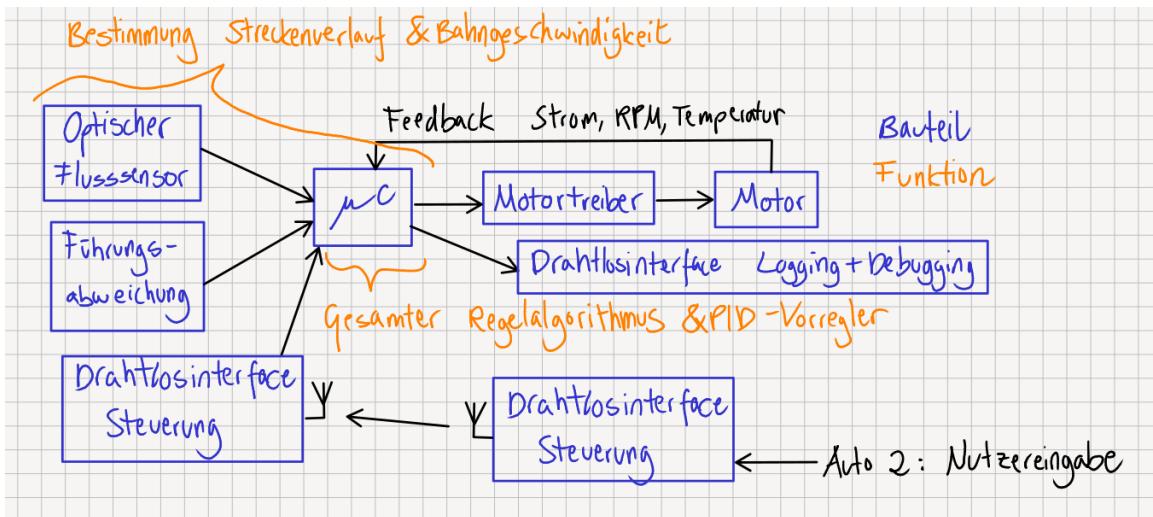


Abbildung 55: Mögliche Architektur mit gesamter Regelung innerhalb des Autos

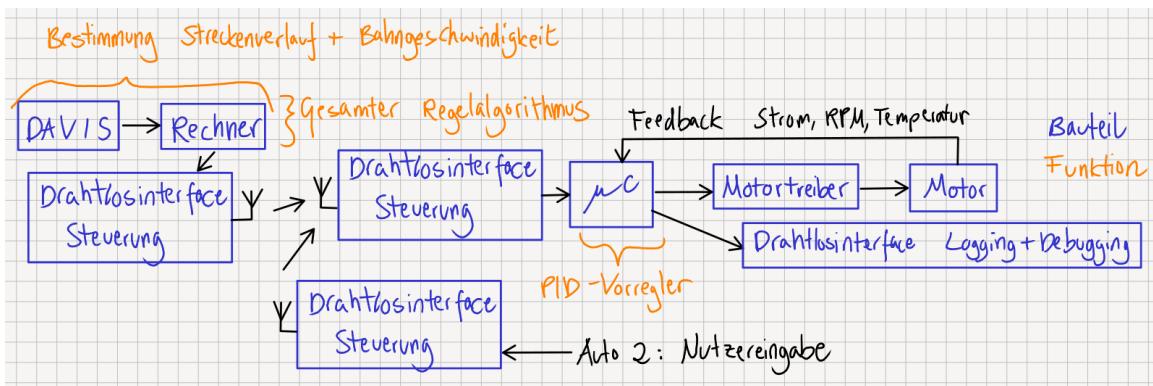


Abbildung 56: Mögliche Architektur mit zwei Teilsystemen

8 Fazit

Das Projekt EVA hat die ersten Grundsteine gelegt, um eine Automatisierung einer digitalen Rennstrecke zu ermöglichen. Es wurden viele Limitierungen und Eigenheiten der Rennstrecke und Sensoren aufgezeigt, an denen angesetzt werden kann, um die Leistung des Projekts zu steigern. Das Auto ist zwar im Moment nicht besonders schwierig zu schlagen, allerdings ist seine Konsistenz schwer zu toppen - es entgleist auf den Teststrecken nie.

9 Quellen

Literatur

- [1] *Zeitliche Meilensteine der Firma Lionel.* <http://www.lionel.com/articles/timeline/>. Besucht am 27.01.2022.
- [2] Matias Nicolás Correa Barceló. „Study of automatic control algorithms for a slot car“. B.S. thesis. Universitat Politecnica de Catalunya, 2013.
- [3] T. Delbruck u. a. „Human vs. computer slot car racing using an event and frame-based DAVIS vision sensor“. In: *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2015, S. 2409–2412.
- [4] *Video zu Ansatz mit DAVIS.* <https://www.youtube.com/watch?v=CnGPGiZuFRI>. Besucht am 07.02.2022.
- [5] Tim C Kietzmann und Martin Riedmiller. „The neuro slot car racer: Reinforcement learning in a real world setting“. In: *2009 International Conference on Machine Learning and Applications*. IEEE. 2009, S. 311–316.
- [6] Souma Mondal. „Autonomous Slot-car System“. Besucht am 03.02.2022. Diss. Citeseer, 2008.
- [7] Steven Kane und Jonathan B Scott. „The slot car stig: Performance and consistency of a slot car driven by a heuristic algorithm in an embedded microcontroller“. In: *The 16th Electronics New Zealand Conference*. Electronics Research, Department of Physics, University of Otago. 2009.
- [8] *Einleitung zum optischen Fluss.* <https://www.centeye.com/technology/optical-flow/>. Besucht am 27.01.2022.
- [9] *Anleitung zu Carrera Digital Face-Off.* <https://carrera-toys.com/api/product/getdownload?downloadid=356b156f-818f-455c-b0fc-6a68253d2967>. Besucht am 31.01.2022.
- [10] *Funktionsweise des Carrera D124/132-Systems.* <http://slotbaer.de/carrera-digital-124-132.html>. Besucht am 25.01.2022.
- [11] *Datenblatt ESP32 Wroom.* https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf. Besucht am 01.02.2022.
- [12] *Datenblatt LSM6DS3.* https://content.arduino.cc/assets/st_imu_lsm6ds3_datasheet.pdf. Besucht am 02.02.2022.

- [13] *Datenblatt ESP32*. http://espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. Besucht am 02.02.2022.
- [14] Milan Brejl und Jaroslav Necessany. „Student’s Contest: Self-Driven Slot Car Racing“. In: *2008 International Multiconference on Computer Science and Information Technology*. IEEE. 2008, S. 589–592.
- [15] *Application Note: Kalibrierung eines 3-Achsen Accelerometers*. https://www.st.com/resource/en/application_note/dm00119044-parameters-and-calibration-of-a-low-g-3-axis-accelerometer-stmicroelectronics.pdf. Besucht am 02.02.2022.
- [16] *Datenblatt von IR-Transceiverpaar ITR9909*. <https://www.digchip.com/datasheets/parts/datasheet/922/ITR9909-pdf.php>. Besucht am 03.02.2022.
- [17] Rainer Dittmar und Bernd-Markus Pfeiffer. *Modellbasierte prädiktive Regelung*. Oldenbourg, 2004.
- [18] Dan Martinec und Marek Bundzel. „Evolutionary algorithms and reinforcement learning in experiments with slot cars“. In: *2013 International Conference on Process Control (PC)*. IEEE. 2013, S. 159–162.
- [19] Johann Penner u. a. „Optimal control of a slot car racer“. In: *PAMM* 17.1 (2017), S. 537–538.