

Schlussdokumentation PREN1

Gruppe 28

Andreas Rebsamen (Elektrotechnik)

Joel Grepper (Informatik)

Manuel Omlin (Maschinentechnik)

Marco Schöni (Maschinentechnik)

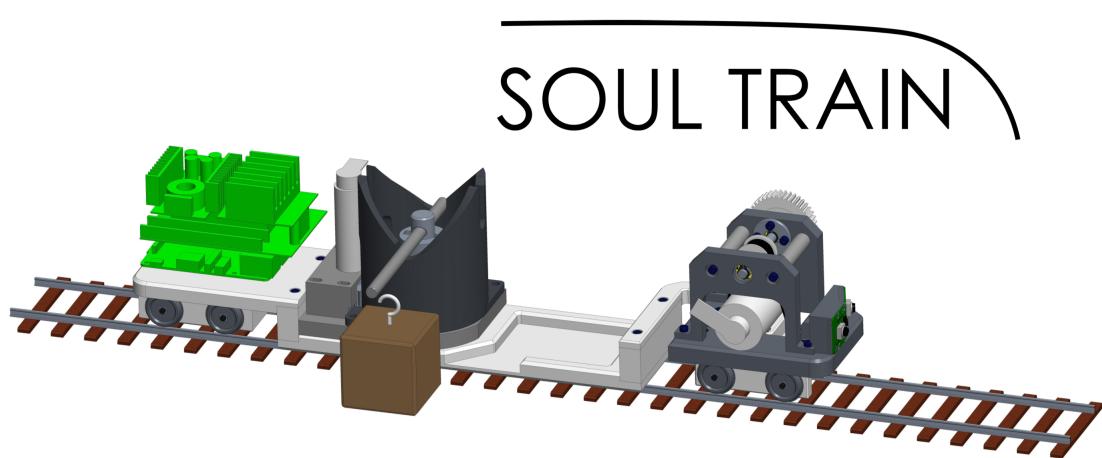
Patrick Marty (Informatik)

Steve Ineichen (Informatik)

Hochgeschwindigkeitsschienenfahrzeug "SOUL TRAIN"

Hochschule Luzern – Technik & Architektur

TA.BA_PREN1.H1801



Horw, Hochschule Luzern – Technik & Architektur, 10.01.2019

Redlichkeitserklärung

Die Verfasser bestätigen mit ihrer Unterschrift, dass die vorliegende Arbeit selbstständig, ohne fremde Hilfe und ohne Benutzung anderer als die angegebenen Hilfsmittel angefertigt worden ist.

Die aus fremden Quellen (einschliesslich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit ist in gleicher oder ähnlicher Form noch nicht vorgelegt worden.

Eingereicht in Horw, am 11. Januar 2019



Andreas Rebsamen



Joel Grepper



Manuel Omlin



Marco Schöni



Patrick Marty



Steve Ineichen

Inhaltsverzeichnis

1 Management Summary	4
2 Einleitung	5
3 Lösungskonzepte	6
3.1 Ablauf	7
3.2 Fahrwerk	9
3.3 Würfelaufnahme/Transport	18
3.4 Motorauslegung	21
3.5 Akustik	22
3.6 Beschleunigung	24
3.7 Elektronik Komponenten	26
3.8 Signalerkennung	34
3.9 Gleiserkennung	37
3.10 Aufgabentrennung zwischen Pi und Tiny	39
3.11 Interface zwischen Pi und Tiny	39
3.12 Software Lösungskonzepte	42
3.13 Grenzgeschwindigkeit	45
4 Projektmanagement	47
4.1 Organigramm	47
4.2 Zeitplanung	48
4.3 Dokumentation	49
4.4 Datenaustausch	49
5 Schlussdiskussion	50
5.1 Kosten	50
5.2 Risikomanagement	51
5.3 Nächste Schritte	52
5.4 Lessons learned	53
6 Verzeichnisse	54

1 Management Summary

Im Rahmen des Moduls PREN (Produktentwicklung) an der Hochschule Luzern wurde dieses Dokument durch das Team 28 erstellt. Das interdisziplinäre Team besteht aus Maschinenbau-, Elektrotechnik- und Informatikstudenten. In dieser Durchführung des Moduls wird in der Aufgabenstellung eine Entwicklung eines Schnellzuges gefordert. Dieser Zug muss eine definierte Strecke mit Geraden und Kurven in zwei Runden so schnell wie möglich passieren. Dabei muss der Zug einen Holzwürfel, welcher in der Startzone mittels eines Krans aufgeladen wird, während der gesamten Strecke transportieren. Während die Strecke zurückgelegt wird, muss der Zug eine Signaltafel mit Nummer erkennen und in der dritten Runde bei dieser Nummer so genau wie möglich anhalten. Die erkannte Nummer soll akustisch wiedergegeben werden. Der Zug muss autonom, also benutzerunabhängig agieren. Im Rahmen des PREN1 wurden mit Technologierecherchen mehrere Lösungsvarianten erarbeitet. Aus diesen Lösungsvarianten wurde dann die optimale Kombination von Teilkomponenten ausgewählt und zu einem Konzept als Gesamtlösung zusammengeführt.

Dieses Konzept beinhaltet eine mechanische Grundkonstruktion mit zwei Trägerwagen. Diese mechanische Grundkonstruktion wurde so entwickelt, dass der Schwerpunkt tief gehalten wird und somit hohe Kurvengeschwindigkeiten erreicht werden können. Angetrieben wird der Schnellzug mit einem DC-Motor mit eingebauter Encoderscheibe, um präzise Geschwindigkeiten zu erreichen. Der Kran, der den Würfel auf den Zug hebt, ist eine eigene Hub-Schwenk-Konstruktion mit DC-Motor. Die Signal- und Nummernerkennung wird mittels Kamera und Machine-Learning Algorithmen realisiert. Als zentrale Recheneinheit dient ein Raspberry PI 3+, welcher von einem Raspberry PI Zero und einem Mikrocontroller unterstützt wird. Im PREN2 wird das entwickelte Konzept realisiert, getestet und optimiert.

2 Einleitung

In dieser Arbeit liegt das Gesamtkonzept für einen autonomen Schnellzug vor. Dieses Konzept wurde im HS18 im Rahmen des PREN1 entwickelt und wird im FS19 im Modul PREN2 realisiert. Die Aufgabe besteht darin, einen Schnellzug zu entwickeln, welcher eine definierte Strecke mit Geraden und Kurven so schnell wie möglich zurücklegt. Im Startbereich muss mittels einer am Zug befestigen Konstruktion ein Holzwürfel auf den Zug gehoben und transportiert werden. Danach muss der Zug eine Lichtschranke passieren und zwei Runden auf der Strecke so schnell wie möglich absolvieren. Dabei muss er ein seitlich befindendes Signal mit Nummer erkennen, welches die Halteposition signalisiert. Diese Halteposition muss in der dritten Runde angefahren werden, und der Zug soll dort so nahe wie möglich anhalten. Die erkannte Nummer soll auch mittels akustischem Signal ausgegeben werden.

Der Zug wurde mit folgenden Schwerpunkten entwickelt:

- Kompaktheit
- Einfachheit
- niedriges Gewicht
- Robustheit
- niedrige Kosten

Im Hauptteil dieser Arbeit wird das Konzept des Zuges beschrieben. Die Beschreibung ist aufgeteilt in die drei Fachgebiete Maschinentechnik, Elektrotechnik und Informatik. Im Maschinentechnik-Bereich wird die mechanische Grundkonstruktion des Zuges und des Krans für den Holzwürfel erläutert. Der Antrieb, die Sensorik und die Stromversorgung des Zuges wird im Elektrotechnikteil beschrieben. Im Abschnitt Informatik wird die Signalerkennung, die akustische Ausgabe und der Softwareaufbau beschrieben. Projektmanagement, Kostenübersicht und Schlussdiskussion sind im hinteren Teil der Arbeit zu finden.

In diesem Konzept wurden bei der Entwicklung die Schwerpunkte Kompaktheit, Einfachheit und niedriges Gewicht berücksichtigt, um eine optimale maximale Geschwindigkeit mit dem Zug zu erreichen. Dabei soll aber auch ein Schwergewicht auf Robustheit und Prozesssicherheit gelegt werden. Zusätzliche Optimierungen, besonders in der maximalen Geschwindigkeit, können während dem PREN2, der Realisierungsphase, erzielt werden.

3 Lösungskonzepte

Damit das Schienenfahrzeug alle Teilaufgaben optimal erfüllen kann, soll im Verlauf von PREN1 ein Lösungskonzept entwickelt und gewisse Telfunktionen getestet werden. Das Gesamtkonzept besteht aus verschiedenen Teilkonzepten, für die jeweils mehrere Lösungsvorschläge aufbereitet und anschliessend bewertet wurden. Der Entscheidungsprozess wurde im Rahmen eines Testates dokumentiert und wird in diesem Dokument nicht thematisiert (befindet sich im Anhang). In den folgenden Unterkapiteln wird jede definierte Telfunktion des Schienenfahrzeugs beschrieben und die jeweilige Lösung dazu präsentiert. Liegen bereits praktische Tests vor, werden diese ebenfalls dargestellt. Für alle Probleme soll eine möglichst einfache und doch effektive Lösungsvariante präsentiert werden. Im unmittelbar nächsten Kapitel wird der Ablauf mit einem Zustandsdiagramm dargestellt.

Übersicht Lösungskonzepte

- Ablauf
- Fahrwerk
- Würfelaufnahme / Transport
- Motorauslegung
- Akustik
- Beschleunigung / Geschwindigkeit
- Elektronik & Komponenten
- Signalerkennung & Gleiserkennung
- Zusammenspiel RPI und Tiny
- Software

3.1 Ablauf

Dieses Kapitel gibt eine Übersicht über den gesamten Ablauf, der nötig ist, um die Aufgabenstellung zu erfüllen. Der Ablauf ist in Abbildung 1 grafisch dargestellt. Dieser Ablauf kann in fünf Zustände unterteilt werden.

- Initialisierung
- Würfelaufnahme
- Hochgeschwindigkeits-Fahrt
- Parkplatzsuche
- Parkieren

Im folgenden werden die Zustände kurz beschrieben.

Initialisierung

Alle Systeme werden gestartet, sobald sie mit Strom versorgt werden. Alle Schnittstellen werden initialisiert und die Kommunikation zwischen den Komponenten beginnt. Beim Ende der Initialisierung soll über das Web Interface angezeigt werden, dass das System bereit ist und die Fahrt gestartet werden kann.

Würfelaufnahme

Nach dem Startsignal über das Web Interface fährt der Zug in langsamer Fahrt vorwärts, bis der Würfle von den Sensoren erfasst wird. Sobald der Würfel erkannt wurde, hält der Zug an und der Schwenker mit dem Würfel wird eingefahren.

Hochgeschwindigkeits-Fahrt

Nachdem der Würfel Aufgeladen ist, kann der Zug beschleunigen. Der Zug soll immer die höchstmögliche Geschwindigkeit fahren. Die Geschwindigkeit wird für die Kurven gemäss der Spurerkennung angepasst. Auch sucht das System dauernd das Info Signal und erkennt die Nummer darauf. Sobald die Nummer erkannt wurde, gibt das System diese Information akustisch aus. Sobald das Signal für das Ende der Zeitmessung erkannt wird, verlangsamt der Zug und beginnt die Parkplatzsuche.

Parkplatzsuche

Um das korrekte Haltesignal zu finden, kann der Zug eine so tiefe Geschwindigkeit fahren wie nötig. Die Bilderkennung sucht das korrekte Signal und entscheidet, wo der Zug parkiert werden soll.

Parkieren

Nachdem das korrekte Signal erkannt ist, wird der Parksensor ausgeklappt. Dieser misst die Distanz zum Haltesignal bis ein bestimmter Schwellwert für die Haltedistanz erreicht ist. Beim Erreichen der Haltedistanz stoppt der Zug und beendet damit die Fahrt.

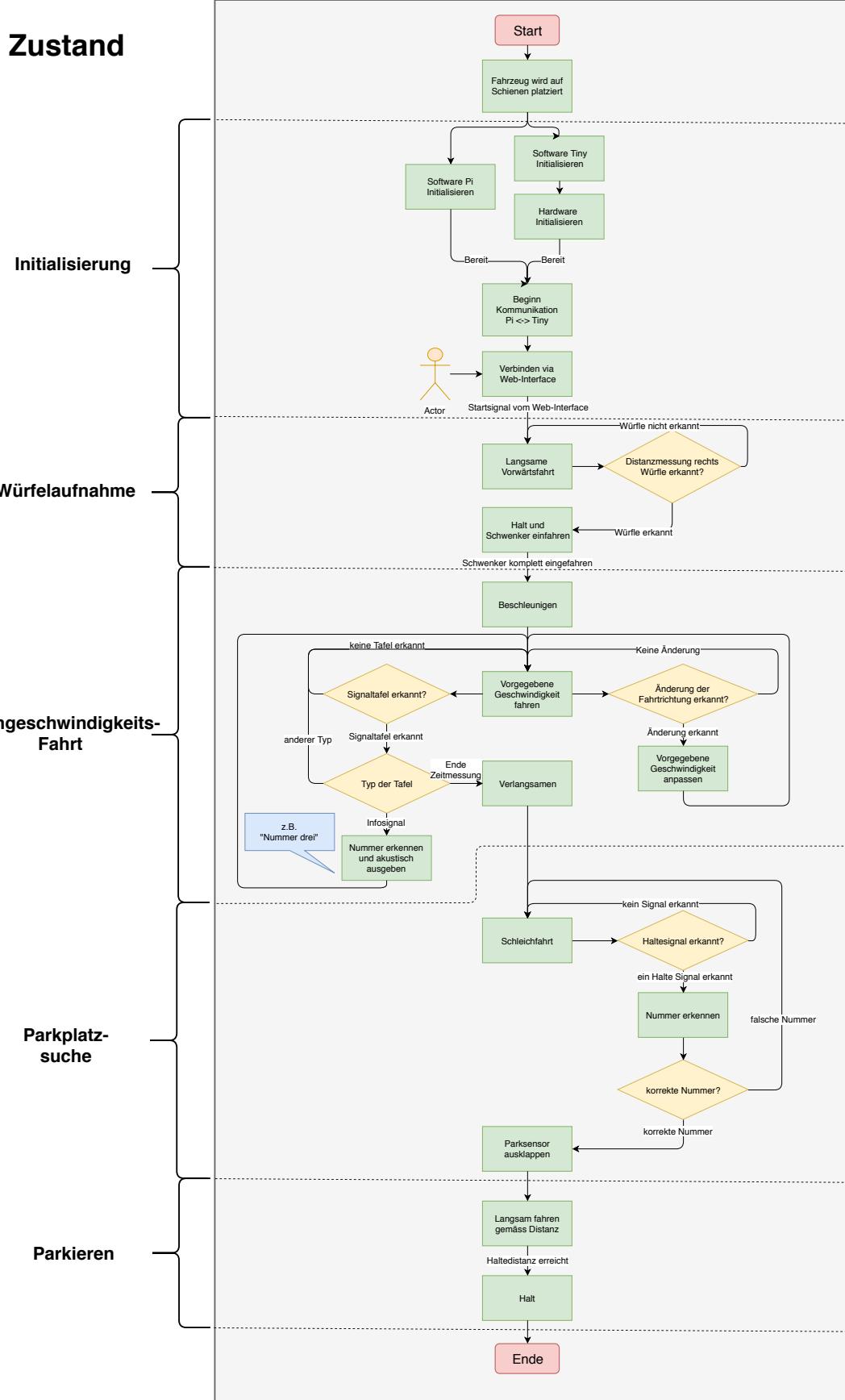


Abbildung 1: Ablaufdiagramm

3.2 Fahrwerk

Die Lokomotive in Abbildung 2 (siehe Tabelle 1) ist in die drei Unterbaugruppen Antriebswagen (Position 1), Führungswagen (Position 2) und Ladungsträger (Position 3) unterteilt. Der Antriebswagen enthält alle notwendigen Komponenten, um die Lokomotive zu beschleunigen und wieder abzubremsen. Zusätzlich sind die Kameras für die Spur- und Signalerkennung an ihm angebracht. Der Führungswagen hingegen dient lediglich als Abstützung für den Ladungsträger. Er bietet aber zusätzlichen Bauraum für elektronische Komponenten.

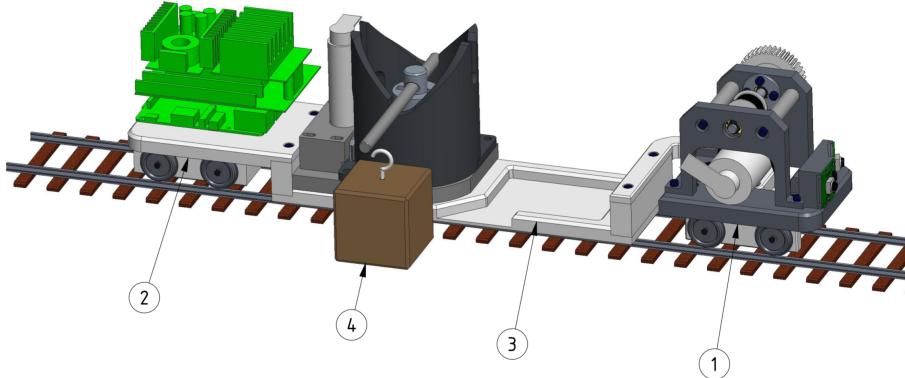


Abbildung 2: Baugruppe Lokomotive

Position	Bezeichnung
Position 1	Antriebswagen
Position 2	Führungswagen
Position 3	Ladungsträger
Position 4	Transportgut (Würfel)

Tabelle 1: Positionsnummern der Lokomotive

In der Querschnittsdarstellung in Abbildung 3 sind die Lagerungen zwischen dem Ladungsträger und dem Antriebs- beziehungsweise dem Führungswagen besser sichtbar. Ebenfalls ist ersichtlich, dass der Antriebswagen durch einen Zahnriemen angetrieben wird. Der Aufbau der einzelnen Unterbaugruppen wird in den nächsten Abschnitten genauer vorgestellt.

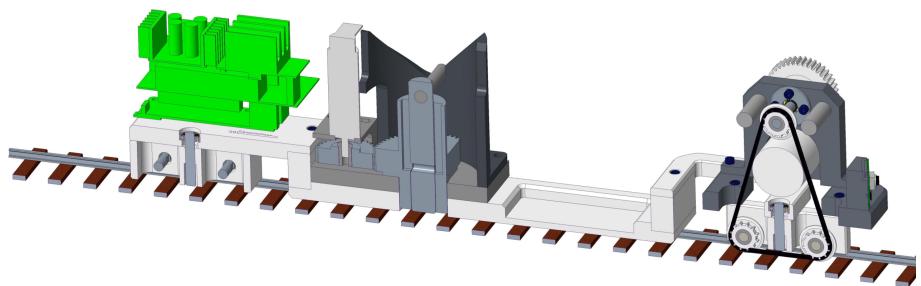


Abbildung 3: Schnittansicht der Lokomotive

3.2.1 Antriebs- und Führungswagen

Der Grundaufbau der beiden Wagen ist derselbe, mit dem Unterschied, dass der Antriebswagen durch einen Motor angetrieben wird. Der Grundwagen, beziehungsweise der Führungswagen in Abbildung 4 (siehe Tabelle 2) besteht aus einem Rahmen und einer Platte, welche miteinander verstiftet (Position 2) und verschraubt (Position 3) sind. Im Rahmen werden die beiden Achsen jeweils mit einem Los- (Position 7) und einem Festlager (Position 4) gelagert und mit Sicherungsringen (Position 6) gesichert. Die Achsen sind an beiden Enden mit einem Gewinde versehen, damit die Räder bei Bedarf schnell und einfach gewechselt werden können, ohne dass der ganze Wagen auseinander genommen werden muss. Die Anfräsfäche auf der Welle (Position 5) dient für das bessere Befestigen der Räder mit einem Gabelschlüssel.

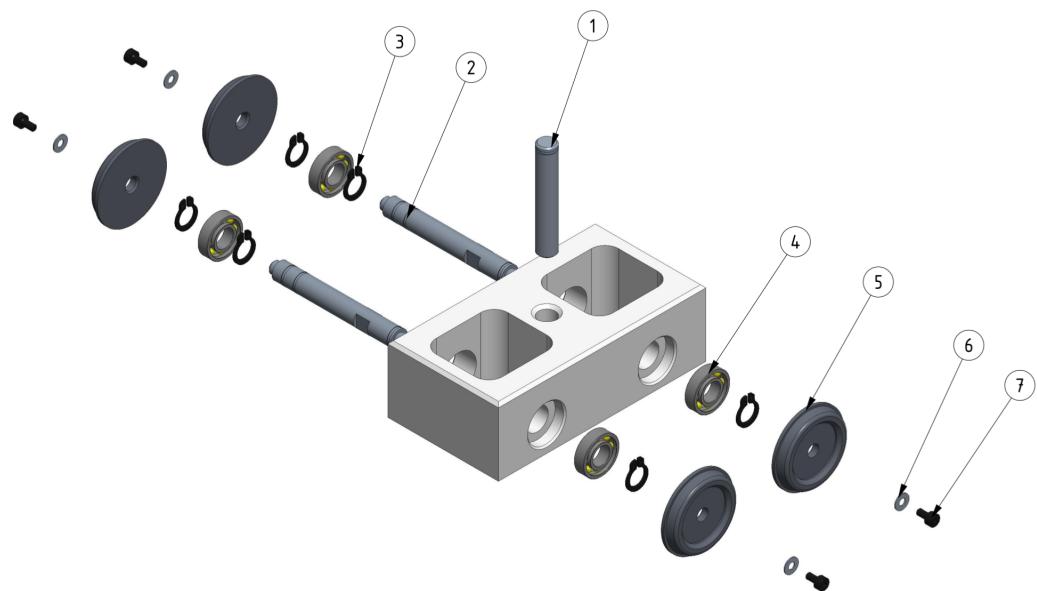


Abbildung 4: Explosionsdarstellung Grundwagens

Position	Bezeichnung
Position 1	Drehachse Wagen-Ladungsträger (eingepresst)
Position 2	Achsen (Gewinde an beiden Seiten, Anfräsfäche für Gabelschlüssel)
Position 3	Sicherungsring für Rillenkugellager
Position 4	Eingepresstes Rillenkugellager (Festlager) bzw. Loslager
Position 5	Rad
Position 6	Unterlagscheibe
Position 7	Zylinderschraube

Tabelle 2: Positionsnummern des Grundwagens

Der Antriebswagen in Abbildung 5 (siehe Tabelle 3) ist, wie bereits erwähnt, grundsätzlich gleich aufgebaut wie der Führungswagen. Jedoch ist der Rahmen aufgrund des Riemenantriebs H-Förmig aufgebaut. Das heisst, die beiden Nuten des Wagenrahmens sind nach Aussen hin offen. Das Ziel dieser Konstruktion ist es den Riemen, falls nötig, schnell wechselbar zu montieren. Durch die H-Form können die Achsen und die Räder für die Riemenmontage am Rahmen montiert bleiben. Die Platte, welche am Rahmen angebracht ist, ist für die Kameras grösser dimensioniert. Die Kamera für die Gleiserkennung wird einstellbar befestigt, damit bei der Testphase des Prototyps Optimierungen des Winkels vorgenommen werden können. Die Kamera für die Signalerkennung wird fix montiert. Der Stromfluss von der Schiene auf die Lokomotive erfolgt über vier Schleifkontakte, welche als Einkaufsteile von Lieferanten bezogen werden. Davon sind pro Wagen zwei jeweils zwischen den Rädern angebracht.

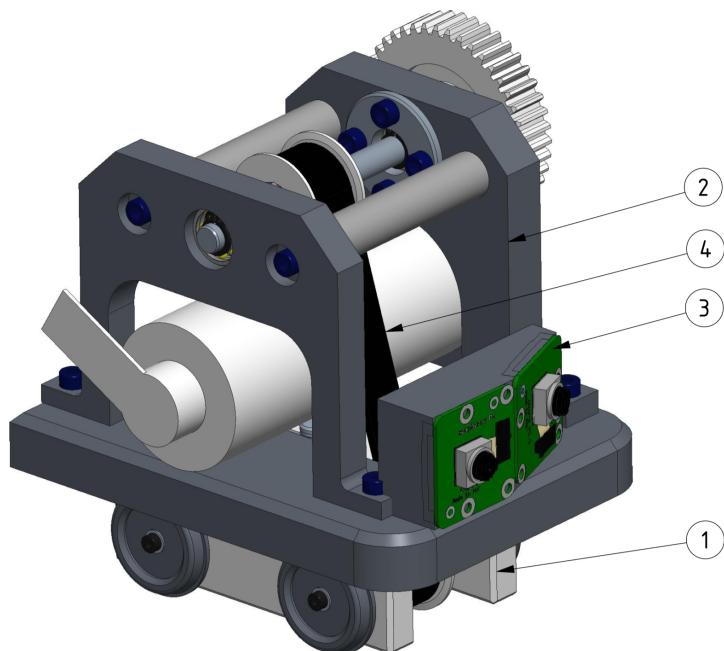


Abbildung 5: Baugruppe Antriebswagen

Position	Bezeichnung
Position 1	Wagen
Position 2	Antriebseinheit
Position 3	Kameras
Position 4	Zahnriemen

Tabelle 3: Positionsnummern des Antriebswagens

Die Antriebseinheit in Abbildung 6 (siehe Tabelle 4) besteht aus einem Grundgestell, an welchem die Lagerung der Antriebswelle und der Motor angebracht sind. Das Drehmoment vom Motor wird über ein geradverzahntes Zahnrad mit einem Übersetzungsverhältnis von 1:2 auf eine Achse übertragen. Von dieser wird es über einen Riemen auf die beiden Radachsen und somit auf die Räder weitergeleitet. Durch die Berechnung der maximalen Geschwindigkeit wird entsprechend der Motor ausgelegt. Dies wird im nächsten Abschnitt genauer beschrieben.

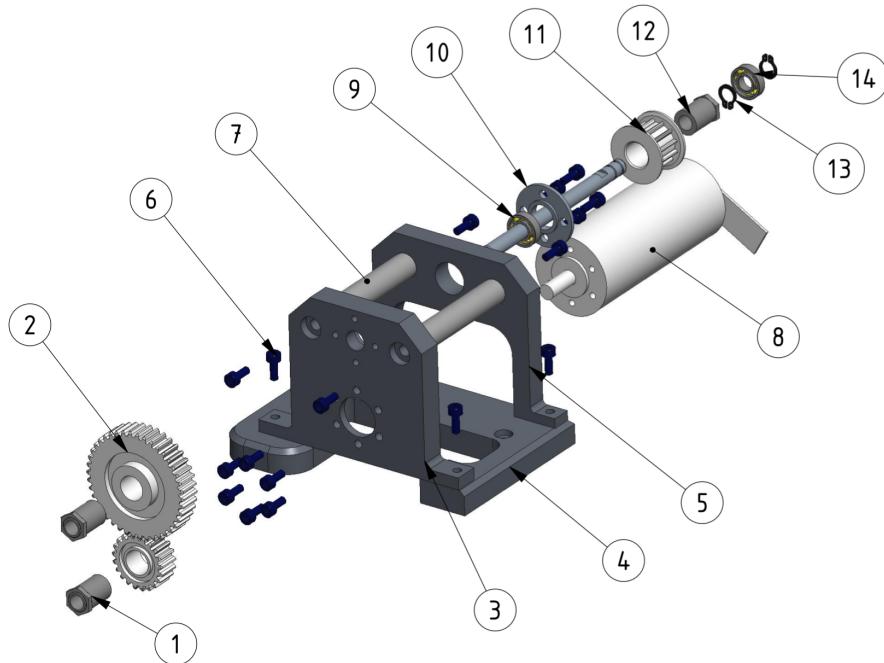


Abbildung 6: Explosionsdarstellung Antriebseinheit

Position	Bezeichnung
Position 1	Spannsatz
Position 2	Zahnräder (Übersetzung 1:2)
Position 3	Motorhalterung
Position 4	Grundplatte
Position 5	Achsenlagerplatte
Position 6	Zylinderschraube
Position 7	Distanzhülsen
Position 8	Motor
Position 9	Rillenkugellager (Loslager)
Position 10	Lagersicherungsplatte
Position 11	Zahnriemenenrad
Position 12	Spannsatz
Position 13	Sicherungsring
Position 14	Rillenkugellager (Festlager)

Tabelle 4: Positionen Antriebseinheit

Beschleunigungsberechnung

Wie schnell die Lokomotive beschleunigen kann, hängt von der Reibung zwischen Rad und Schiene und der Masse des Zuges ab. Die Grunddefinition der Beschleunigung ist der Quotient von Kraft und Masse. Die Reibung ist durch den Reibkoeffizienten bestimmt, welcher sich von Materialpaarung zu Materialpaarung unterscheidet (siehe Tabelle 5). Zusätzlich ist der Reibkoeffizient von der Oberflächenbeschaffenheit, der Rauigkeit, abhängig. Je grösser die Rauigkeit, desto mehr Reibung entsteht und desto schneller kann beschleunigt werden.

Material Schiene	Material Rad	Reibungskoeffizient
Stahl	Stahl	0.12
Stahl	Holz	0.3
Stahl	Kunststoff	0.08
Stahl	Gummi	0.3

Tabelle 5: Reibungskoeffizienten von Materialpaarungen (durch Versuche ermittelt)

Um die maximale Beschleunigung zu berechnen, wird das Gesamtgewicht der Lokomotive auf die vier Räder aufgeteilt. In der Tabelle 6 sind die gegebenen Grössen für die nachfolgenden Berechnungen aufgelistet.

Grösse	Wert
Durchmesser Rad [D]	22 Millimeter
Reibungskoeffizient [k]	0.3

Tabelle 6: Grössen für die Beschleunigungsberechnung

$$F_{Rad} = \frac{F_G}{8} = \frac{m \cdot g = 3kg \cdot 9.81m/s^2}{8} = 0.375N$$

$$F_{Reibung} = F_{Rad} \cdot k = 0.375N \cdot 0.3 = 0.1125N$$

$$M_{Rad} = F_{Rad} \cdot 0.5 \cdot D_{Rad} = 0.1125N \cdot 0.5 \cdot 22mm = 1.24mNm$$

$$a_{max} = \frac{F_{Reibung}}{\frac{F_{Rad}}{g}} = \frac{0.1125N}{\frac{0.375N}{9.81m/s^2}} = 2.94m/s^2$$

Geschwindigkeitsberechnung

Die Fahrgeschwindigkeit der Lokomotive wird durch zwei Faktoren bestimmt. Einerseits muss die maximale Geschwindigkeit der Anforderungsliste eingehalten werden, und andererseits wird die Geschwindigkeit in der Kurvenfahrt durch den Schwerpunkt des Fahrzeuges eingeschränkt. Nachdem die Lokomotive auf die Fahrtgeschwindigkeit beschleunigt wurde, ist die Rollreibung das einzige, was der Motor mit seiner Leistung

kompensieren muss. In der Anforderungsliste wurde eine minimale Geschwindigkeit von 0.5 Meter pro Sekunde festgelegt. Der begrenzende Faktor der Geschwindigkeit in der Kurve ist der Schwerpunkt der Lokomotive. Je tiefer dieser ist, umso schneller kann die Kurve abgefahren werden. Über die Zentripedalkraft und die Gewichtskraft der Lokomotive wird die Momentengleichung aufgestellt und anhand der gegebenen Werte in Tabelle 7 wird die maximal erreichbare Geschwindigkeit in der Kurve, ohne aus den Gleisen zu kippen, berechnet. Sie sind durch getroffene Annahmen entstanden, da noch nicht alle Komponenten und die dazugehörigen Massen festgelegt sind.

Das Kippmoment wird durch den Aufbau der Lokomotive minimiert, da der Schwerpunkt durch den Ladungsträger mehr in das Zentrum des Kreismittelpunktes rückt. Dies wird in den nachfolgenden Berechnungen nicht berücksichtigt.

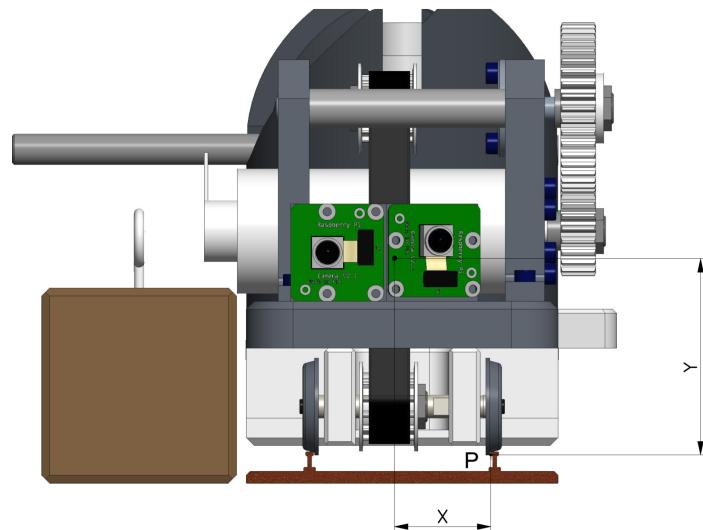


Abbildung 7: Schwerpunkt der Lokomotive (Symbolisch)

Grösse	Wert
Minimaler Radius [r]	0.8 Meter
Masse [m]	3 Kilogramm
Schwerpunkt in x-Achse (maximaler Wert) [x]	0.0225 Meter
Schwerpunkt in y-Achse (maximaler Wert) [y]	0.05 Meter

Tabelle 7: Größen für die Geschwindigkeitsberechnung

Die Gewichts- und Zentripedalkraft, welche das Gleichungssystem für die Geschwindigkeitsberechnung bilden, sind wie folgt definiert:

$$F_G = m \cdot g = 3kg \cdot 9.81m/s^2 = 29.4N$$

$$F_{max,z} = \frac{F_G \cdot x}{y} = \frac{29.4N \cdot 0.0225m}{0.05m} = 13.24N$$

Da das Drehmoment eine vektorielle Grösse ist, müssen die beiden entstehenden Momente am Drehpunkt "P" am Gleis zusammen Null ergeben. Oder anders gesagt, müssen die beiden Momente gleich gross sein, damit das System "statisch" bestimmt ist. Die Berechnungen sind auf den kleinsten Kurvenradius ausgelegt, da dort die grösste Zentripedalkraft entsteht. Somit ergibt sich eine maximale Geschwindigkeit von 1.53 Meter pro Sekunde.

$$F_{max,z} = \frac{m \cdot v_{max}^2}{r}$$

$$v_{max} = \sqrt{\frac{F_{max,z} \cdot r}{m}} = \sqrt{\frac{13.24N \cdot 0.8m}{3kg}} = 1.53m/s$$

3.2.2 Ladungsträger

Der Ladungsträger ist das Verbindungselement von Antriebswagen und Führungswagen. Er wird an beiden Enden drehbar mit je einem Radiallager gelagert. Die Flächen der Platten reiben auf den beiden Wagen. Durch eine optimale Materialpaarung wird diese Reibkraft jedoch vernachlässigbar klein. Der Träger besteht aus den drei Teilen mit Position 1,3 und 7 (siehe Abbildung 8), welche durch Zylinderschrauben (Position 5) und einer Zwischenplatte (Position 2) zusammengebaut wird. Der Hauptgrund ist die einfacheren, materialsparenden, sowie kostengünstigere Herstellung.

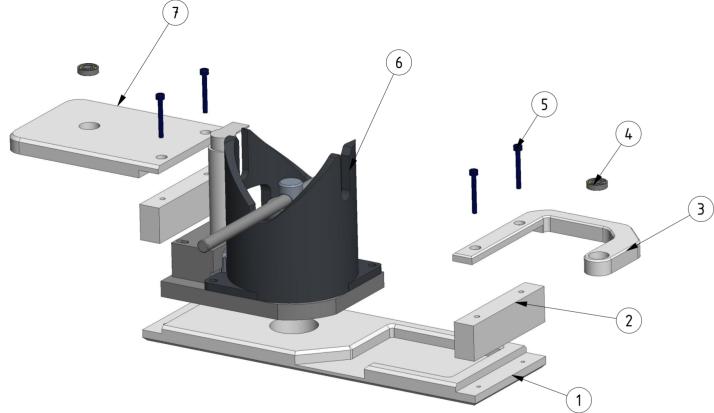


Abbildung 8: Explosionsdarstellung Ladungsträger

Position	Bezeichnung
Position 1	Grundplatte
Position 2	Zwischenplatte
Position 3	Bügelgelenk
Position 4	Rillenkugellager
Position 5	Zylinderschrauben
Position 6	Würfelkran
Position 7	Plattengelenk

Tabelle 8: Positionen des Ladungsträgers

3.3 Würfelaufnahme/Transport

Um den Würfel rechts neben der Gleisstrecke aufzunehmen, wird eine einfache Lösung angestrebt, steuerungstechnisch sowie mechanisch. Aus dem morphologischen Kasten (Anhang) und der Nutzwertanalyse geht hervor, dass die Würfelaufnahme mit einem Draht und einem Stab durchgeführt wird. Damit nur ein Aktor angesteuert werden muss, wird von dem Prinzip einer Kurvenscheibe Gebrauch gemacht. Die gesamte Vorrichtung besteht grundsätzlich aus drei Elementen: Einem Kran zur Lastaufnahme, einem Antriebsstrang und der Kurvenscheibe.

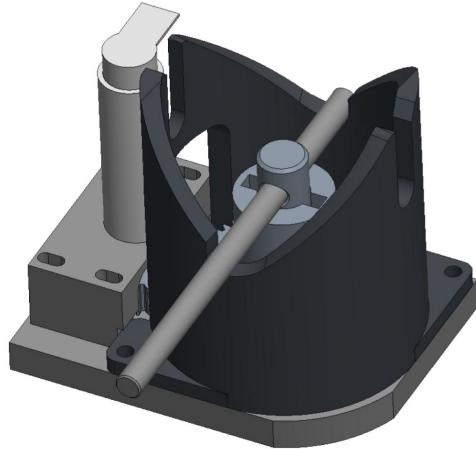


Abbildung 9: Baugruppe

3.3.1 Kran

Der Kran besteht aus drei Drehteilen, welche mit einer Pressverbindung zusammengefügt wurden. Das zentrale Element des Krans wird auf Grund seiner optimalen Gleiteigenschaften und der geringen Dichte aus Teflon gefertigt. Der kleinere Stahlstift ist für die Drehmomentübertragung zuständig. Der grössere der beiden Stahlstifte ist der eigentliche Ausleger. An dessen Ende wird ein Draht aus Federstahl geformt und angehängt. Dieser Draht soll als Haken zur Lastaufnahme dienen. Weiter ist der Ausleger in beide Richtungen von der Drehachse ausgedehnt, da die Kurvenscheibe zwei Laufflächen hat, um für einen stabilen Hub zu sorgen. Der ganze Aufbau wird mittels einer Spielpassung in einer Bohrung mit zwei Längsnuten in einem 3D gedruckten, modifizierten Zahnrad gelagert.



Abbildung 10: Draufsicht

3.3.2 Antriebsstrang

Der Antrieb besteht aus einem Motor, dessen Aufhängung und zwei Zahnrädern. Der Motor ist ein bürstenbehafteter Motor mit Encoder und Getriebe vorne drauf. Mit dieser Variante und der Übersetzung, zusammengesetzt aus Getriebe und Zahnrädern, kann von der Steuerung aus genau definiert werden, wie viele Umdrehungen der Motor benötigt, um mit dem Kranausleger eine Viertelumdrehung zu fahren. Das eine Zahnrad ist Standard und von Mädler eingekauft. Das zweite Zahnrad jedoch wurde nur als STEP von Mädler heruntergeladen und anschliessend im CAD bearbeitet. Die Bohrung und der Flansch in der Mitte wurden verlängert und mit zwei Längsnuten versehen. Die Bohrung gilt als Axialführung und die Nuten dienen als Drehmomentübertragung.

3.3.3 Kurvenscheibe

Die Kurvenscheibe ist ebenfalls ein 3D-Druckteil. Der Grundkörper ist ein Rohr mit dem Aussendurchmesser 80 mm. An diesem wurden zwei Bahnführungen mittels Freiformflächen für den Kranausleger erzeugt. Die Steigung dieser Flächen ist variabel. Zu Beginn ist die Steigung gering und wird dann exponentiell grösser. Dies wurde aus dem einen Grund gewählt, damit das Anfahren für den Motor nicht zu streng ist. Nachdem die Drehbewegung und der vertikale Hub gemacht wurden, stoppt der Motor und der Kranausleger sollte durch die Schwerkraft heruntergezogen werden. Der Würfel wird nun in der für ihn vorgesehenen Aufnahme auf dem Zug platziert.

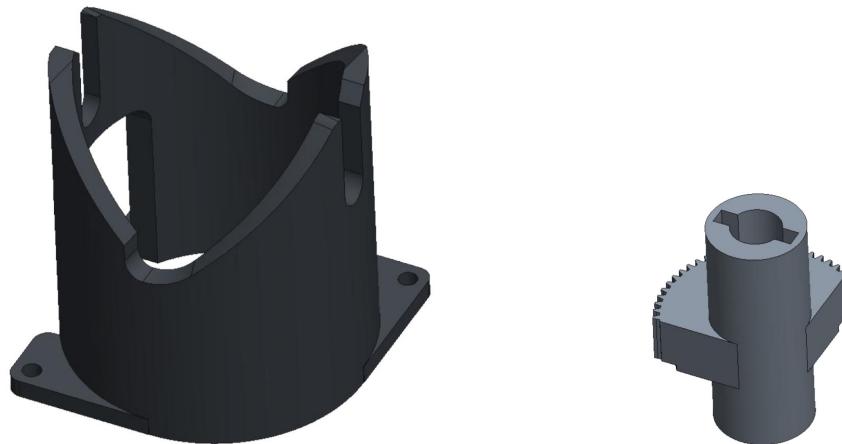


Abbildung 11: Links: Kurvenscheibe, Rechts: Zahnrad

3.3.4 Testaufbau

Der Testaufbau besteht hauptsächlich aus 3D-Druckteilen und weichen Kunststoffen. Er dient momentan als Funktionsmuster. Wenn sich dieser weiter bewährt, möchte man mit den gefertigten Teilen weiterarbeiten. Der Test dient zur Probe des ausgewählten Lösungskonzepts der Würfelaufnahme. Erste Tests zeigen, dass die Funktion mit kleinen Anpassungen direkt umgesetzt werden kann.

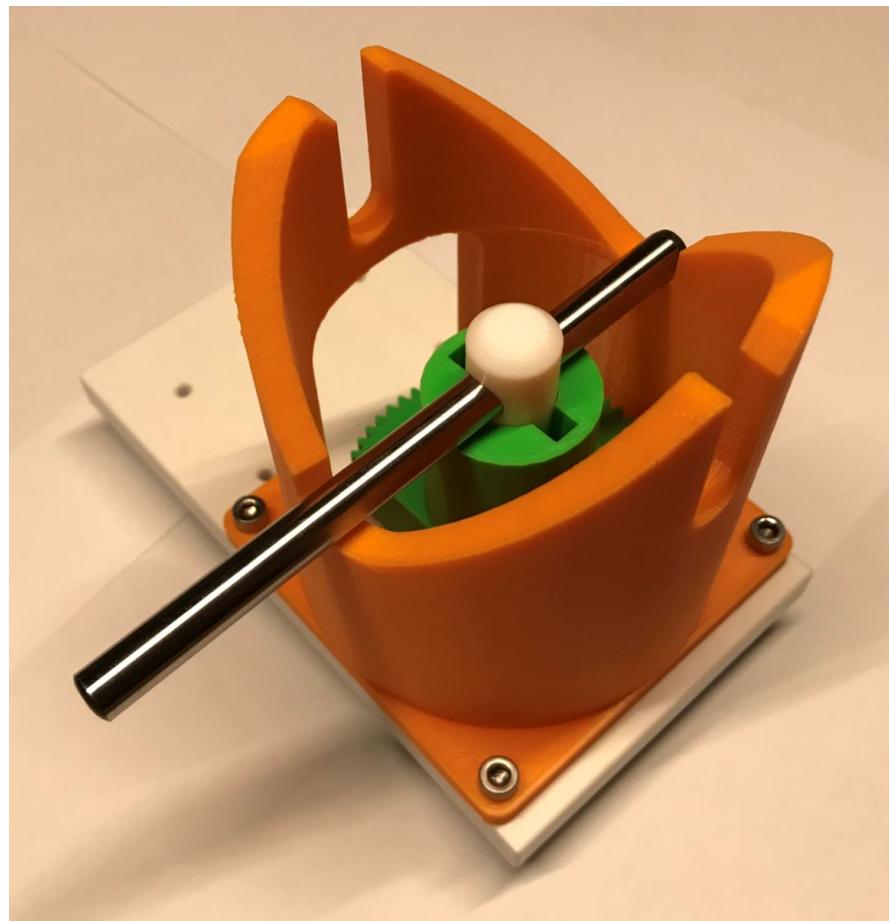


Abbildung 12: Testaufbau

3.4 Motorauslegung

Um die Aufgabenstellung der schnellen Fahrt auf Schienen bestmöglichst zu erfüllen, wird ein zuverlässiger starker Motor benötigt. Um einen solchen aus einer Vielzahl von Auswahlmöglichkeiten zu definieren, hat man sich auf den Katalog definiert maxon motor ag beschränkt. Im Anhang findet man ein Dokument mit den Berechnungen zur Motorauswahl. In Absprache der Disziplinen Elektrotechnik und Maschinentechnik wurde ein bürstenbehafteter Gleichstrommotor als geeignetes Modell definiert. Im vorher schon erwähnten Dokument wird für den gesamten Zug eine Masse von 3 Kilogramm gerechnet und einen Raddurchmesser von 22 mm. Für eine Endgeschwindigkeit von 3 m/s mit den definierten Raddurchmessern ergibt sich eine Drehzahl von 2600 1/min. Mit einer Übersetzung von 2, was mit Zahnrädern und den Platzverhältnissen gut realisierbar ist, ergibt sich eine Abgangsdrehzahl für den Motor von 5200 1/min. Das ist im Rahmen der Motoren von maxon motor ag. Was jedoch den Motor an seine Grenzen führt, wird die Grenzbeschleunigung sein. Durch die Beschleunigung bedingte Trägheitskraft, welche durch ein Moment vom Motor überwunden werden muss. Das Moment rechnet sich aus der gewollten Beschleunigung, der Masse und dem Hebel auf den Rädern. Wird nun die Übersetzung von 2 noch eingerechnet, ergibt sich ein Moment von 110 mNm. Durch die Schienen haben wir eine gewisse elektrische Leistung zur Verfügung. Diese ergibt sich aus dem Produkt der Spannung 20 Volt und dem Strom von 3 Ampere. Theoretisch stehen also 60 Watt zur Verfügung. Als eine preiswerte Lösung in Form eines DC Motors kommen bei maxon motor ag nur die zwei Produktreihen DCX und RE in Frage. Maxon motor ag bietet ein Sponsoring in Form von Motoren mit kleinen Makeln an. Diese dürfen nicht mehr ausgeliefert werden. Die Wunschmotoren der Gruppe 28 sind auf Grund der Leistung der DCX 32 oder der RE 30.



Abbildung 13: DC Motoren

3.5 Akustik

Während der Fahrt wird ein Signal mit einer Nummer gelesen, diese Nummer soll am Ende der Fahrt akustisch ausgegeben werden. In diesem Kapitel wird das Lösungskonzept für die akustische Komponente aufgezeigt.

Anforderungen

- Zahl akustisch wiedergeben (Speaker oder Buzzer)
- Korrekte Zahl wird wiedergegeben
- Kompakt
- Günstig
- Keine eigene Stromquelle
- Verständliche Ausgabe

Konzept Das Audiosignal wird über einen Buzzer wiedergegeben. Der Buzzer kann über GPIO (General Purpose Input Output) angesprochen werden. Für unseren Anwendungsfall wird eine Frequenz an einem Ausgang ausgegeben. Die 3.3V des Raspberry Pi reichen aus, um den Buzzer zu versorgen und auch der Signalpegel des GPIO ist genügend hoch.

Komponente Als Komponenten verwenden wir einen 3.3V Passiv Buzzer. Der Buzzer ist mit seinen 25mm x 25mm sehr kompakt und sollte ohne Probleme auf dem Zug Platz finden. Auch ist er für unter 5 Fr. zu erwerben und schont somit das Budget. Die Verbindung von Buzzer zu Raspberry Pi ist online gut dokumentiert und sollte keine unerwarteten Probleme mit sich bringen.

Name	Passiver Buzzer / Speaker, 3.3V
Preis	5 Fr.
Länge	25mm
Breite	25mm
Höhe	7mm
Gewicht	10g
Versorgungsspannung	3.3V

Tabelle 9: technische Daten (www.playzone.ch)

Bauplan / Interface Über die GPIO Header des Raspberry Pi kann der Buzzer direkt angesprochen und versorgt werden. Ein einzelnes Signalkabel reicht für die Kommunikation aus und hält den Aufbau einfach.

Bezeichnung	GPIO Header	Buzzer
Stromversorgung	3V3	VCC
Ground	GND	GND
Signal	GPIO17	SIG

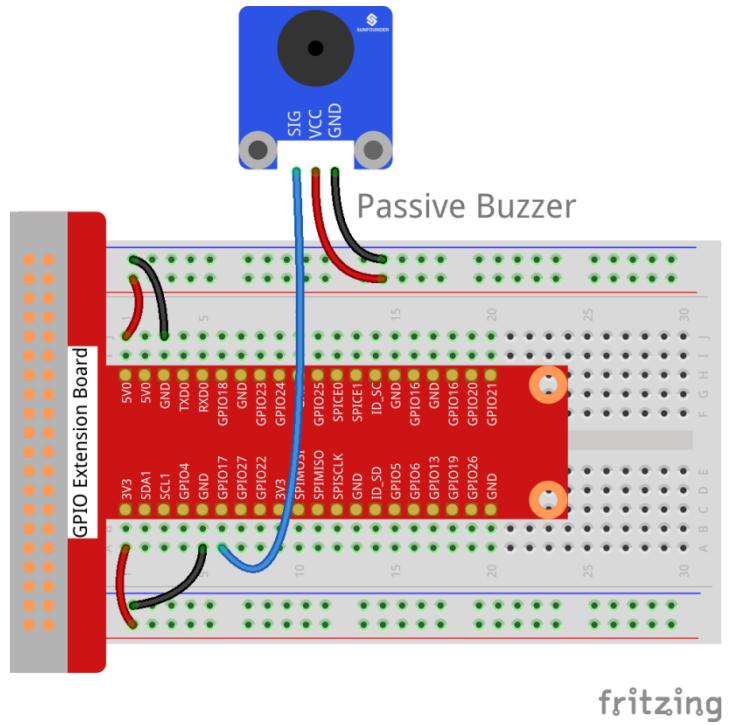


Abbildung 14: Verkabelung Buzzer

Daten Dem Buzzer können verschiedene Frequenzen angelegt werden. Somit auch das Spielen einer Melodie möglich. Für unsere Anwendung reicht jedoch eine einzelne Frequenz. Die Frequenz wird im Intervall von 300ms an den Buzzer angelegt und ist somit in der Lage die höchste mögliche Zahl "9" innerhalb von 2.7s abzuspielen.

Die Frequenz wird in Form einer Zahl (von 100 bis 1000) angegeben, der Frequenzbereich kann je nach Buzzer variieren.

Realisierung Der Code wird in Python realisiert und macht Verwendung von den Bibliotheken GPIO und time. Es wird in einem Intervall (300ms) eine Frequenz auf den GPIO Port ausgegeben. Alle Module/Komponente werden asynchron ausgeführt, das Ausführen von time.sleep(ms) sollte somit kein Problem sein.

Das Buzzern wird aus einer selbst implementierten "Sound"-Bibliothek über die Funktion "buzz_by_number(number)" ausgeführt. Dabei wird über eine Schnittstelle die gewünschte Nummer an den Buzzer gesendet.

3.6 Beschleunigung

Mithilfe elektronischer Komponenten kann Beschleunigung, Geschwindigkeit und Distanz (vom Startbereich) gemessen und analysiert werden. Zusätzlich kann eine approximierte Position des Zuges auf der Strecke berechnet werden. Die Berechnungen dazu sind in Kapitel 3.7.2 beschrieben.

Anforderungen

- Momentane Beschleunigung auslesen
 - Fahrtrichtung
 - Querbeschleunigung
- Beschleunigung in Geschwindigkeit und Distanz umrechnen
- Approximierte Position berechnen

Konzept Über einen Beschleunigungssensor werden Beschleunigung und Rotation ausgelesen. Die Beschleunigung kann zusätzlich in Geschwindigkeit und Distanz umgerechnet werden.

Komponente Bei der Komponentenwahl fiel der Entscheid auf einen Adafruit I^2C 3-Achsen Beschleunigungssensor, dieser kann Beschleunigung sowie Rotation berechnen. Er weist eine sehr kompakte Bauform auf und ist relativ günstig zu erwerben. Online haben verschiedene Nutzer mit dieser Komponente positive Erfahrung gesammelt. Auch ist die Komponente gut dokumentiert und man findet verschiedene Tutorien, wie man diese mit einem Raspberry Pi kombinieren kann.

Name	Adafruit ADXL 345
Preis	1Fr
Länge	25mm
Breite	19mm
Höhe	3.14mm
Gewicht	1.27g
Versorgungsspannung	3-5V

Tabelle 10: Technische Daten (<https://www.adafruit.com/product/1231>)

Bauplan / Interface Der Sensor wird über die I^2C Schnittstelle angesprochen und verwendet somit eine Datenleitung (SDA) und eine Clockleitung (SCL). Für die Stromversorgung werden die Anschlüsse für 3.3V des GPIO Headers benutzt.

Bezeichnung	GPIO Port	MPU 6050
Stromversorgung	3V3	VCC
Ground	GND	GND
Daten	SDA	SDA
Clock	SCL	SCL

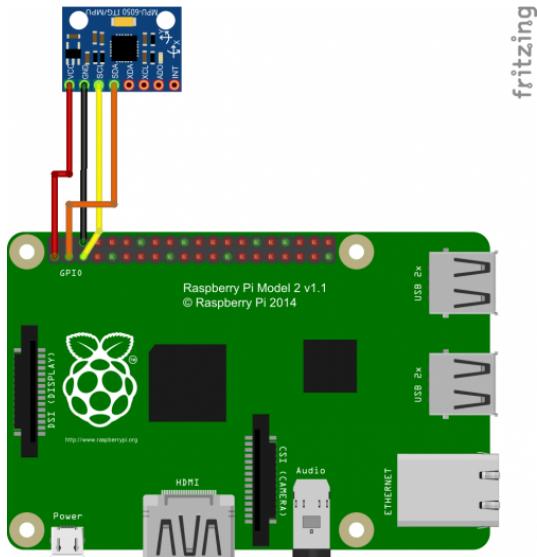


Abbildung 15: Verkabelung Beschleunigungssensor (<http://fritzing.org>)

Daten

Mithilfe eines Skripts erhalten wir eine gute Übersicht der erhaltenen Daten.

Gyroskop

```
gyroskop\_xout: -260  skaliert: -2
gyroskop\_yout: -154  skaliert: -2
gyroskop\_zout: 78    skaliert: 0
```

Beschleunigungssensor

```
beschleunigung\_xout: -1048  skaliert: -0.06396484375
beschleunigung\_yout: -676   skaliert: -0.041259765625
beschleunigung\_zout: 16644  skaliert: 1.01586914062
X Rotation: -2.32121150537
Y Rotation: 3.59994842011
```

Realisierung Daten werden in einem von uns festgelegten Intervall über die i^2c Schnittstelle eingelesen. Die i^2c Schnittstelle muss in der Raspberry Pi Konfiguration aktiviert werden. Weiter müssen die benötigten Tools «i2c-tools» sowie «python-smbus» installiert werden. Dem Raspberry Pi wird eine Adresse auf dem i^2c Datenbus zugewiesen, welche über «sudo i2cdetect -y 1» abgerufen werden kann.

Softwaretechnisch wird die Schnittstelle in Python realisiert. Python verfügt über mächtige Bibliotheken in den Bereichen Mathematik und i^2c . Der Beschleunigungssensor kann direkt über seine Adresse angesprochen werden und gibt die Daten in Form eines Words zurück. Der Beschleunigungssensor muss für jedes Datenwort (z.B. gyroskop_xout) eine Anfrage auf den Datenbus schreiben und lesen.

Die erhaltenen Daten werden in momentane Geschwindigkeitsdaten (Beschleunigung, Geschwindigkeit, Distanz) umgerechnet und anschliessend analysiert.

3.7 Elektronik Komponenten

In diesem Kapitel wird der Aufbau der Elektronik des Zuges beschrieben. Die Abbildung 16 veranschaulicht den Aufbau der Elektronik. Darauf sind alle logischen Verbindungen eingezeichnet. Weitere elektrische Verbindungen für die Stromversorgung werden im Kapitel 3.7.1 erläutert.

Zentral dabei ist der Mikrocontroller Tiny K22. Die Software auf dem Mikrocontroller initialisiert alle Komponenten, überwacht deren Status und sendet die nötigen Informationen an das Pi. Diese Schnittstelle ist detailliert im Kapitel 3.11 beschrieben.

Die Schnittstelle über den Debugger zum PC wird hier nicht weiter beschrieben, da diese Verbindung nur zum Entwickeln benutzt wird und für das endgültige Produkt nicht mehr von Bedeutung ist.

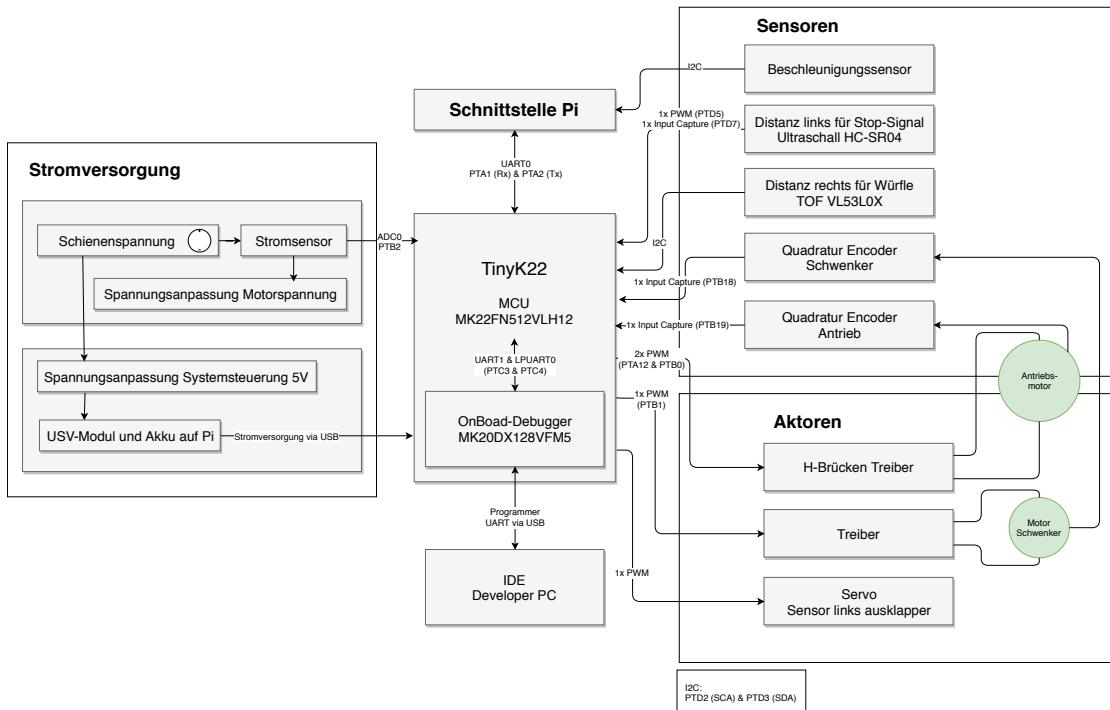


Abbildung 16: Komponentendiagramm Elektronik

3.7.1 Stromversorgung

Die Energie für das System wird über die Schienen bezogen. Die Antriebsenergie wird direkt von den Schienen bezogen und lediglich auf die korrekte Motorspannung angepasst. Die Systemsteuerung wird über ein StromPi 3 mit Strom versorgt. Als primäre Stromquelle wird dafür die Schienenspannung benutzt. Bei der Hochgeschwindigkeitsfahrt wird diese Quelle abgeschaltet, damit die gesamte Energie für den Antrieb zur Verfügung steht. Das StromPi schaltet automatisch auf die sekundäre Stromversorgung für das Pi um. Diese sekundäre Quelle wird durch einen LiFePO4-Akku auf dem StromPi realisiert. Sobald die primäre Stromquelle von den Schienen wieder zur Verfügung steht wird der Akku wieder nachgeladen.

Das Pi zero und das TinyK22 mit diversen Sensoren und Aktoren werden über die USB-Anschlüsse des Pi versorgt.

Antriebsenergie

Die Energie für den Antrieb wird direkt von den Schienen bezogen. Bei der Hochgeschwindigkeitsfahrt werden alle anderen Verbraucher (z.B. Raspberry PI) von den Schienen entkoppelt, um die gesamte Energie für den Antrieb nutzen zu können.

Verpolungsschutz

Auf den Schienen steht eine Gleichspannung zur Verfügung, wobei eine Schienenseite der + Pol und die andere Seite der – Pol ist. Daraus ergibt sich das Problem, dass beim Platzieren des Zuges entgegen der vorgesehenen Fahrtrichtung eine Verpolung stattfindet. Auch ist die Zuordnung der Pole in der Aufgabenstellung noch nicht spezifiziert. Um sicherzustellen, dass das System unabhängig von der Polung der Schienen funktioniert, muss eine Gleichrichtung realisiert werden. Dies kann mit einem Brückengleichrichter realisiert werden. (siehe Abbildung 17)

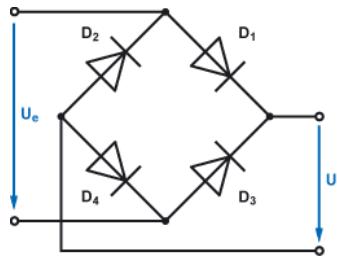


Abbildung 17: Brückengleichrichter (www.elektronik-kompendium.de)

Die Ausgangsspannung U_a hat dabei immer dieselbe Polarität, unabhängig welche Polarität U_e hat.

Für den Hochgeschwindigkeitszug wird ein "B40C 3700-2200" verwendet. Dabei handelt es sich um eine Integrierte Schaltung für einen Brückengleichrichter. Der maximal zulässige Strom ist dabei $3.7A$. Bei diesem Strom fällt über dem Gleichrichter eine Spannung von ca. $1V$ ab.

Spannungsanpassung

Gemäß der Aufgabenstellung stehen $20 \pm 2V$ bei bis zu $3A$ zur Verfügung. Aufgrund der grossen Toleranz von $4V$ muss die Spannung angepasst werden, um eine stabile Spannung sicherzustellen. Dafür wird ein DC-DC Converter verwendet. Dabei ist zu beachten, dass der Converter für den maximalen Strom von $3A$ ausgelegt ist.

Unterbrechungssicherheit

Um sicherzustellen, dass ein allfälliger Wackelkontakt der Schleifkontakte überbrückt werden kann und um grössere Spannungsschwankungen zu vermeiden, wird ein Kondensator zur Stützung eingesetzt.

Stromüberwachung

Um den Stromverbrauch des Antriebs zu überwachen, wird ein Strom Messwiderstand eingesetzt. Eine Differenzverstärker Schaltung bereitet die Spannung über dem Strom Messwiderstand auf, damit der Mikrocontroller mit dem Analog-Digital Wandler den Strom bestimmen kann. Mit dieser Information kann die Software des Mikrocontrollers auf Stromspitzen reagieren und z.B. die Geschwindigkeit reduzieren. Der Stromverbrauch ist auch für die Entwicklungs- und Testphase eine sehr wertvolle Information, damit das System optimal ausgelegt werden kann. Zur Messung wird ein "13FR200E - Strom Messwiderstand" verwendet. Dieser hat einen Widerstand von 0.2Ω . Der Strom kann

daraus mit dem Ohmschen Gesetz bestimmt werden.

$$I = \frac{U_R}{R} = \frac{U_R}{0.2\Omega}$$

Akku

Während der Hochgeschwindigkeitsfahrt wird die Systemsteuerung über einen Akku mit Storm versorgt. Die benötigte Leistung der einzelnen Komponenten ist in Tabelle 11 aufgeführt. Im schlimmsten Fall soll der Akku das System dabei mindestens für zwei komplette Abläufe mit Energie versorgen können.

Eine Runde darf maximal vier Minuten dauern. Das bedeutet die Energie des Akkus muss für mindestens acht Minuten reichen. Daraus ergibt sich für die Energie

$$E = P \cdot t = 14.2W \cdot 8min = 113.6Wmin = 1.9Wh$$

Komponente	Leistung (max)
Raspberry Pi 3 Model B	12.50W
Raspberry Pi zero W	1.20W
Tiny K22	0.10W
Encoder	0.07W
H-Brückentreiber	0.02W
Treiber Schwenker Motor	0.5mW
Beschleunigungssensor	0.3mW
TOF-Sensor	20mW
Ultraschall Sensor	75mW
Total	14.20W

Tabelle 11: benötigte Leistung der Komponenten

Das StormPi Modul wird mit einem LiFePO4-Akku ergänzt. Der verbaute Akku hat $1000mAh$. Bei einer Spannung von $3.7V$ entspricht dies $3.7Wh$. Dies ist genügend für zwei Durchläufe. Ausserdem wird der Akku ausserhalb der Hochgeschwindigkeitsfahrt nachgeladen, was eine zusätzliche Reserve ergibt.

3.7.2 Sensoren

Mit diversen Sensoren sollen folgende Daten aufgenommen werden.

- Beschleunigung
- Geschwindigkeit
- Position
- Distanz rechts (für Erkennung Würfel)
- Distanz links (für Erkennung Haltesignal)

Beschleunigung

Die Beschleunigung wird mit einem Beschleunigungssensor ADXL345 aufgenommen. Dieser kommuniziert über eine I^2C Schnittstelle mit dem Mikrocontroller. Er liefert jeweils die Beschleunigung in x-, y-, und z-Richtung. Der Sensor wird parallel zur Fahrtrichtung montiert, so dass es genügt, eine Beschleunigungsrichtung für die Querbeschleunigung und eine Richtung für die Längsbeschleunigung auszuwerten. Die Verwendung des Sensors ist in Kapitel 3.6 detailliert beschreiben.

Geschwindigkeit

Die Geschwindigkeit wird hauptsächlich über den Quadratur Encoder am Antriebsmotor aufgenommen. Zusätzlich wird die Geschwindigkeit über die Beschleunigung zur Kontrolle nachgerechnet.

Quadratur Encoder: Der Quadratur Encoder MR, Typ L gibt 1024 Impulse pro Umdrehung. Der Verlauf eines Impulses ist in Abbildung 18 dargestellt. Der Encoder stellt drei Kanäle zur Verfügung. Über die Kanäle A und B kann einzeln die Geschwindigkeit bestimmt werden. Durch Auswerten der Phasenverschiebung der beiden Kanäle ("A eilt B vor" oder "A eilt B nach") kann zusätzlich noch die Drehrichtung bestimmt werden. Der Kanal I ist mit Kanal A und B synchronisiert und kann ebenfalls zur Bestimmung der Geschwindigkeit dienen.

Für die Bestimmung der Geschwindigkeit kann die Dauer zwischen zwei Impulsen gemessen werden, oder es können die Anzahl Impulse in einer bestimmten Zeit gezählt werden. Für dieses Projekt soll die Dauer des Impulses gemessen werden. Der Vorteil dieser Methode ist die bessere Präzision, da nach jedem einzelnen Impuls die durchschnittliche Geschwindigkeit seit dem letzten Impuls sofort bestimmt werden kann. Das Risiko ist, dass bei hohen Umdrehungszahlen der Mikrocontroller nicht schnell genug ist mit dem Zählen, oder dass der Zähler des Mikrocontrollers bei sehr tiefen Umdrehungszahlen überläuft.

Die maximale Umdrehungszahl des Antriebsmotors für die angestrebte Maximalgeschwindigkeit liegt bei 5200min^{-1} . Mit 1024 Impulsen pro Umdrehung ergibt dies

$$5200\text{min}^{-1} \cdot 1024\text{Impulse} = 5'324'800 \frac{\text{Impulse}}{\text{min}}$$

Dies sind dann

$$5'324'800 \frac{\text{Impulse}}{\text{min}} / 60\text{s} = 88'747 \frac{\text{Impulse}}{\text{s}}$$

Dies ergibt eine Impulsdauer von

$$\frac{1}{88'747 \frac{\text{Impulse}}{\text{s}}} = 11.29\mu\text{s}$$

Bei der schnellsten möglichen Timer-Einstellung auf dem TinyK22 (60MHz) ergibt dies noch $676 \frac{\text{Ticks}}{\text{Impuls}}$. Bei der Implementierung muss also beachtet werden, dass bei hoher Geschwindigkeit möglichst wenig Zeit in der Interrupt Routine verbracht wird, da diese dann sehr oft aufgerufen wird. Sollte sich zeigen, dass der Mikrocontroller durch die hohe Impulsrate überlastet ist, muss ein Encoder mit weniger Impulsen pro Umdrehung gewählt werden oder ein Frequenz Teiler dazwischen geschaltet werden.

Damit der Timer nicht überläuft, muss eine Impulspause fertig sein, bevor der Timer den Wert

$$2^{32} \approx 4.29\text{Mrd}$$

erreicht. Dies ergibt bei $60MHz$ eine Zeit von

$$2^{32} \cdot \frac{1}{60MHz} = 71.58s$$

Also ist ein tiefstmögliche Umdrehungszahl

$$\frac{1}{71.58s \cdot 1024} = 13.6 \cdot 10^{-6}s^{-1}$$

Diese Umdrehungszahl sollte nicht unterschritten werden. Diese Zahl ist jedoch so klein, dass dies als Stillstand gewertet werden kann.

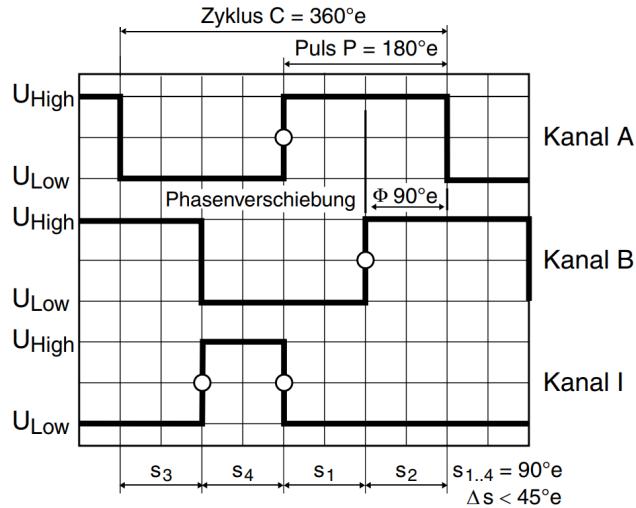


Abbildung 18: Signalverlauf Encoder (www.maxonmotor.ch)

Die Geschwindigkeit des Zuges (v_{Zug}) kann somit aus der Umdrehungszahl des Motors (N_{Motor}) mittels der Mechanischen Übersetzung und der Grösse der Räder berechnet werden.

$$v_{Zug}(N_{Motor}) = \frac{N_{Motor}}{2} \cdot d \cdot \pi$$

d : Raddurchmesser $22mm$

Nachrechnen der Geschwindigkeit: Unter der Annahme, dass zu Beginn der Messung zum Zeitpunkt $t = 0$ die Geschwindigkeit 0 ist ($v(t = 0) = 0$), kann die Geschwindigkeit zum Zeitpunkt t bestimmt werden mit

$$v(t) = \int_0^t a(x)dx$$

Da aber auf einem Digitalen System die Daten nur zu diskreten Zeitpunkten ausgewertet werden können, ergibt sich dann eine Summe der Beschleunigungen zum Zeitpunkt k

$$v[k] = \sum_{i=0}^k a[i]\Delta t$$

Position

Über den Beschleunigungssensor kann man die aktuelle Position auf der Fahrbahn berechnen. Die zurückgelegte Strecke errechnet sich durch Integration der Geschwindigkeit oder durch zweifache Integration der Beschleunigung.

Unter der Annahme, dass zu Beginn der Messung zum Zeitpunkt $t = 0$ die zurückgelegte Strecke 0 ist ($s(t = 0) = 0$), kann die Geschwindigkeit zum Zeitpunkt t bestimmt werden mit

$$s(t) = \int_0^t v(x)dx$$

Da aber auf einem Digitalen System die Daten nur zu diskreten Zeitpunkten ausgewertet werden können, ergibt sich dann eine Summen der Geschwindigkeiten zum Zeitpunkt k

$$s[k] = \sum_{i=0}^k v[i]\Delta t$$

Die Geschwindigkeit wird gemäss der Beschreibung oben bestimmt.

Distanz

Auf beiden Seiten des Zuges wird eine Distanzmessung benötigt. Auf der rechten Seite des Zuges muss der Würfel erkannt werden, und auf der linken Seite soll ein ausklappbarer Sensor am Schluss die genaue Distanz zum Haltesignal bestimmen.

Würfelerkennung: Gemäss der Aufgabenstellung befindet sich der Würfel in einem Abstand von $8 \pm 1\text{cm}$ von der Gleismitte. Somit muss der Distanzsensor Distanzen zwischen ca. 20mm und 80mm erkennen können. Der exakte Wert der Distanz ist dabei nicht entscheidend, da nur ein bestimmter Schwellwert erkannt werden muss. Die Distanz zum Würfel wird mit einem TOF (Timo-Of-Flight) Sensor VL53L0X ermittelt.

Haltesignal Erkennung: Um möglichst präzise anhalten zu können, muss die Systemsteuerung den exakten Wert der Distanz zum Haltesignal kennen. Dies wird mit einem Distanzsensor ermittelt. Diese Distanz wird mit einem Ultraschall Sensor HC-SR04 gemessen. Es ist entscheidend, dass der Sensor korrekt und mit wenig Toleranz auf der Mechanik befestigt wird, um eine exakte Ausrichtung auf das Haltesignal sicherzustellen.

3.7.3 Aktoren

Die Aktoren stellen die Schnittstelle zur Mechanik dar. Diese sollen alle nötigen mechanischen Bewegungen auf Befehl des Mikrocontrollers ausführen.

H-Brücken Treiber für Antriebsmotor

Um den Antriebsmotor anzusteuern, wird ein H-Brückentreiber verwendet. Damit kann der Motor durch ein PWM Signal in der Geschwindigkeit fast beliebig eingestellt werden. Über die wahlweise Ansteuerung einer der beiden Eingänge der H-Brücke wird die Richtung bestimmt.

Es wird ein Arduion IBT _ 2 DC-Motoren Treiber mit einem BTS7960 eingesetzt. Dieses

Bauteil kann Motoren mit einem Strom von bis zu 43A versorgen.

Antriebsmotor

Als Antriebsmotor dient ein Maxon DCX 32 L. Dieser kann eine Leistung von bis zu 70 Watt umsetzen. Dabei zu beachten ist, dass der Anlaufstrom bis zu 70A betragen kann. Dieser Strom muss begrenzt werden, indem der Mikrocontroller die Beschleunigung gemäss dem gemessenen Stromverbrauch anpasst. Maxon [2018a] Dies ist bei einer Versorgungsspannung von 24V spezifiziert. Da aber nicht die maximale Drehzahl benötigt wird, kann auch eine entsprechend tiefere Spannung angelegt werden. Die Drehzahlkonstante des Motors ist $350 \frac{\text{min}^{-1}}{\text{V}}$. Für die angestrebte Drehzahl von 5200min^{-1} ergibt sich eine Spannung von

$$\frac{5200 \text{min}^{-1}}{350 \frac{\text{min}^{-1}}{\text{V}}} = 14.8 \text{V}$$

Somit muss der Antriebsmotor mit einer Spannung von mindestens 14.8V versorgt werden.

Motortreiber für Schwenker-Motor

Da die Richtung immer dieselbe ist reicht für den Schwenker ein normaler DC-Motoren Treiber. Um eine sanfte Beschleunigung und Bremsung der Konstruktion zu ermöglichen, soll auch der Schwenker-Motor mit einem PWM angesteuert werden. Als Treiber wird ein Board mit einem L298N verwendet.

Schwenker-Motor

Für den Schwenker wird ein Maxon DCX 19 S verwendet. Um die Position des Schwenkers zu bestimmen, ist auch an diesem Motor ein Encoder befestigt. Durch die Bestimmung der nötigen Umdrehungen, bis der Schwenker eingefahren ist, kann genau festgestellt werden, wie viele Impulse abgewartet werden müssen, bis der Motor anhalten muss. Maxon [2018b]

Distanzsensor links ausklappen

Um den Sensor beim Parkieren auszuklappen zu können, wird dieser an einem Servo befestigt. Sobald die Systemsteuerung entscheidet, dass das korrekte Haltesignal das Nächste ist, gibt sie den Befehl den Sensor auszuklappen. Dafür wird ein "Tower Pro Micro Servo SG90" verwendet. Dieser hat ein Drehmoment von bis zu $2.5 \text{kg} - \text{cm}$. Mi-PC [2016]

3.7.4 Proof-of-Concept

Um die Funktionsfähigkeit der einzelnen Komponenten sicherzustellen, wurden diverse Tests gemacht. Das Ziel dieser Tests war es, die einfachste Form jeder Funktionalität zu realisieren. Auf eine quantitativ genaue Anpassung wurde in diesem ersten Schritt verzichtet, dies ist für die Realisierung des Projekts vorgesehen. Das Ziel der Tests war somit nur, zu zeigen, dass die Komponente wie vorgesehen funktionieren kann. Eine Übersicht der durchgeführten Tests ist in Tabelle 12 ersichtlich.

Komponente	Beschreibung des Tests	Ok
Tiny K22	Der Mikrocontroller konnte erfolgreich programmiert werden und Hardware kann darüber angesteuert werden.	✓
UART Kommunikation	Mittels dem "Raspberry Pi APROG HAT" konnte erfolgreich zwischen dem Pi und dem Tiny mittels UART kommuniziert werden.	✓
Beschleunigungssensor	Der Beschleunigungssensor wurde über die Schnittstelle angesteuert und die Werte für die x-, y- und z-Richtung konnten erfolgreich ausgelesen werden.	✓
Motorenansteuerung	Vom Mikrocontroller wurde ein PWM generiert. Der Pin mit dem PWM wurde mit dem Motorentreiber Arduion IBT 2 und dieser dann mit einem Motor verbunden. Durch Vorgaben des Mikrocontrollers konnte die Geschwindigkeit und die Richtung des Motors eingestellt werden.	✓
Encoder	Der Encoder wurde vom Mikrocontroller ausgelesen und die Zeiten und Impulse ausgewertet. Es zeigte sich, dass der Encoder korrekt 1024 Impulse pro Umdrehung liefert. Daraus lässt sich dann die Umdrehungszahl berechnen. Auch wurde eine Zeitmessung durchgeführt, um festzustellen wie viel Zeit die Interrupt-Routine von der Flanke des Encoders, bis die Routine beendet ist, benötigt. Hochgerechnet für die maximale Geschwindigkeit ergibt dies eine Auslastung des Mikrocontrollers von ca. 10%.	✓
Ultraschallsensor	Der Ultraschallsensor wurde vom Mikrocontroller gemäß dem Datenblatt angesteuert. Mittels der gemessenen Zeit konnte eine approximative Distanz berechnet werden. Für einen exakten Wert sind noch genauere Abstimmungen nötig. Der Sensor reagiert jedoch zuverlässig auf Objekte in der Grösse des Würfels oder der Haltesignale.	✓
TOF-Sensor	Mit einer Code-Bibliothek, die für das Tiny K22 angepasst wurde, konnte der TOF-Sensor erfolgreich angesteuert werden. Damit kann eine approximative Distanz berechnet werden. Für einen exakten Wert sind noch genauere Abstimmungen nötig. Der Sensor reagiert jedoch zuverlässig auf Objekte in der Grösse des Würfels oder der Haltesignale.	✓
Spannungsanpassung	Der DC-DC Converter wurde mit einer Spannungsquelle verbunden und die Ausgangsspannung bei diversen Einstellungen korrekt gemessen.	✓
Stromüberwachung	Eine Differenzverstärker Schaltung wurde mit unterschiedlichen Lastwiderständen simuliert und durch die Messspannung den Strom bestimmt und mit dem Strom der Simulation verglichen.	✓

Tabelle 12: Übersicht Proof-of-Concept Elektronik

3.8 Signalerkennung

In der Aufgabenstellung wird gefordert, dass der Schnellzug während der Bewältigung der Strecke ein Signal mit aufgedruckter Nummer erkennt. Die Nummer ist auf einer 3x3cm grossen Tafel mit weissem Hintergrund, schwarz aufgedruckt oder weiss auf schwarzem Hintergrund. Die Aufgabe Signalerkennung wird in zwei Teilaufgaben unterteilt:

- Erkennung der Signalkennung mit Tafel
- Erkennung der aufgedruckten Nummer

Wie bereits in der Übersicht beschrieben, werden im Gesamtkonzept zwei Kameras verwendet. Eine Kamera wird zur Erkennung der Schienenrichtung verwendet. Die zweite Kamera wird für die Signalerkennung eingesetzt. Der Raspberry PI 3+, welcher als Hauptrecheneinheit geplant ist, verfügt nur über einen CSI-Anschluss (Camera Serial Interface). Für die Signalerkennung wird auf einen weiteren kleineren Raspberry PI zero gesetzt. Dieser verfügt wie der Raspberry PI 3+ einen vollwertigen CSI-Anschluss und kann somit die zweite Kamera bedienen. Dies führt auch dazu, dass der Raspberry PI 3+ zusätzlich entlastet werden kann.

Architektur

Die Signalerkennung in zwei Teilaufgaben zu trennen hat noch einen weiteren Vorteil. Die beiden Teilaufgaben werden auf beide Raspberry PIs verteilt. Der Raspberry PI zero ist für die Bildaufnahme Verarbeitung und für die Signalerkennung mit Tafel zuständig. Der Raspberry PI 3+ übernimmt dann die Erkennung der aufgedruckten Nummer auf der Tafel. So können die Ressourcen des Raspberry PI 3+ gezielter eingesetzt werden.

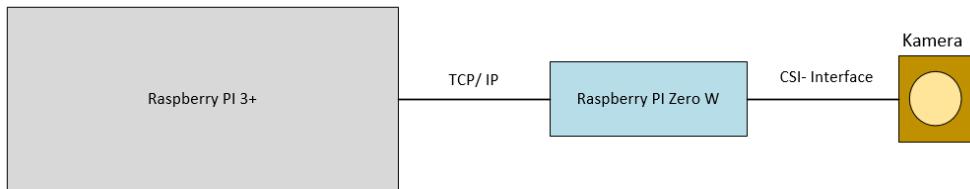


Abbildung 19: Architektur Hardware Signalerkennung

Für die Nummernerkennung auf der Tafel wird ein Machine-Learning Algorithmus verwendet. Damit die Trainingsdaten des Modells eingelesen werden können und somit ausführbar werden, braucht es eine gewisse Grösse des Arbeitsspeichers. Der Arbeitsspeicher des Raspberry PI zero (512Mb) reicht dafür nicht aus. Der Raspberry PI 3+ verfügt über mehr Speicher (1Gb) und kann somit den Algorithmus bearbeiten. Für die Kameraaufnahmen und die Bearbeitung der Bilder und schlussendlich die Konturenerkennung der Signalkennung ist der Raspberry PI zero gut geeignet.

Software Tafelerkennung

Für die Erkennung der Tafel wird auf das Framework OpenCV gesetzt. OpenCV hat sich als Standardframework im Bereich der Echtzeitbildverarbeitung durchgesetzt. Weiter zeichnet sich OpenCV in seiner Effizienz und seiner breiten Community aus. In der Abbildung 20 ist der Ablauf der Tafelerkennung und in der Tabelle 13 die jeweilige Funktion beschrieben.

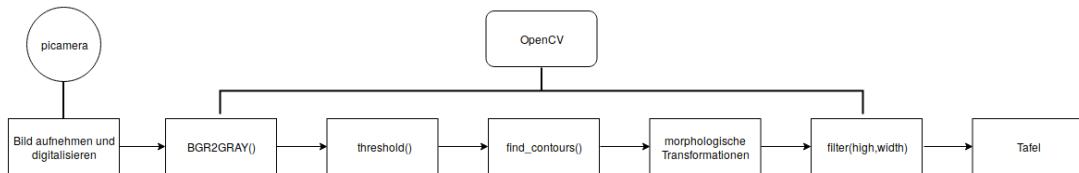


Abbildung 20: Ablauf Tafelerkennung

Funktion	Beschreibung
Bild aufnehmen und digitalisieren	Die Umgebung wird mittels der Raspberry PI Kamera und der Bibliothek picamera aufgenommen und dem OpenCV übergeben
BGR2GRAY	Die Farbinformationen des Bildes werden nicht benötigt und deshalb entfernt (Effizienz)
threshold	Für die Konturenerkennung braucht es nur die Schattierungen des Bildes. Aus diesem Grund wird ein Threshold auf das Bild gelegt, damit ein Binary-Picture generiert werden kann.
find-contours	Nun wird die OpenCV Funktion find-contours angewendet. Dabei werden benachbarte schwarze oder weisse Pixel miteinander zusammengefügt, bis eine Kontur entstehen kann.
morphologische Transformationen	Hier wird das Bild mittels morphologische Transformationen nachbearbeitet
filter	Die erkannten Konturen werden gefiltert nach Grösse und Form der Tafel.
Tafel	Nun ist das Bild mit der Tafel erkannt worden und wird zur Nummererkennung vorbereitet.

Tabelle 13: Beschreibung der OpenCV Funktionen

In der Abbildung 21 links sieht man die erkannte Position der Nummer auf der Tafel. Rechts sieht man die gleiche Aufnahme nach der Filterung. Dieses Bild wird nun auf die Grösse des rechten erkannten Rechteckes zugeschnitten und dem Raspberry PI 3+ zur Nummererkennung übergeben.

Konfiguration Raspberry PI Kamera

Eine weitere Schwierigkeit bei der Nummererkennung ist die Aufnahme mittels Kamera. Bei hohen Geschwindigkeiten neigt das Bild dazu unscharf zu werden. Dies liegt daran, dass die Verschlusszeit der Kamera zu lange ist. Mit der Raspberry PI Kamera und der Schnittstelle piCamera können viele Parameter eingestellt werden. Hier liegt der Schlüssel zur Behebung von unscharfen Bildern. Wenn Parameter wie Belichtungszeit, Helligkeit, Sättigung und Weissabgleich fest eingestellt werden, können in erster Linie

reproduzierbare Bilder aufgenommen und die Belichtungszeit kann auf ein Minimum gesetzt werden. Wenn die Belichtungszeit kürzer gesetzt wird, ergibt dies aber dunklere Bilder. Mit digitaler Aufhellung oder sogar mit externer Beleuchtung kann aber diesem Effekt entgegengewirkt werden.



Abbildung 21: Nummerposition- und Nummererkennung

Software Nummererkennung

Damit nun die Nummer auf der Tafel erkannt werden kann, wird ein Machine-Learning Algorithmus verwendet. Dafür wird die high-level API Keras verwendet. Als Backend kommt Tensorflow zum Einsatz. Als Trainingsdatabase kommt die frei Verfügbare MNIST- Database zum Einsatz. Die MNIST- Datenbank verfügt über 60'000 Zahlen in Handschrift. Die Tabelle 14 zeigt die Zusammenfassung der Tensorflowbackend bezüglich dem ausgewählten Keras Model. In den ersten Tests hat die Erkennung gut funktioniert auch auf dem Raspberry PI. Probleme sind in erster Linie mit der Nummer "1" aufgetaucht. Die MNIST Datenbank verfügt nur über die amerikanisch geschriebene "1". Die "1" in Arial wird dadurch nicht erkannt. Sobald in PREN2 die Schriftart der Zahlen bekannt gegeben wird, muss eventuell ein eigener Trainingsdatensatz erstellt werden oder auf einen anderen Datensatz, wie zum Beispiel von UCI Machine Learning Repository, zurückgegriffen werden.

Layer(type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 26, 26, 32)	320
conv2d_2 (Conv2D)	(None, 24, 24, 64)	18496
max_pooling2d_1 (MaxPooling2)	(None, 12, 12, 64)	0
dropout_1 (Dropout)	(None, 12, 12, 64)	0
flatten_1 (Flatten)	(None, 9216)	0
dense_1 (Dense)	(None, 128)	1179776
dropout_2 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 10)	1290

Tabelle 14: Auflistung aller Testklassen mit jeweiligen Testarten

3.9 Gleiserkennung

Ein Beschleunigungssensor für die Erfassung der Fahrdaten ist bereits in der Aufgabenstellung vorgeschrrieben. Um aber eine ideale Regelung zu ermöglichen, wird zusätzlich eine Gleiserkennung implementiert. Mit welchem schon im Voraus erkennt werden kann ob eine Kurve bevorsteht oder ob sich der Hochgeschwindigkeitszug auf einer Geraden befindet. Während der Konzept Phase wurden verschiedene Verfahren evaluiert und zum Teil ausprobiert. Jedoch gab es keine Lösung, welche zusätzlich zur Richtung noch den Radius der Kurve bestimmen kann. Die momentane Lösung, welche angestrebt wird, kann also nur sagen, ob es geradeaus, nach links oder nach rechts geht. Diese Information wird in den Regelungskreis eingespeist. Dadurch wird die maximale Geschwindigkeit für den engsten Schienen-Radius festgelegt. Anschliessend wird mithilfe des Beschleunigungssensors versucht, an das Limit der Zentripetalkraft zu gelangen. Somit kann die bestmögliche Regelung implementiert werden, ohne genauere Angaben zum aktuellen Kurvenradius zu haben. Mehr Details zum Regelkreis und dem ganzen Ablauf sind im unter 3.1 zu finden.

3.9.1 Verfahren

Nachfolgend werden die Schritte zur Bearbeitung eines einzelnen Bildes aufgelistet. Die einzelnen Teilschritte werden für jedes Bild, welches aufgenommen wird, neu berechnet.

- Bild skalieren
- Umrechnen zu Schwarz-Weiss
- Kantenglättung anwenden (Canny Algorithmus)
- Konturen erkennung (Suzuki85 Algorithmus)
- Kleine Konturen erkennen und verwerfen
- Bild in zwei Hälften unterteilen
- Berechnen der Anzahl Konturen in beiden Hälften
 - Beide Hälften etwa gleich viele Konturen \Rightarrow Gleis geht geradeaus
 - Linke Hälfte hat mehr Konturen \Rightarrow Gleis geht nach links
 - Rechte Hälfte hat mehr Konturen \Rightarrow Gleis geht nach rechts

Bild skalieren

Das Bild muss für die Berechnungen runter skaliert werden, um die Rechenzeit zu reduzieren. Das Format wird dabei beibehalten, um Verzerrungen zu vermeiden. Zuerst wird also das Seitenverhältnis berechnet, um dann die kürzere Seite auf einen konfigurierbaren Wert zu setzen. Die längere Seite wird dann mit dem Seitenverhältnis Faktor berechnet.

Umrechnen zu Schwarz-Weiss

Der Canny Algorithmus für die Kantenglättung muss mit einem Schwarz-Weiss Bild gefüttert werden. Deshalb wird das farbige Bild zu einem Schwarz-Weiss Bild umgerechnet.

Kantenglättung anwenden (Canny Algorithmus)

Der Canny Algorithmus sucht Kanten im Bild und erzeugt ein binäres Bild (Schwarz oder Weiss), wobei die weissen Pixel eine Kante beschreiben.

Konturen erkennung (Suzuki85 Algorithmus)

Das binäre Bild, welches mithilfe vom Canny Algorithmus erzeugt wird, kann mit dem Suzuki85 Algorithmus analysiert werden, um Konturen zu erkennen. Die Konturen sind jeweils eine Liste von Punkten, welche zusammen eine Kontur bilden.

Kleine Konturen erkennen und verwerfen

Die Anzahl Punkte einer Kontur werden mit einem Schwellwert verglichen. Sind die Anzahl Punkte innerhalb der Kontur zu gering, wird sie verworfen. Somit können kleine Störungen im Bild bzw. kurze Linien ignoriert werden.

Bild in zwei Hälften unterteilen

Das Bild wird nun in zwei Hälften geteilt, um die Entscheidung zu treffen ob geradeaus, nach links oder nach rechts gefahren wird.

Berechnen der Anzahl Konturen in beiden Hälften

Um die Entscheidung zu treffen, wird einfach die Anzahl Punkte in jeder Hälfte verglichen. Hat es in der linken Hälfte mehr, geht das Gleis nach Links. Sind es in der rechten Hälfte mehr, geht es nach Rechts. Ist die Anzahl ungefähr identisch, dann geht es geradeaus. Da es nie genau gleich viele Punkte haben wird, gibt es eine minimal Differenz, die es zwischen links und rechts geben muss, damit eine Kurve detektiert wird.

3.10 Aufgabentrennung zwischen Pi und Tiny

Dieses Kapitel beschreibt die Aufgabentrennung zwischen der Systemsteuerung auf dem Raspberry Pi 3+ (im folgenden Pi genannt) und dem Mikrocontroller MK22FN512VLH12 auf dem Entwicklerboard TinyK22 (im folgenden Tiny genannt). Das Pi dient im System als Master. Damit fällt das Pi alle Entscheidungen. Das Tiny dient als Slave, es führt die Entscheidungen vom Pi aus und gibt Rückmeldung zum aktuellen Status und zu den Sensordaten.

Im System sind folgende Aufgaben für die jeweiligen Steuerungen vorgesehen:

Pi:

- Entscheidung Start gemäss Befehl vom Webinterface
- Auswertung der Kameraaufnahmen
 - Entscheidung über bevorstehende Kurven
 - Entscheidungen gemäss erkannter Schilder
- Entscheidung der Fahrgeschwindigkeit
- Auswertung des Beschleunigungssensors
- Versenden der Sensordaten an das Webinterface

Tiny:

- Ansteuerung / Regelung Antriebsmotor
- Ansteuerung Schwenkermotor
 - inkl. Auswertung Position des Schwenkers bis vollständig eingefahren
- Auslesen von Sensordaten
 - Objekterkennung Würfel
 - Objekterkennung Haltesignal
 - Stromverbrauch
 - aktuelle Ist-Geschwindigkeit

Somit ist jedes System auf Informationen des anderen angewiesen. Deshalb ist eine klare Definition der Schnittstelle der beiden Komponenten nötig. Im folgenden Kapitel 3.11 wird diese Interface genauer beschrieben.

3.11 Interface zwischen Pi und Tiny

Dieses Kapitel beschreibt die Kommunikation zwischen der Systemsteuerung auf dem Raspberry Pi und dem Mikrocontroller MK22FN512VLH12 auf dem Entwicklerboard TinyK22. Dabei sollen Informationen zur aktuellen Situation, sowie auch Informationen zum aktuellen Status des jeweiligen Systems ausgetauscht werden.

Diese Kommunikation soll es den Systemen ermöglichen, die zugewiesenen Aufgaben gemäss Kapitel 3.10 zu erfüllen.

3.11.1 Hardware Schnittstelle

Als Hardware Schnittstelle wird UART (auch RS-232 genannt) verwendet. Dies ist eine asynchrone serielle Schnittstelle. Die Kommunikation kann mit zwei Leitungen realisiert werden. Eine dient als Empfangsverbindung (Rx) und eine als Sendeverbindung (Tx). Diese Trennung erlaubt eine voll-duplexe Kommunikation.

3.11.2 Übertragungsprotokoll

In regelmässig wiederholenden Zeitpunkten werden Datenpakete (im folgenden Frame genannt) in einem fest vorgelegten Format ausgetauscht. Dabei gibt es ein Grundformat für die Informationen und zwei Formate für den Informationsinhalt, eines für Frames von Pi zum Tiny und ein anderes Format für die Frames vom Tiny zum Pi. Der Informationsinhalt der beiden Formate unterscheidet sich gemäss der Aufgabentrennung in Kapitel 3.10.

Grundformat

Die Frames bestehen jeweils aus einem Bezeichner (Key) und einem Wert (Value). Diese werden mit einem Komma getrennt. Als Abschluss dient das New Line Zeichen '\n'. Somit sieht ein Informationsstück folgendermassen aus:

$$\{Key\}, \{Value\}\n$$

Zum Beispiel würde eine Information zur Geschwindigkeit folgendermassen aussehen:

$$speed, 200$$

Die gesamten Informationen werden als Zeichenkette (String) im Ascii Format verschickt. Zahlenwerte werden jeweils vor dem Senden in einen String umgewandelt und nach dem Empfangen wieder in einen Zahlenwert zurückkonvertiert. Dies verbessert die Leserlichkeit, was zum Testen und Simulieren der Kommunikation sehr hilfreich sein kann.

Frames: Pi \Rightarrow Tiny

Das Pi schickt dem Tiny eine vorgegebene Soll-Geschwindigkeit und Richtung der nächsten Kurve. Auch Informationen über den aktuellen Status des Pi werden verschickt. Die genauen Bezeichner und die Bedeutung und Grösse der zugehörigen Werte sind in Tabelle 15 aufgelistet.

Bezeichner	Wert Beschreibung	Wert Zahlengrösse
speed	Soll-Geschwindigkeit	signed 32-Bit Integer
dir	Kurvenrichtung	signed 32-Bit Integer
status	Moral und weitere Informationen des Pi	unsigned 8-Bit Integer

Tabelle 15: Kommunikation Frames: Pi \Rightarrow Tiny

Frames: Tiny \Rightarrow Pi

Das Tiny nimmt die nötigen Sensordaten auf und schickt die nötigen Informationen dar-aus dem Pi. Auch Informationen über den aktuellen Status des Tiny werden verschickt. Die genauen Bezeichner und die Bedeutung und Grösse der Zugehörigen werte sind in Tabelle 16 aufgelistet.

Bezeichner	Wert Beschreibung	Wert Zahlengrösse
is-speed	Ist-Geschwindigkeit (gemäss Encoder)	signed 32-Bit Integer
obj	Objekt erkannt (einzelne Bits 1 oder 0)	unsigned 8-Bit Integer
status	Moral und weitere Informationen es Tiny	unsigned 8-Bit Integer

Tabelle 16: Kommunikation Frames: Tiny \Rightarrow Pi

Der Wert der Moral ist bestimmt für Statusinformationen, welche den einzelnen Bits in der Zahl zugeordnet werden. Der genaue Inhalt kann sich je nach Status (gemäss Kapitel 3.1) unterscheiden. Der nötige Informationsinhalt, um den Ablauf zu erfüllen, ist in Tabelle 17 aufgelistet. Für Tests soll es möglich sein, den Inhalt je nach Bedarf zu erweitern.

Informationen Pi \Rightarrow Tiny	Informationen Tiny \Rightarrow Pi
Bereit	Bereit
Ablauf gestartet	Würfel erkannt
Befehl zum Schwenker einfahren	Schwenker vollständig eingefahren
Haltesignal erkannt (Parksensor ausklappen)	Haltedistanz erreicht
	Stromverbrauch zu hoch

Tabelle 17: Informationsinhalt des Statusbytes

3.12 Software Lösungskonzepte

Die Steuerungssoftware, welche auf dem Hauptrechner ausgeführt wird, soll in mehrere Teilprogramme aufgeteilt werden. Teilprogramme können so unabhängig voneinander implementiert und getestet werden. Um anschliessend zwischen den Teilprogrammen zu kommunizieren, wird eine Middleware verwendet. Die Middleware kann auch genutzt werden, um die einzelnen Teilprogramme zu testen. Zusätzlich können verschiedene Programmiersprachen für die einzelnen Teilprogramme verwendet werden. Dies erhöht zusätzlich die Flexibilität während der Entwicklung.

3.12.1 Architektur

Im folgenden Diagramm wird die Architektur der Steuerungssoftware aufgezeigt

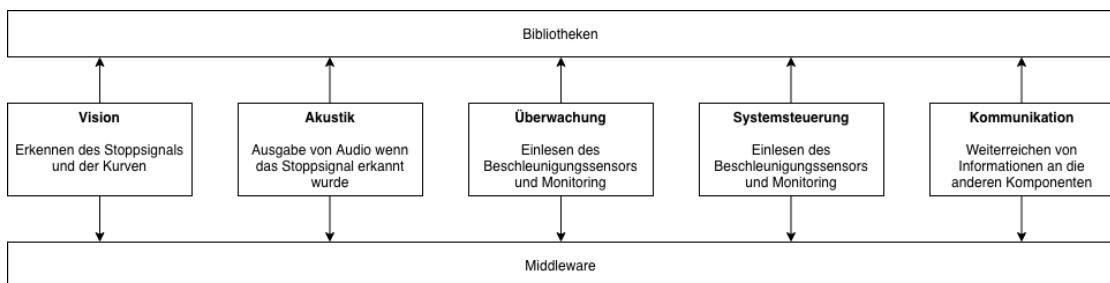


Abbildung 22: Software Architektur Middleware. Gezeichnet mit <https://draw.io>

Legende:

- Die Pfeile visualisieren die Abhängigkeiten innerhalb der Architektur

3.12.2 Technologie

Während der Technologierecherche wurde eine Analyse gemacht, um zu entscheiden welche Technologie eingesetzt werden soll. Bald wurde ZeroMQ auserkoren, da diese schlank ist und ohne Performance Probleme auf kleinen Einplatinenrechner läuft. Ausserdem ist die Middleware bekannt und hat eine hervorragende Dokumentation. Um die Daten, welche über die Middleware geschickt werden, mit verschiedenen Programmiersprachen zu nutzen, wird Protobuffers verwendet. Protobuffers ist ein Format zur Beschreibung von Daten. Dieses Format kann anschliessend verwendet werden, um Klassen bzw. Funktionen für fast jede Programmiersprache zu benutzen. Das heisst, dass die Daten einmalig definiert werden und anschliessend von allen Teilprogrammen verwendet werden können.

Beispiel Protobuf:

```
syntax = "proto3";  
  
message Direction {  
    string direction = 1;  
}
```

Man sieht nun, dass eine 'Direction' Mitteilung definiert wird. Diese hat ein String Attribut, welches direction heisst. Mit dieser Definition wird mithilfe eines 'protobuf'

compiler' Code generiert. Dieser kann die definierte Message Serialisieren und Deserialisieren.

Beispiel Generierung für Python:

```
protoc -I=pb --python_out=pb pb/direction.proto
```

Mithilfe von diesem Beispiel wird die Protobuf Definition (direction.proto) zu Python Code generiert.

Beispiel Generierter Python Code:

```
# Generated by the protocol buffer compiler. DO NOT EDIT!
# source: direction.proto
```

```
import sys
_b=sys.version_info[0]<3 and (lambda x:x) or (lambda x:x.encode('latin1'))
from google.protobuf import descriptor as _descriptor
from google.protobuf import message as _message
from google.protobuf import reflection as _reflection
from google.protobuf import symbol_database as _symbol_database
# @@protoc_insertion_point(imports)

_sym_db = _symbol_database.Default()

DESCRIPTOR = _descriptor.FileDescriptor(
    name='direction.proto',
    package='',
    syntax='proto3',
    serialized_options=None,
    serialized_pb=_b('\n\x0f\x64irection.proto\x1e\n\tDirection\x12\x11\n\tdirection')
)

_DIRECTION = _descriptor.Descriptor(
    name='Direction',
    full_name='Direction',
    filename=None,
    file=DESCRIPTOR,
    containing_type=None,
    fields=[
        _descriptor.FieldDescriptor(
            name='direction', full_name='Direction.direction', index=0,
            number=1, type=9, cpp_type=9, label=1,
            has_default_value=False, default_value=_b("").decode('utf-8'),
            message_type=None, enum_type=None, containing_type=None,
            is_extension=False, extension_scope=None,
            serialized_options=None, file=DESCRIPTOR),
    ],
    extensions=[],
    nested_types=[],
    enum_types=[]
```

```

] ,
serialized_options=None,
is_extendable=False,
syntax='proto3',
extension_ranges=[],
oneofs=[
],
serialized_start=19,
serialized_end=49,
)

DESCRIPTOR.message_types_by_name[ 'Direction' ] = _DIRECTION
_sym_db.RegisterFileDescriptor(DESCRIPTOR)

Direction = _reflection.GeneratedProtocolMessageType( 'Direction' , ( _message.Message , )
DESCRIPTOR = _DIRECTION,
__module__ = 'direction_pb2',
# @@protoc_insertion_point(class_scope:Direction)
))
_sym_db.RegisterMessage(Direction)

# @@protoc_insertion_point(module_scope)

```

Das generierte Python File, welches mithilfe des 'protobuf compiler' erzeugt wurde.

3.12.3 Proof-of-Concept

Um zu testen, ob eine Kommunikation zwischen zwei Prozessen mithilfe der Middleware möglich ist, wurde eine kleine Testsoftware entwickelt. Dabei wurde das 'direction.proto' File verwendet, um von einem Prozess eine Richtung (Direction) zu einem anderen Prozess zu senden und zu empfangen.

3.12.4 Performance

Die Performance wurde nicht gemessen und verifiziert. Jedoch wurden bei den Tests keine Limitationen gefunden. Deshalb kann davon ausgegangen werden, dass die Performance für unseren Anwendungsfall ausreicht. Außerdem werden nur wichtige Events zwischen den Prozessen ausgetauscht und somit ist die Datenrate eher zweitrangig. Die Latenz ist jedoch sehr zentral um einen möglichst agilen Regelkreis zu implementieren. Hier bewegt sich ZeroMQ zwischen 450us - 700us.

Performance Tests:

- <http://zeromq.org/area:results>
- <http://nikolaveber.blogspot.com/2011/04/if-you-are-planning-large-or-not-even.html>

3.13 Grenzgeschwindigkeit

Die Grenzgeschwindigkeit ist nicht nur abhängig von der Masse des Zuges, des Schwerpunktes, der Leistung des Antriebs und den Kurven Radien. Es ist zentral, dass die Bilderkennung für die Signalerkennung und ebenfalls für die Gleiserkennung noch funktioniert. Deshalb werden in diesem Kapitel die verschiedenen Grenzgeschwindigkeiten aufgelistet, die näher betrachtet wurden. Daraus kann der Flaschenhals bezüglich Grenzgeschwindigkeit herausgelesen und dementsprechend auch die Risiken bestimmt werden.

3.13.1 Auflistung

Nachfolgend werden die einzelnen Grenzgeschwindigkeiten, die berechnet wurden, kurz beschrieben. Anschliessend wird in einem Fazit erläutert wo der Flaschenhals bezüglich der Geschwindigkeit liegt.

Grenzgeschwindigkeit ohne Bildverarbeitung

Die Grenzgeschwindigkeit ohne Bildverarbeitung basiert auf der Masse des Zuges, des Schwerpunktes, der Leistung des Motors und den Kurven Radien. Alle Berechnungen zu diesem Thema sind im Kapitel 3.2.1.

Gemäss den Berechnungen handelt es sich um 1.28m/s beim engsten Kurvenradius von 0.8m . Geradeaus kann wesentlich mehr erreicht werden. Wenn wir eine Gerade von 3m annehmen was wohl der maximal lange einer Geraden in PREN2 entspricht annehmen erreichen wir maximal 3m/s.

Grenzgeschwindigkeit mit Signalerkennung

Basierend auf der Grenzgeschwindigkeit ohne Bildverarbeitung, wird zusätzlich berücksichtigt, wie lange eine Aufnahme eines einzelnen Bildes dauert und wieviel Bilder gemacht werden müssen um die Umgebung genügend Abzudecken. Somit kann gewährleistet werden, dass ein verwendbares Bild aufgenommen werden kann während den Hochgeschwindigkeits-Runden. Die eigentliche Bildanalyse wird dabei vernachlässigt, weil dies nicht während den Hochgeschwindigkeits-Runden passiert, sondern anschliessend bei der Fahrt zum Haltesignal.

Maximal Geschwindigkeit: 3m/s Umgebung Abdeckung: 10cm (des Weges)

Pi Bilddaufnahme: 20ms Bilder Pro Meter: $3m/s * 0.02s = 0.06m = 6cm$

Die normale Bilddaufnahme ist also genügend schnell, deckt alle Bereiche ab da alle 6cm ein Bild gemacht werden kann bei der maximalen Geschwindigkeit und ist somit kein Hindernis für eine rasche Fahrt.

Grenzgeschwindigkeit mit Signalerkennung & Gleiserkennung

Zusätzlich zu den anderen beiden Grenzgeschwindigkeiten wird nun noch die Gleiserkennung berücksichtigt welche gemessen etwa 100ms pro Bild benötigt. Die Gleiserkennung dient dazu, Kurven zu erkennen und bietet somit einen weiteren Parameter für die Regelung um eine maximale Geschwindigkeit zu fahren. Die Kamera kann das Gleis mindestens 20cm im voraus genügend gut erkennen. Dies ist wie vorhin gezeigt wichtig bei der Berechnung.

Maximal Geschwindigkeit: 3m/s Umgebung Abdeckung: 20cm (des Weges)

Pi Bilddaufnahme: 100ms Bilder Pro Meter: $3m/s * 0.1s = 0.1m = 10cm$

Die Gleiserkennung ist somit wie die Signalerkennung genügend schnell, um auch mit der maximalen Geschwindigkeit von 3m/s zu fahren.

3.13.2 Fazit

Die verschiedenen Grenzgeschwindigkeiten wurden analysiert und das Ergebnis ist klar. Der vermutliche Flaschenhals liegt bei der vorgegebenen Leistung von 60 Watt und der daraus resultierenden Kombination von Motor & Mechanik. Dennoch ist es bemerkenswert was für Geschwindigkeiten möglich sind. Ob wir diese auch erreichen wird sich in PREN2 zeigen, wir sind aber sicher eine gute Konstruktion, ein gutes elektrisches Design und passende Softwaremittel gewählt zu haben.

4 Projektmanagement

Dieses Kapitel beschreibt das Projektmanagement des Team 28. Es wird auf die Organisation im Team, die Aufteilung der Arbeitspakete und Zeitplanung eingegangen.

4.1 Organigramm

Die Abbildung 23 zeigt die Organisation im Team auf. Das Team ist in die einzelnen Disziplinen Maschinentechnik, Elektrotechnik und Informatik aufgeteilt. Zusätzlich gibt es einen Projektleiter, dieser ist verantwortlich für die Kommunikation mit den Fachdozenten und ist bei allfälligen Abwesenheiten von Teammitgliedern informiert.

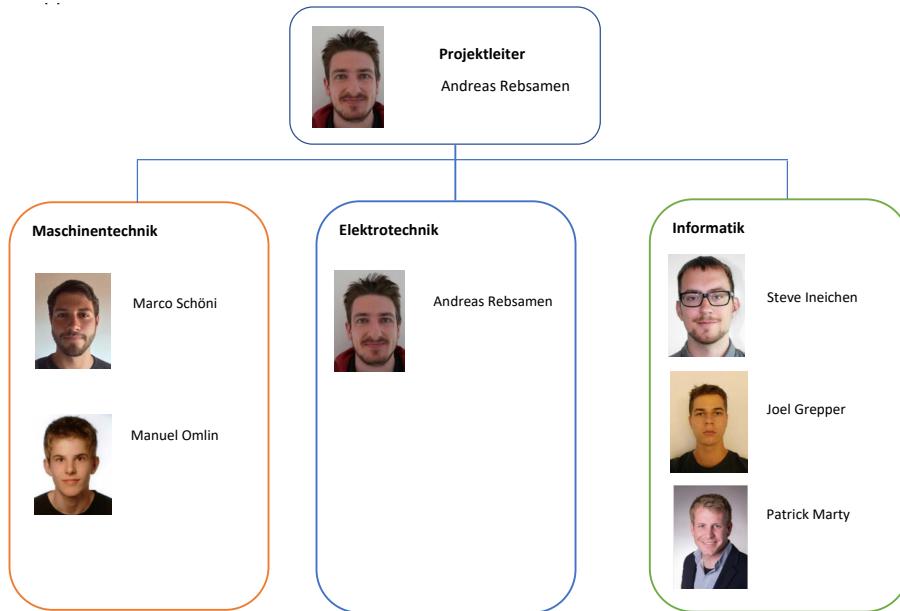


Abbildung 23: Organigramm Team 28

4.2 Zeitplanung

Mit den groben Arbeitspaketen wurde eine zeitorientierte Projektstruktur (Abbildung 24) erstellt. In dieser ist festgehalten, welcher Person oder Disziplin ein Arbeitspaket zugeteilt ist, ob diese neu, in Arbeit oder abgeschlossen ist.

Projektplan PREN1 Team 28	PF	Vor	Zur	Status	Wochen												Lernphase
					SW01	SW02	SW03	SW04	SW05	SW06	SW07	SW08	SW09	SW10	SW11	SW12	
Aufgabe	Verantwortlich	Deadline	Start	Ende	Auftr.	Fr.	Mo.	Ti.	Do.	Fr.	Sa.	Su.	Mo.	Di.	Mi.		
Meilensteine																	
0.1	Testat 1 Abgabe Anforderungsliste	12.10.2018			in work												
0.2	Testat 2 Auswahl der optimalen Lösungskombination(en)	09.11.2018			done												
0.3	Testat 3 Gesamtkonzept & Dokumentationen zu 80% erledigt	14.12.2018															
1 Einstieg und Planungsphase																	
1.1	Projektplan erstellen	alle	04.10.2018		done												
1.2	Projektbeschreibung erstellen (LaTeX)	Joel Gresser (IT)	20.09.2018	21.09.2018	done	Y											
1.3	Cloud Ordner erstellen	Andreas (IT)	20.09.2018	04.10.2018	done												
1.4	Git einrichten	Stefan (IT)	04.10.2018	04.10.2018	done	Y											
1.5	Anforderungsliste	alle	11.10.2018	20.09.2018	done	Y											
2 Recherche und Ideenfindung																	
2.1	Technologierecherche	alle	20.09.2018		done	Y											
2.2	Technologien für Akustik und Fahrzeugelektronik aufzeichnen	Joel Gresser (IT)	05.10.2018		done	Y											
2.3	Vacantebildung (Morphologische Kosten)	alle			done	Y											
2.4	Entscheid der Variante (Nutzwertanalyse / Fairweiser von alle)	alle			done	Y											
2.5	Recherche und Ideenfindung Dokumentieren	alle			done	Y											
3 Konzept Ausarbeitung																	
3.1	Mechanik	AR			in work												
3.2	Software	J			in work												
3.3	Technik	MS			in work												
3.4	Grenzgeschwindigkeit	alle			done	Y											
3.5	Konzepte zusammenführen	alle			done	Y											
4 Dokumentation fertigstellen																	
4.1	Komplette Doku schreiben	alle			in work	Y											
4.2	Dokumentation überarbeiten	alle			done	Y	Y										
4.3	Dokumentation drucken	alle			done												
Sonstige Termine																	
Joel Gresser im WK (evtl. Teilweise abwesend)	Joel Gresser (IT)																

Abbildung 24: Zeitorientierte Projektstruktur Team 28

Stand: SW 12

Die detailliertere Aufteilung dieser Arbeitspakte wird dann in Trello (Abbildung 25) vorgenommen. In der Projektstruktur wird lediglich eingetragen, ob ein bestimmtes Arbeitspaket darin in Trello erfasst ist. Dort kann es in mehrere Teilkästen aufgeteilt und bestimmten Personen zugewiesen werden. Die Arbeitspakte in Trello werden je nach Status in die Kategorien "To do", "doing", oder "done" eingeteilt. Separat werden auch noch die Meilensteine festgehalten und als "done" gekennzeichnet sobald diese abgeschlossen sind.

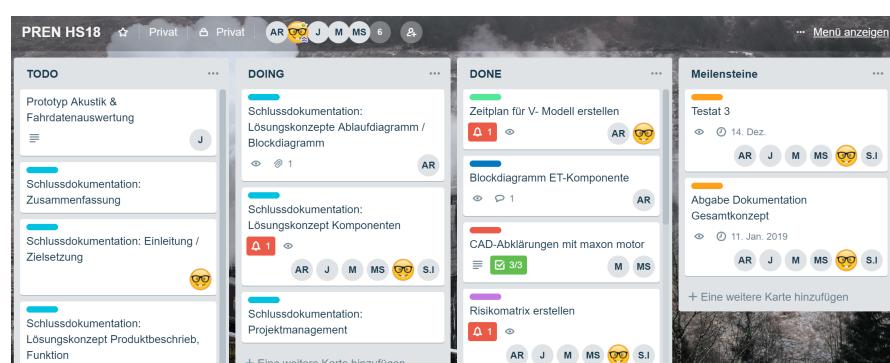


Abbildung 25: Bildschirmausschnitt Trello Team 28

Stand: SW 12

4.3 Dokumentation

Die Dokumentation wird in L^AT_EXgeschrieben. Der Quell-Text wird in eine normale Text-Datei mit der Endung ".tex" geschrieben. Ein Programm übersetzt diese Quell-Datei dann in ein Dokument wie z.B. ein PDF oder PostScript. McPeak

Für die Dokumentation dieses Projekts wird aus dem L^AT_EXCode ein PDF erstellt. Der Austausch der Daten im Team erfolgt gemäss Kaptitel 4.4.

4.4 Datenaustausch

Um Daten im Team auszutauschen, wird auf zwei verschiedene Cloud Plattformen gesetzt. Alle Teammitglieder haben vollständigen Zugriff auf beide Plattformen.

One Drive

Für diverse Ablagen, Word- und Exceldokumente oder ähnliches steht ein Ordner in Microsoft One Drive zur Verfügung. Diese Daten werden jederzeit mit allen Teammitgliedern synchronisiert. In Dokumenten von Microsoft Office ist es auch möglich, dass mehrere Personen zur selben Zeit am selben Dokument arbeiten.

Github

Es steht ein Github Repository zur Verfügung. Mit dem Programm Git kann man auf die Daten zugreifen und seine Änderungen hochladen. Dabei können alle Änderungen von allen Personen verfolgt werden. Die genaue Verwendung und die benutzten Tools sind für die Teammitglieder direkt im Repository beschrieben.

Dieses Repository wird für die Dokumentation und für den Source-Code benutzt. Es gibt einen Ordner für die Dokumentation (alle L^AT_EXDateien, Bilder, Zeichnungen, usw.) und einen weiteren für den Quell-Code. Dort ist der Programmcode für das Raspberry Pi und das Tiny K22 abgelegt.

Das Repository kann online unter <https://github.com/Inux/pren> eingesehen werden.

5 Schlussdiskussion

5.1 Kosten

Gemäss der Aufgabenstellung steht ein Budget von 500 Franken zur Verfügung. In Tabelle 18 wurden alle Komponenten in einer Kostenübersicht zusammengetragen. Gemäss dieser Übersicht gibt es noch eine Reserve von ca. 50 Franken für nicht vorhergesehene Änderungen.

In dieser Übersicht nicht enthalten sind Komponenten, welche aus der Mechanikwerkstatt oder dem Elektronik Labor der HSLU bezogen werden können. Auch sind Stundenansätze für Werkstattpersonal oder 3D-Drucker nicht berücksichtigt.

Beschreibung	Lieferant	Anzahl	Stückpreis	Gesamtpreis
Raspberry Pi 3 Model B+	pi-shop	1	CHF 39.00	CHF 39.00
Raspberry Pi Zero W	pi-shop	1	CHF 10.80	CHF 10.80
Raspberry Pi Kamera Beleuchtung	pi-shop	2	CHF 24.90	CHF 49.80
Raspberry Pi Kamera	pi-shop	2	CHF 32.90	CHF 65.80
Passiver Buzzer	play-zone	1	CHF 5.00	CHF 5.00
StromPi 3	Reichelt	1	CHF 41.64	CHF 41.64
StromPi 3 Battery Pack, 1000 mAh	Reichelt	1	CHF 31.20	CHF 31.20
Beschleunigungssensor ADXL345	aliexpress	1	CHF 1.00	CHF 1.00
Tiny K22	hslu	1	CHF 26.00	CHF 26.00
Arduino IBT_2 (DC-Motoren Treiber) (43A)	wish	1	CHF 9.00	CHF 9.00
DCDC CC Converter (12A)	wish	1	CHF 3.00	CHF 3.00
Ultraschallsensor HS-SR04	aliexpress	1	CHF 0.77	CHF 0.77
TOF Sensor VL53L0X	aliexpress	1	CHF 3.69	CHF 3.69
LM2596 DC-DC Schritt-down Converter (2.5A)	aliexpress	1	CHF 0.90	CHF 0.90
Strommesswiderstand 13FR200E	Distrelec	1	CHF 2.80	CHF 2.80
LogicLevelConverter 5V zu 3,3V	aliexpress	1	CHF 0.90	CHF 0.90
Tower Pro Micro Servo SG90	wish	1	CHF 1.60	CHF 1.60
Mini Modul PWM Speed Control Über L298N	aliexpress	1	CHF 0.69	CHF 0.69
Antriebsmotor mit Encoder	Sponsor	1	CHF 30.00	CHF 30.00
Schwenker Motor mit Encoder	Sponsor	1	CHF 25.00	CHF 25.00
Zahnriehmen	Mädler	1	CHF 15.00	CHF 15.00
Zahnriehmenrad	Mädler	3	CHF 8.00	CHF 24.00
Kupplungen	Mädler	1	CHF 30.00	CHF 30.00
Zahnräder	Mädler	3	CHF 10.00	CHF 30.00
Total		30		CHF 447.59

Tabelle 18: Kostenübersicht Gesamtkonzept

5.2 Risikomanagement

Während der Konzeptionsphase, in welcher dieses Konzept entstanden ist, wurden in bestimmten zeitlichen Intervallen Risikoanalysen durchgeführt. Diese flossen in die Entscheidungsfindung für alle Lösungskonzepte mit ein. Zum Schluss der Konzeptphase, anhand des fertigen Konzeptes, wurde nochmals eine Risikoanalyse durchgeführt pro Disziplin. Diese Risikoanalyse wird für das PREN2 entscheidend sein, um die geplanten Lösungskonzepte erfolgreich realisieren zu können.

Nr	Disziplin	Risiko	Massnahmen PREN1	Massnahmen PREN2
1	Elektrotechnik	Spannungsabfall/ Wackelkontakt bei Schleifkontakte- ten	Stützkondensator/ Akku	weitere Schleifkontakte
2	Elektrotechnik	MC Überlastung durch Antriebsen- coder	aktuelle Messung zei- gen 10% Auslastung	Encoder tauschen
3	Elektrotechnik	Strom-Pi mit Ak- ku zu teuer	-	alternative suchen
4	Elektrotechnik	zu hoher Anlauf- strom beim Motor	-	Momentenregelung mittels MC
5	Elektrotechnik	Ultraschallsensor zu ungenau	-	Systematische Fehler mittels Software korri- gieren / TOF Sensor verwenden
6	Maschinenbau	zu wenig Reibung Räder => Schie- nen	Aufgummierung ge- plant	Oberfläche der Räder optimieren (Rauheit)
7	Maschinenbau	Kupplungen schleifen	Grössere Haftreibung durch erhöhte Rauheit	alternative Befestigungs- methode wählen (Gewindestifte)
8	Maschinenbau	Zug Gewichtsver- teilung nicht opti- mal	statische Berechnun- gen durchgeführt	Gegengewichte ein- setzen
9	Maschinenbau	Kippen in der Kurve oder bei Würfelaufnahme	Konzept mit Kraftaus- gleich ausgearbeitet	Alternatives Konzept verfolgen
10	Maschinenbau	Riemen defekt	Ersatzriemen auf La- ger	Riemenart wechseln
11	Maschinenbau	Reibfläche zwi- schen Ladungs- träger und Wägen	Material mit günstigen Gleiteigenschaften ver- wendet	Axiallager einbauen
12	Maschinenbau	Würfel Aufnah- me / Platzierung	Optimierung des Kon- zeptes «Greifers»	alternatives Konzept verfolgen
13	Maschinenbau	Reibung zu hoch bei Kran	Material mit günstigen Gleiteigenschaften ver- wendet	Lagerung einbauen
14	Informatik	Tafelerkennung zu langsam	Optimierung der Algo- rithmen	wechsel auf Raspberry PI 3 A+

15	Informatik	Nummererkennung zu langsam	Optimierung der Algorithmen	wechsel der ML-Framework
16	Informatik	Kommunikation MC <=> RPI zu langsam	max. Baudrate verwendet	wechsel zu anderem Bus
17	Informatik	Bildverarbeitung zu langsam	C++ als Programmiersprache verwenden	OpenCL verwenden

Tabelle 19: Risikotabelle

5.3 Nächste Schritte

Nach der Konzeptionsphase im PREN1 folgt nun die Realisierung des Zuges im PREN2. Das Ziel am Anfang des PREN2 wird sein, so schnell wie möglich die mechanische Grundplattform des Zuges zu erstellen, damit die Elektronik und die Software getestet werden können. Währenddessen arbeiten Elektrotechnik und Informatik an der Finalisierung ihrer Komponenten. Mitte des PREN2 ist geplant, den fertigen Zug in physischem Zustand fertig zu haben, damit in der zweiten Hälfte das Testen und Optimieren beginnen kann.

Elektrotechnik

1. ersten Prototyp erstellen (damit Informatiker testen können)
 - Aufbau auf erstem Zug / Prototyp vom Maschinenbau
 - PCB für Stromversorgung und Antrieb
 - Mikrocontroller mit Software für Antrieb und Kommunikation mit Pi
2. Ansteuerung / Auslesen Sensoren
3. Ansteuerung Schwenker
4. Anpassungen (speziell für präzisen Halt)
5. Kompletter Prototyp
 - PCB für alle Komponenten
 - Komplette Software für Mikrocontroller
6. Finales Zusammenführen mit Informatik und Maschinenbau

Informatik

1. Finalisieren bestehender Komponenten
 - Signalerkennung, Gleiserkennung, Akustik, Sensorik
2. Die restlichen Komponenten implementieren
 - Interface zu Motor/ Kran
3. Hauptablauf implementieren
4. Testing & Bugfixing

Maschinenbau

1. Konstruktionszeichnungen erstellen
 - Funktionsmasse
 - Materialwahl
2. Fertigen eines voll funktionsfähigen Protoyps, damit die Elektronik angebracht werden kann und die Informatik ihre Versuche durchführen kann.
 - Einkaufsteile bestellen
 - Fertigung durch Werkstattpersonal oder Eigenfertigung
3. Weitere Versuche an Prototyp durchführen sowie Optimierungen und Änderungen vornehmen

5.4 Lessons learned

Lessons learned Elektronik Es hat sich gezeigt, dass eine enge Zusammenarbeit unter den Disziplinen entscheidend ist. Ein sehr gutes Beispiel dafür ist die Auswahl des Motors. Weder die Fachleute Elektrotechnik, noch die Fachleute Maschinentechnik können diese Auswahl alleine treffen. Bevor eine genauere Auswahl vorgenommen werden kann müssen zuerst von beiden Disziplinen alle Anforderungen und Limitierungen zusammengetragen und diskutiert werden. Im Team kann diese Auswahl dann sehr gut getroffen werden.

Lessons learned Informatik Für die Informatik ist es schwierig etwas Greifbares zu erschaffen ohne die Mithilfe der anderen Disziplinen. Wir versuchten deshalb möglichst viel zu testen und haben auch begonnen die Architektur der finalen Lösung zu implementieren. Unsere grösste Lektion ist die Realisation des Wertes einer guten Softwarearchitektur. In unserem Fall konnten wir dank der Middleware Architektur schon finale Komponenten entwickeln und diese testen obwohl der Zug noch nicht vollständig gebaut ist. Dies gilt speziell für die Signalerkennung, Gleiserkennung, Akustik und die Sensorik.

Lessons learned Maschinenbau Ein grosser Vorteil im Bereich der Maschinentechnik sind die Erfahrungen, welche die Teammitglieder der Mechanik durch ihre Ausbildung als Konstrukteure einbringen können. Dadurch sind rasch gute Lösungskonzepte entwickelt und ausgearbeitet worden. Teilkonzepte konnten somit bereits mit Prototypen getestet werden. Es wurde die Erfahrung gemacht, dass für die Maschinentechnik die Zusammenarbeit mit den anderen Disziplinen wesentlich ist, da sie das Grundgerüst des Zuges bildet und die Verbindung zwischen Elektronik und Informatik ist.

6 Verzeichnisse

Abbildungsverzeichnis

1	Ablaufdiagramm	8
2	Baugruppe Lokomotive	9
3	Schnittansicht der Lokomotive	10
4	Explosionsdarstellung Grundwagens	11
5	Baugruppe Antriebwagen	12
6	Explosionsdarstellung Antriebseinheit	13
7	Schwerpunkt der Lokomotive (Symbolisch)	15
8	Explosionsdarstellung Ladungsträger	16
9	Baugruppe	18
10	Draufsicht	18
11	Links: Kurvenscheibe, Rechts: Zahnrad	19
12	Testaufbau	20
13	DC Motoren	21
14	Verkabelung Buzzer	23
15	Verkabelung Beschleunigungssensor (http://fritzing.org)	25
16	Komponentendiagramm Elektronik	26
17	Brückengleichrichter (www.elektronik-kompendium.de)	27
18	Signalverlauf Encoder (www.maxonmotor.ch)	30
19	Architektur Hardware Signalerkennung	34
20	Ablauf Tafelerkennung	35
21	Nummerposition- und Nummererkennung	36
22	Software Architektur Middleware. Gezeichnet mit https://draw.io	42
23	Organigramm Team 28	47
24	Zeitorientierte Projektstruktur Team 28 Stand: SW 12	48
25	Bildschirmausschnitt Trello Team 28 Stand: SW 12	48

Tabellenverzeichnis

1	Positionsnummern der Lokomotive	9
2	Positionsnummern des Grundwagens	11
3	Positionsnummern des Antriebwagens	12
4	Positionen Antriebseinheit	13
5	Reibungskoeffizienten von Materialpaarungen (durch Versuche ermittelt) .	14
6	Größen für die Beschleunigungsberechnung	14
7	Größen für die Geschwindigkeitsberechnung	15
8	Positionen des Ladungsträgers	17
9	technische Daten (www.playzone.ch)	22
10	Technische Daten (https://www.adafruit.com/product/1231)	24
11	benötigte Leistung der Komponenten	28
12	Übersicht Proof-of-Concept Elektronik	33
13	Beschreibung der OpenCV Funktionen	35
14	Auflistung aller Testklassen mit jeweiligen Testarten	36
15	Kommunikation Frames: Pi ⇒ Tiny	40
16	Kommunikation Frames: Tiny ⇒ Pi	41

17	Informationsinhalt des Statusbytes	41
18	Kostenübersicht Gesamtkonzept	50
19	Risikotabelle	52

Literatur

- Pi-Shop. Raspberry pi - raspberry pi 3 model b+, 2018. URL <https://www.pi-shop.ch/raspberry-pi-3-model-b>.
- Alex Eames. How much power does pi zero w use?, 2018. URL <https://raspi.tv/2017/how-much-power-does-pi-zero-w-use>.
- NXP. *Kinetis® K22: 120MHz Cortex-M4F 512KB Flash (64-121 pin) Data Sheet*, 2016.
- Maxon. *DCX 32 L Katalogseite 86*, 2018a.
- Maxon. *DCX 19 S Katalogseite 78*, 2018b.
- Mi-PC. *SERVO MOTOR SG90 DATA SHEET*, 2016.
- Scott McPeak. What the heck is latex? <http://scottmcpeak.com/latex/whatislatex.html>. (Accessed on 12/14/2018).
- Pi-Shop. Raspberry pi - raspberry pi zero w. <https://www.pi-shop.ch/raspberry-pi-zero-w>, a. (Accessed on 12/22/2018).
- Pi-Shop. Raspberry pi - pi supply bright pi - bright white und ir kamera licht für raspberry pi | pi-shop.ch. <https://www.pi-shop.ch/pi-supply-bright-pi-bright-white-und-ir-kamera-licht-fuer-raspberry-pi>, b. (Accessed on 12/22/2018).
- Pi-Shop. Raspberry pi - original raspberry pi kamera module v2 | pi-shop.ch. <https://www.pi-shop.ch/raspberry-pi-kamera-module-v2>, c. (Accessed on 12/22/2018).
- Play-Zone. Play-zone.ch passiver buzzer / speaker, 3.3v. <https://www.play-zone.ch/de/passiver-buzzer-speaker-3-3v.html>. (Accessed on 12/22/2018).
- Reichelt. Rasp strom pi 3: Raspberry pi - der strompi 3 bei reichelt elektronik. https://www.reichelt.de/raspberry-pi-der-strompi-3-rasp-strom-pi-3-p232866.html?&trstct=pos_0, a. (Accessed on 12/22/2018).
- Reichelt. Rpi strompi akku: Raspberry pi - strompi 3 battery pack, 1000 mah bei reichelt elektronik. https://www.reichelt.de/raspberry-pi-strompi-3-battery-pack-1000-mah-rpi-strompi-akku-p232867.html?&trstct=pos_2, b. (Accessed on 12/22/2018).
- AliExpress. I43 1 stücke gy 291 adxl345 digital dreiachsen beschleunigung der schwerkraft tilt modul iic/spi übertragung auf lager in i43 1 stücke gy-291 adxl345 digital dreiachsen beschleunigung der schwerkraft tilt-modul iic/spi übertragung auf lager aus integrierte schaltungen auf aliexpress.com | alibaba group. https://de.aliexpress.com/item/Free-Shipping-GY-291-ADXL345-digital-three-axis-acceleration-of-gravity-tilt-module-IIC-32346306872.html?spm=a2g0x.search0104.3.8.3a781d56R05pDY&ws_ab_test=searchweb0_0,searchweb201602_5_10065_10068_319_10059_10884_317_10887_10696_100031_321_322_10084_453_10083_454_10103_10618_10307_538_537_536_10134,searchweb201603_51,ppcSwitch_0&

algo_expid=a352fecc-60ea-4079-b651-cade58a1b700-1&algo_pvid=a352fecc-60ea-4079-b651-cade58a1b700, a. (Accessed on 12/22/2018).

wish. Wish | halbleiter-motor-treiber auto bts7960 43a h-bruecke pwm-antrieb fuer arduino hot. <https://www.wish.com/product/5b97621dab410c29feae20c2>, a. (Accessed on 12/22/2018).

wish. Wish | dc-dc cc cv buck converter step-down power module 7-32v to 0.8-28v 12a 300w. <https://www.wish.com/product/5976f5aadc35dc104728e52d>, b. (Accessed on 12/22/2018).

AliExpress. 1 stücke shengyang hc sr04 zu welt ultraschall welle detektor bis hin modul für arduino abstand sensor in 1 stücke shengyang hc-sr04 zu welt ultraschall welle detektor bis hin modul für arduino abstand sensor aus sensoren auf aliexpress.com | alibaba group. <https://de.aliexpress.com/item/1pcs-HC-SR04-to-world-Ultrasonic-Wave-Detector-Ranging-Module-for-arduino-Distance-Sensor/32786781050.html?spm=a2g0s.9042311.0.0.25dd4c4dbC08tr>, b. (Accessed on 12/22/2018).

AliExpress. Gy 530 vl53l0x welt kleinste zeit o f flug (tof) laser ranging sensor in gy-530 vl53l0x welt kleinste zeit-o f-flug (tof) laser ranging sensor aus sensoren auf aliexpress.com | alibaba group. <https://de.aliexpress.com/item/GY-530-VL53L0X-World-smallest-Time-o-f-Flight-ToF-laser-ranging-sensor/32773126352.html?spm=a2g0s.9042311.0.0.25dd4c4dbC08tr>, c. (Accessed on 12/22/2018).

AliExpress. 2 stücke lm2596 dc dc schritt down converter power supply module lm2596s einstellbare buck step down power module spannung regler in 2 stücke lm2596 dc-dc schritt-down converter power supply module lm2596s einstellbare buck step-down power module spannung regler aus wechselrichter & konverter auf aliexpress.com | alibaba group. <https://de.aliexpress.com/item/2-pcs-LM2596-Step-Down-Power-Module-LM2596S-DC-DC-3A-Adjustable-Buck-Step-Down-Power/32802169864.html?spm=a2g0s.9042311.0.0.25dd4c4dbC08tr>, d. (Accessed on 12/22/2018).

Distrelec. 13fr200e | kaufen strommesswiderstand | ohmite | distrelec. <https://www.distrelec.ch/de/strommesswiderstand-ohmite-13fr200e/p/30046857>. (Accessed on 12/22/2018).

5 stücke iic i2c uart spi logic level converter bi directional modul 5 v zu 3,3 v in 5 stücke iic i2c uart spi logic-level-converter bi-directional modul 5 v zu 3,3 v aus integrierte schaltungen auf aliexpress.com | alibaba group. <https://de.aliexpress.com/item/5PCS-IIC-I2C-UART-SPI-Logic-Level-Converter-Bi-Directional-Module-5V-to-3-3V-For/32826629975.html?spm=a2g0s.9042311.0.0.25dd4c4dbC08tr>. (Accessed on 12/22/2018).

wish. Tower pro micro servo sg90 | wish. <https://www.wish.com/search/Tower%20Pro%20Micro%20Servo%20SG90/product/57c8e0946f5941265132b849>, c. (Accessed on 12/22/2018).

Mädler. Zahnriemen profil at 5, breite 10 mm - mädler webshop. <https://www.maedler.ch/product/1643/1616/963/zahnriemen-profil-at-5-breite-10-mm>, a. (Accessed on 12/22/2018).

Mädler. Zahnriemenräder at5 für riemenbreite 10 mm - mädler webshop. <https://www.maedler.ch/product/1643/1616/996/zahnriemenraeder-at5-fuer-riemenbreite-10-mm>, b. (Accessed on 12/22/2018).

Mädler. Spannsätze sig, rostfrei, bohrung 4 bis 40mm - mädler webshop. <https://www.maedler.ch/product/1643/1621/spannlsaetze-sig-rostfrei-bohrung-4-bis-40mm>, c. (Accessed on 12/22/2018).

Mädler. Stirnzahnräder kunststoff pom schwarz, gefräst, modul 1 - mädler webshop. <https://www.maedler.ch/product/1643/1618/1034/2545/stirnzahnraeder-kunststoff-pom-schwarz-gefraest-modul-1>, d. (Accessed on 12/22/2018).