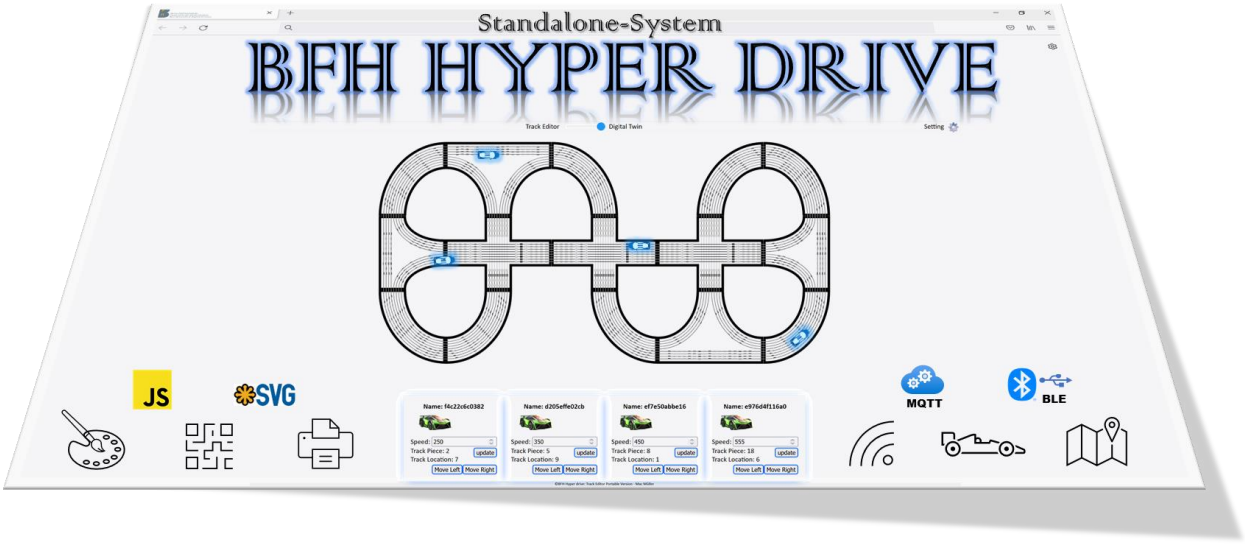


Standalone Track Editor and Observer for Anki Overdrive

BFH Hyperdrive

Bachelorthesis

Studiengang:	Bachelor of Science in Informatik
Autor:	Mac Müller
Betreuer:	Prof. Dr. Ing. Reto E. Koenig
Experte:	Thomas Jäggi
Datum:	15. Juni 2022

Berner Fachhochschule

Technik und Informatik

Management-Summary

«BFH Hyperdrive» ist ein Standalone-System für das ursprüngliche Auto-Rennspiel «Anki Overdrive». Es eignet sich hervorragend als Basissystem für Anwendungsbereiche in der Forschung wie zum Beispiel Verkehrssimulationen oder in der Steuerungs- und Prozesskontrolle. Bei dieser Thesis wurde festgestellt, dass die skalierbare Vektorgrafik (SVG) mehr Vorteile als andere Grafik-Formate bringt und sich in Zukunft bei der Digitalisierung wahrscheinlich durchsetzen wird.

Ausganglage
Das bestehende System für Anki Overdrive ermöglicht es mit dem Frontend «Track Editor» die Strecken zu kreieren und auf Papier auszudrucken. Die Fahrzeuge folgen autonom der gedruckten Spur und lesen gleichzeitig den Strichcode neben der Spur ab. Im Strichcode enthalten sind die Positionsangaben, welche die Fahrzeuge dem Frontend übermitteln. Anhand dieser Daten können digitale Zwillinge die echten Fahrzeuge und die ausgedruckten Streckenteile auf dem Display visualisieren.

Bei der Analyse des bestehenden Systems zeigten sich zwei erhebliche Probleme. Zum einen müssen verteilte Anwendungen auf verschiedenen Computer und Servern mit Internetverbindung ausgeführt werden, was die Applikation unnötig komplex macht. Zum anderen werden die Streckenbilder im PNG-Format erzeugt. Die PNG-Bilder sind in der Qualität teilweise ungenügend und benötigen bei der Erstellung zudem massiv Rechenleistung, was die gesamte Applikation unnötig verlangsamt.

Zielsetzung
Die Bachelorthesis hat daher zwei Hauptziele:

- Das Frontend wird weiterentwickelt, um die Streckenbilder im SVG-Format erzeugen zu können. Alle Funktionen werden komplett neu implementiert, damit diese direkt SVG verarbeiten können.
- Die Zusammenführung sämtlicher benötigten Anwendungen auf einem lokalen Standalone-System.

Umsetzung
Mit der Struktur von SVG wurde ausführlich experimentiert. Das Frontend wurde mit JavaScript ohne Hilfe eines Frameworks implementiert. Um die Benutzerfreundlichkeit zu erhöhen, wurde das Frontend als Single-Page-Webanwendung mit dem MVC-Pattern aufgebaut. Der bereits bestehende Node.JS Fahrzeugkontroller, welcher die Befehle an die Fahrzeuge via das BLE- Protokoll übergibt, wurde ohne Anpassungen in das neue System übernommen. Die Kommunikation zwischen dem Fahrzeugkontroller und dem Frontend wird mittels MQTT-Protokolls über den lokalen MQTT-Broker vermittelt. Der MQTT-Broker wurde für das Frontend so konfiguriert, dass MQTT über Websocket erreichbar ist. Die Kollaborativ-Funktion der Systeme wurde mit Hilfe einer MQTT-Bridge realisiert.

Ergebnisse
Die Implementierung von SVG war sehr zeitaufwendig, da die Komplexität von der SVG-Struktur hoch ist. Heute kann der Track Editor die Streckenbilder mit SVG Format erzeugen. Im Gegensatz zu PNG ist die Dateigrösse um bis zu 40mal kleiner geworden. Dies zeigt sich nun auch in der Programmlaufzeit, welche um den Faktor 3 schneller geworden ist. Erstaunlich ist auch die Qualität der ausgedruckten Strecken. Die Animation des Digitalzwillings ist fliessend und präziser geworden. Das System ist nach dem Start des Computers sofort bereit. Optional kann es im Netzwerk mit weiteren «BFH Hyperdrive» kollaborativ arbeiten.

Inhaltsverzeichnis

1 Einleitung

1.1 Anki Overdrive

Abbildung 1: Anki Overdrive

Anki Overdrive ist ein Rennspiel. Es besteht aus Fahrzeugen (Modell-Autos) und einer Mobile Applikation. Die Rennbahn (Track) kann der Spieler beliebig aus verschiedenen Teilen (Track-Piece) zusammenbauen.
Folgende Track-Piece sind online erwerbbar:

- Gerade
- Kurve
- Kreuzung
- Startstück

Als Alternative, geeignet für Softwareentwickler, können die Fahrzeuge über ein Software-Development Kit (SDK) [1] gesteuert werden. Diese übernimmt die Kontroller Funktion.

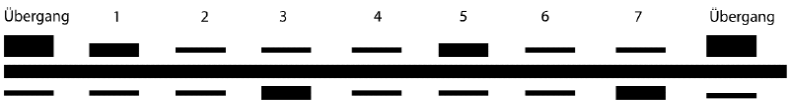


Abbildung 2: Track-Piece Code: 1/3 Länge von einer Spur [2]

Unter jedem Track-Piece befinden sich unsichtbare, durchgezogene 16 Linien (Spuren). Rechts und links neben der Linie existieren drei unterschiedliche Block-Typen, welche den 7 Bit Code wie auf der Abbildung 2 repräsentieren. Unter jedem Fahrzeug ist ein Infrarotsensor (IR-Sensor) eingebaut. Somit kann das Fahrzeug einer Linie folgen, den Code ablesen und diesen zum Kontroller übermitteln.

1.2 IST-System

Bei der letzten Bachelorthesis "Anki Overdrive Track Editor" von Herrn Dominique Marc Hofmann [2] wurde ein System für das Rennspiel "Anki Overdrive" entwickelt.

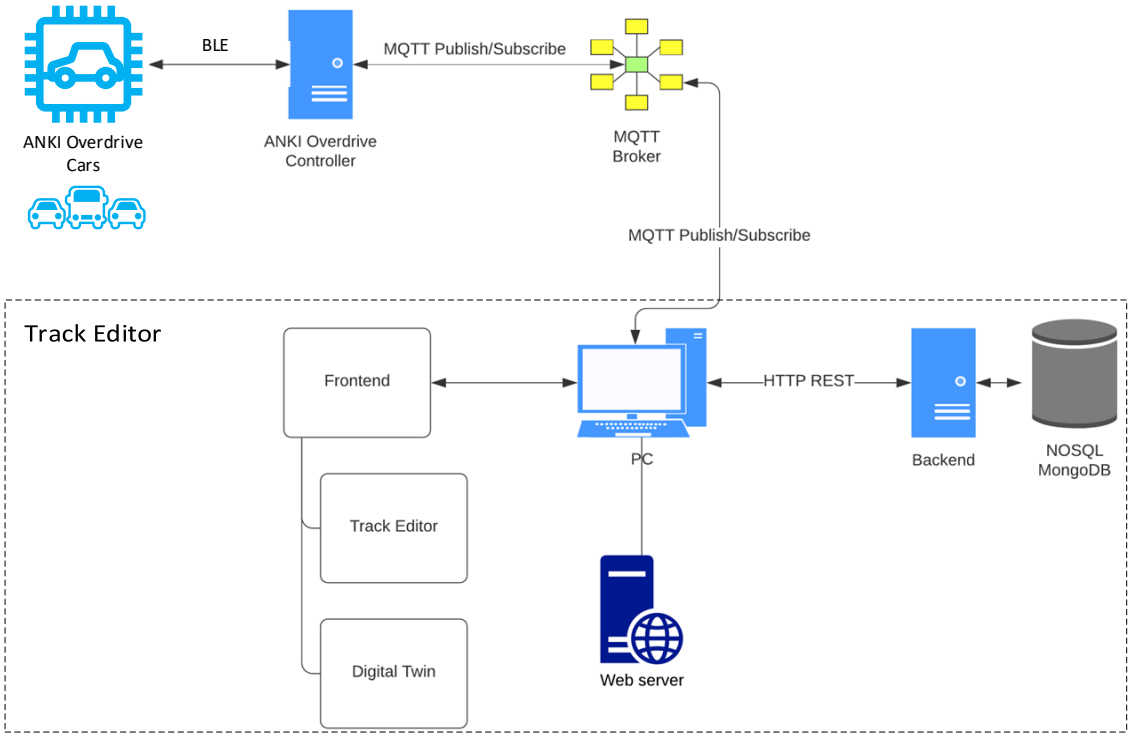


Abbildung 3: System von der letzten Bachelorthesis

Die Abbildung 3 zeigt die Softwarearchitektur des übernommenen Systems.

- Der Fahrzeugkontroller gibt die Befehle an die Fahrzeuge über das BLE-Protokoll.
- Der MQTT Broker ist ein Vermittler des MQTT-Protokolls zwischen dem Fahrzeugkontroller und der Applikation "Track Editor". Er befindet sich auf einem Server, welcher durch eine Internetverbindung erreichbar ist.
- Die Applikation "Track Editor" wurde mit AngularJS-Framework, sowie das Backend mit Node.js entwickelt. Ausserdem wurde zusätzlich ein «MongoDB» Datenbanksystem benötigt.

1.3 Ausgangslage

Bei der Analyse des bestehenden Systems zeigten sich zwei erhebliche Probleme:

1. Zum einen müssen verteilte Anwendungen auf verschiedenen Computer und Servern mit Internetverbindung ausgeführt werden, was die Applikation unnötig komplex macht.

2. Zum anderen werden die Streckenbilder im PNG-Format erzeugt. Die PNG-Bilder sind in der Qualität teilweise ungenügend und benötigen bei der Erstellung massiv Rechenleistung, was die gesamte Applikation verlangsamt.

Im Rahmen von Projekt 2 BTI3041HS2021/22 zeigte das Experiment auf, dass der Track Editor neu als Frontend-Basis entwickelt werden sollte. Das existierende System müsste integriert werden und sollte ohne mühsame Installation funktionieren. Die Nachteile des PNG-Formates, könnten durch den Einsatz des skalierbaren SVG Bild-Formates abgelöst werden (SVG ist XML basiert und heutzutage ein Standard für den Webbrowser).

2 Zielsetzung und Anforderung

2.1 Persona

Nr.	Persona
1	**Thomas Schweizer studiert Vollzeit Informatik an der BFH** Er kennt das bestehende Anki Overdrive System von einer Bachelor Veranstaltung und möchte gern das System ausprobieren und eventuell weiterentwickeln. Die Installation ist zu kompliziert, weiter treten viele Probleme bei der Installation auf. Er wünscht, dass die Installation einfacher wird. Er verfügt über keine Kenntnisse von AngularJS-Framework, würde das System jedoch gerne weiterentwickeln. Der Broker für das System liegt irgendwo im Internet. Die Einstellung des MQTT-Brokers wurde fest im Code definiert.
2	**Dr. Ivan Bisatz technischer Professor an der BFH** Für das neue System soll die Installation einfacher werden oder noch besser, man muss das System gar nicht installieren. Er wünscht, dass das System kollaborativ ist, sodass man von zwei Standorten aus das System bedienen kann. Optional zu entwickeln wäre zusätzlich noch eine Smartphone App.
3	**Bea Stauber studiert Vollzeit Informatik an der BFH** Sie hat das System bereits verwendet. Sie bemängelt das die Animation nicht wirklich fliesst, da das Auto von einem Streckenteil zum anderen Streckenteil gesprungen ist. Die gesamte Applikation soll schneller reagieren.

Tabelle 1: Persona

2.2 [HZ] Hauptziele

Nr.	Ziel
HZ1	**Standalone-System** Das Ziel ist erreicht, wenn das System auf einem Computer nach dem Start automatisch läuft, und den Webservice zur Verfügung stellt.
HZ2	**Verbesserung des Frontend** Das Ziel ist erreicht, wenn das Frontend den Track mit dem SVG-Format erzeugt. Der erzeugte Track muss von allen Funktionen der Applikation weiterverarbeitet werden können.

Tabelle 2: Hauptziele

2.3 [UZ] Unterziele

Nr.	Ziel
UZ1	**Statischer Webserver für den Track Editor** *"Das System stellt das Frontend zur Verfügung"* Das Ziel ist erreicht, wenn das Frontend in marktüblichen Webbrowsern bei dem/-r Benutzer/-in einwandfrei funktioniert.
UZ2	**Kollaborativ** *"Mehrbenutzer/-innen System"* Das Ziel ist erreicht, wenn mehr als ein/-e Benutzer/-in mit dem System sowohl im LAN als auch über die Internetverbindung (WAN) eine Verbindung zum Fahrzeugkontroller haben.
UZ3	**Verbesserung des Digitalzwillings Fahrzeugkontrollers** *"Der/Die Benutzer/-innen muss mehrere Einstellungen der Fahrzeuge anpassen können"* Das Ziel ist erreicht, wenn der/die Benutzer/-in die Geschwindigkeit, die Beschleunigung sowie die Fahrspur festlegt, woraufhin das Fahrzeug korrekt auf den Befehl antwortet.

Tabelle 3: Unterziele

2.4 Anforderungsarten

In diesem Abschnitt werden alle Anforderungen formuliert. Diese wurden dabei in funktionale und nichtfunktionale (Qualität) Anforderungen unterteilt.

Legende:
In den Tabellen werden folgende Eigenschaften der Anforderungen benutzt:

P: Geschätzte Priorität der Anforderung.
V: Variabilität der Anforderung.
K: Geschätzte Komplexität der Anforderung.
R: Geschätztes Risiko der Anforderung im Zusammenhang mit P, V, K Eigenschaften.
Quelle: Von wem kommt diese Anforderung:
Stakeholder (**SH**), Projektleiter (**PL**) oder durch eine andere Anforderung.
Ziele: Welches Hauptziel bzw. Unterziel wird mit dieser Anforderung erreicht.

Bewertungsgewicht:
↑ = hoch/muss → = mittel/soll ↓ = niedrig /tief/"nice to have"

2.4.1 [FA] Funktionale Anforderungen

Nr.	Kurzbezeichnung	Beschreibung	P	V	K	R	Quelle	Ziele
FA-1	Standalone	Alle erforderlichen Anwendungen müssen auf einem Computer installiert und nach dem Systemstart automatisch ausgeführt werden.	↑	↓	→	→	SH	HZ1
FA-2	Track Editor mit SVG	Der Track Editor muss den Track-Piece und Track mit dem SVG-Format ohne eingebettete PNG-Bilder erzeugen.	↑	→	↑	↑	SH	HZ2
FA-3	Statischer Webserver	Das System muss in der Lage sein, den Track Editor als Frontend bereitzustellen.	↑	→	↓	→	H1	UZ1

FA-4	Lokale Multiuser	Mehrere Benutzer/-innen sollen die Fahrzeuge im LAN steuern können.	→	→	↓	↓	SH	HZ1
FA-5	Globale Multiuser	Mehrere Benutzer/-innen können die Fahrzeuge über die Internetverbindung (WAN) steuern.	↓	→	↓	↓	SH	UZ2
FA-6	Verbesserung des Kontrollers	Benutzer/-innen müssen beim Digitalzwilling die Geschwindigkeit und die Fahrspur der Fahrzeuge anpassen können.	↑	↓	→	→	PL	UZ2
FA-7	Neuer Track	Der Track Editor soll mindestens ein neues Track-Piece zusätzlich zu den Track-Pieces der vorherigen Bachelorthesis (Gerade, Kreuzung, Kurve, Verzweigung) anbieten.	→	↑	↑	↑	SH	HZ2
FA-8	Verbesserung der Export-/Import-Funktion	Die Export-/Import-Datei sollte im SVG-Format sein.	→	↓	↓	↓	SH	HZ2
FA-9	Mobile-App	Eine einfache Applikation für ein Smartphone kann zusätzlich für die Fahrzeugkontrolle entwickelt werden.	↓	→	→	→	SH	HZ1

Tabelle 4: Funktionale Anforderung

2.4.2 [NFA] Nicht funktionale Anforderungen

Nr.	Kurzbezeichnung	Beschreibung	P	V	K	R	Quelle	Ziele
NFA-1	Lightweight Frontend	Der Track Editor muss weiterhin ohne Backend-Software funktionieren.	↑	↓	↓	↓	SH	HZ2
NFA-2	Stabilität des Systems	Das System soll stabilisiert werden.	→	↓	↓	↓	PL	HZ1
NFA-3	Clean Code	Die Codes der Anwendung sollen nach den folgenden 4 Eigenschaften programmiert werden: Lesbarkeit, Änderbarkeit, Erweiterbarkeit, und Wartbarkeit.	↓	↓	↓	→	PL	HZ2

Tabelle 5: Nicht Funktionale Anforderung

3 Implementation

3.1 Design und Modelling

3.2 Systemumfeld

Das System: Das "BFH Hyperdrive" soll ein eigenständiges System sein. Der/Die Benutzer/-in kann mit dem Frontend des Systems, ein Track für die Fahrzeuge selbst entwerfen und die Kommunikation mit den Rennwagen herstellen.

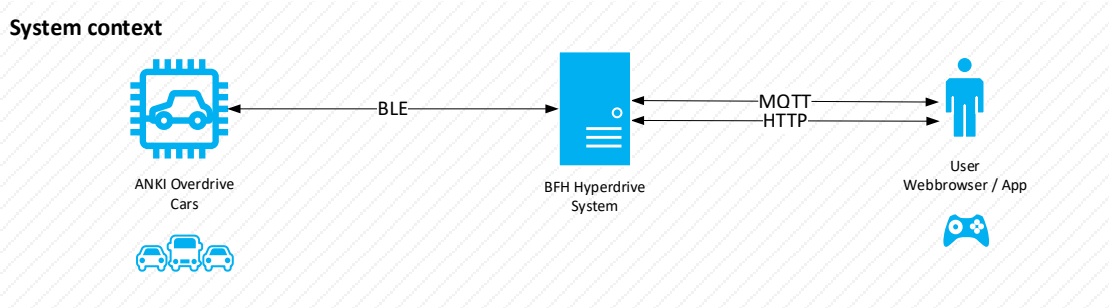


Abbildung 4: Kontextdiagramm

Das Kontextdiagramm stellt die Systemkontextabgrenzung sowie den Datenfluss zu dem System dar.

3.3 Architekturbeschreibungen

Das Standalone-System "BFH Hyperdrive" ist ein Computer, auf dem sich die Geschäftslogik mit einem Fahrzeugkontroller und ein Webdienst des Frontendes befindet, die nach dem Start des Systems automatisch ausgeführt werden. Das Frontend kommuniziert mit dem Fahrzeugkontroller über das MQTT-Protokoll. Die Fahrzeuge können sich über das BLE-Protokoll mit dem Fahrzeugkontroller des Systems verbinden. Der/Die Benutzer/-in kann das Frontend via einem Webbrowser oder einer App bedienen.

3.4 Systemfunktionalität

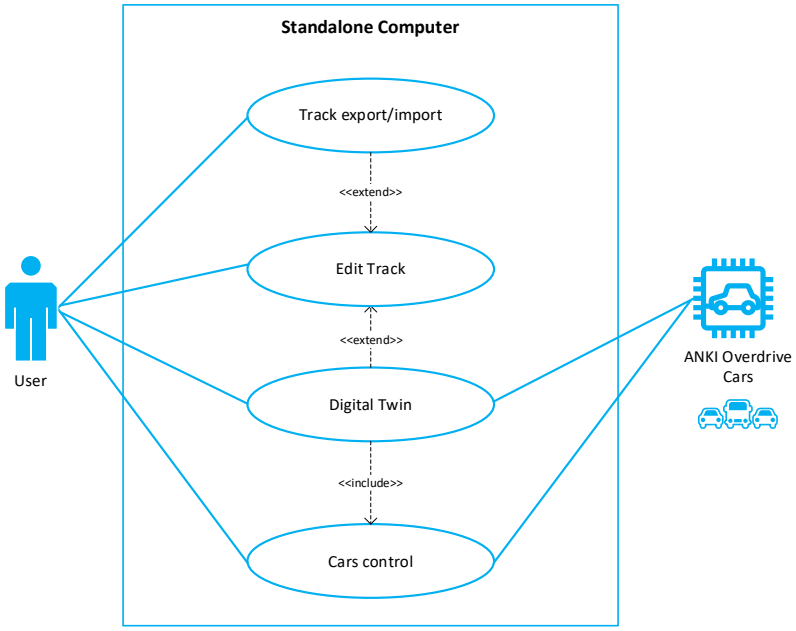


Abbildung 5: Use Case Diagramm

Auf der Abbildung 5 stellt das Use Case Diagramm die Systemfunktionalitäten dar. Mit der Applikation des Systems kann der/die Benutzer/-in den Track erstellen bzw. editieren. Der zusammengestellte Track kann auf dem Computer des/der Benutzers/-in exportiert und importiert werden. Der/Die Benutzer/-in kann die virtuelle Darstellung des physikalischen Fahrzeugs (Digitalzwilling) auf der Webapplikation sehen und währenddessen die verbundenen Fahrzeuge steuern.

3.5 Nutzer und Zielgruppe

Die Zielgruppe des neuen Systems sind Leute in der Entwicklung. Das System kann sich hervorragend als Basissystem für Anwendungsbereiche in der Forschung, wie zum Beispiel Verkehrssimulationen oder in der Steuerungs- und Prozesskontrolle eignen.

3.6 Randbedingung

Das System funktioniert im LAN ohne Internetverbindungen. Für die erweiterten Funktionen wird jedoch eine Internetverbindung erforderlich (Kabel oder Wireless). Der/Die Benutzer/-in muss dafür die Netzwerkeinstellungen des Standalone-Systems manuell im Betriebssystem konfigurieren.

3.7 Annahmen

Abbildung 6: Anki Overdrive Fahrzeuge mit Raspberry Pi

Der/Die Benutzer/-in kann im lokalen Netzwerk ohne Internetverbindung mit den Fahrzeugen, einem Computer mit BFH Hyperdrive und einem ausgedruckten Track, das Anki Overdrive starten. Das System soll optional kollaborativ sein, so dass mehrere Nutzer teilnehmen können. Der/Die Benutzer/-in kann den Track mit dem Track-Editor zusammenstellen und in verlustfreier Bildqualität (SVG) ausdrucken. Die offiziellen Rennbahn-Elemente müssen nicht mehr beschafft werden.

3.8 Technologie des neuen Systems

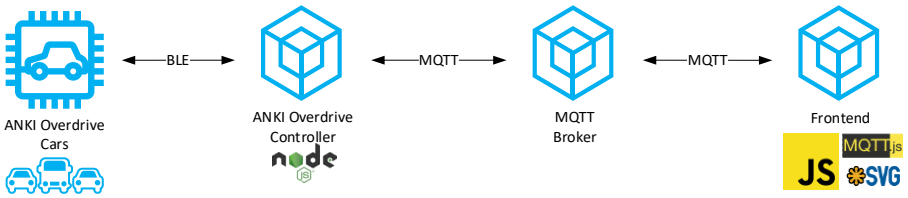


Abbildung 7: Technologie des neuen Systems

Fahrzeugkontroller

Datei-Name	Beschreibung
Node.js	Der Fahrzeugkontroller wurde bei JavaScript-Laufzeitumgebung Node.js ausgeführt.
Noble	Eine Softwarebibliothek für Node.js, welche die Kommunikation mit BLE-fähigen Geräten ermöglicht.

Tabelle 6: Technology Fahrzeugkontroller

Frontend

Datei-Name	Beschreibung
JavaScript	Das Frontend wurde mit JavaScript ohne Hilfe eines Frameworks implementiert.
SVG	Alle Tracks werden mit dem SVG-Format erzeugt.
SPA	Die Single-Page-Webanwendung für die Benutzerfreundlichkeit.
MVC-Pattern	Der Architekturmuster Model-View-Controller für die Softwareentwicklung.
MQTT.js	Eclipse Paho JavaScript Client ermöglicht die Kommunikation zwischen dem Track Editor und MQTT Broker über das Websocket.

Tabelle 7: Technology Frontend

Kommunikation

Datei-Name	Beschreibung
MQTT	Für die Kommunikation zwischen dem Frontend und dem Fahrzeugkontroller.
BLE	Für die Kommunikation zwischen dem Fahrzeugkontroller und den Fahrzeugen.

Tabelle 8: Technology Kommunikation

3.9 Standalone-System

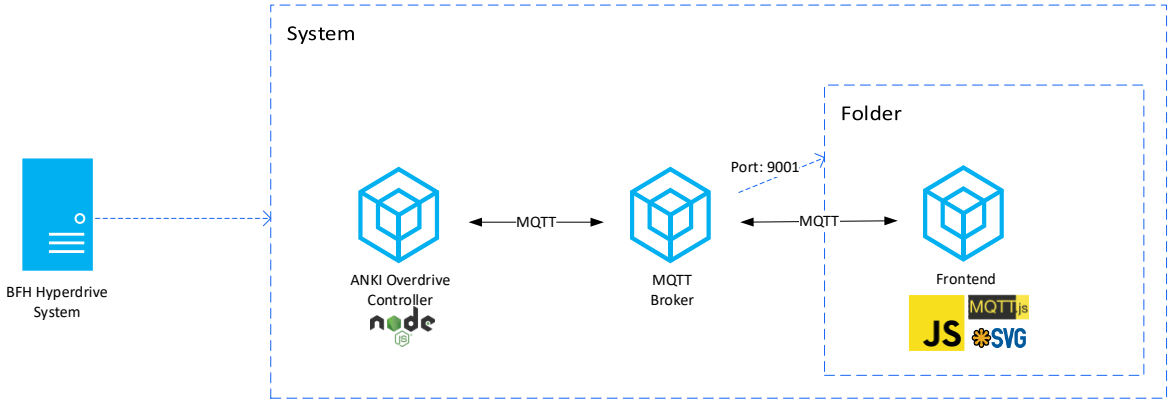


Abbildung 8: BFH Hyperdrive

Das BFH Hyperdrive ist mit dem Betriebssystem Raspbian GNU/Linux 10 ausgestattet und sofort nach dem Einschalten des Computers einsatzbereit. Der Ethernet-Port des Gerätes wurde mit einer statischen IP-Adresse 192.5.5.5/24 konfiguriert.

Auf dem BFH Hyperdrive befinden sich drei Anwendungen. Der Fahrzeug-Kontroller und der MQTT-Broker werden als Systemdienst ausgeführt. Wenn ein Laufzeitfehler vorliegt und den Dienst versagt, wird sie nach 10 Sekunden automatisch neu gestartet.

Das Frontend befindet sich in einem Datei Ordner. Um auf eine Anfrage des Webbrowsers reagieren zu können, wurde eine Portweiterleitung mit MQTT-Broker konfiguriert, die in Kapitel 3.9.2 näher erläutert wird.

3.9.1 Softwarestruktur des gesamten Systems

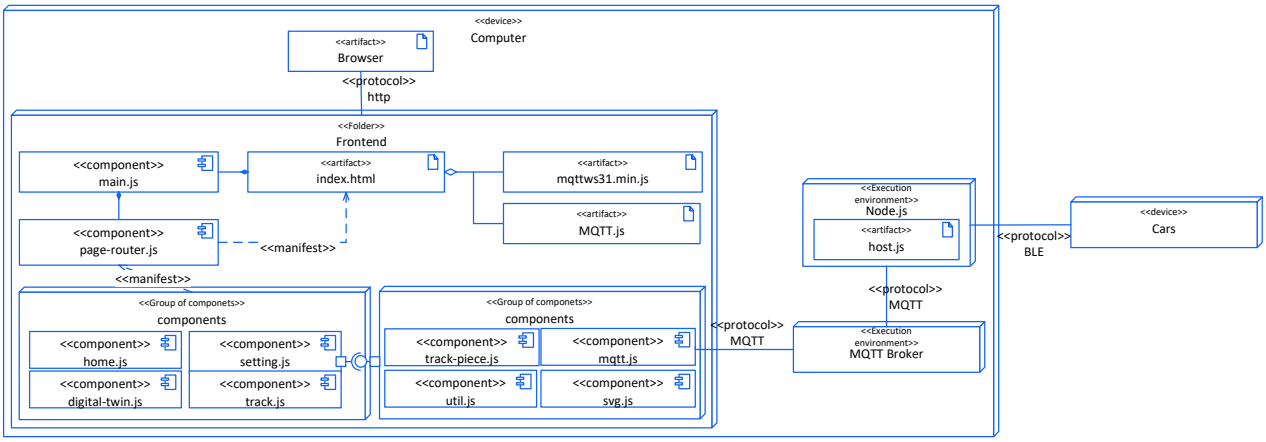


Abbildung 9: BFH Hyperdrive Verteilungsdiagramm

Die Abbildung 9 zeigt ein Verteilungsdiagramm bzw. die Ausführungsarchitektur, welche Hardware-Geräte, Prozessoren und Software-Ausführungsumgebungen Inhalte des Systems sind.

3.9.2 MQTT-Broker

```
pi@raspberrypi:~ $ cat /etc/mosquitto/mosquitto.conf
listener 1883
protocol mqtt
allow_anonymous true

listener 9001
protocol websockets
http_dir /home/pi/Desktop/Frontend
allow_anonymous true
```

Abbildung 10: Mosquitto.conf

Auf der Abbildung 10 ist die Konfiguration des MQTT-Brokers zu erkennen. Die TCP Ports 1883 und 9001 wurden für die Dienste des MQTT-Brokers bereitgestellt. Der Fahrzeug-Kontroller verwendet den Port 1883. Das Frontend verwendet den Port 9001. Die «http_dir» Einstellung sorgt für die Weiterleitung zum Datei Ordner des Frontend. Wenn im Browser die IP-Adresse des Systems zusammen mit dem Port 9001 (z.B. http://192.168.5.5:9001) eingegeben wird, kann der Benutzende den Track Editor erreichen.

3.10 Frontend

Das gesamte Frontend wurde mit JavaScript ohne Hilfe eines Frameworks implementiert. Eine einzige Softwarebibliothek, die Eclipse Paho JavaScript Client «mqttws31.js», ermöglicht die Kommunikation über das Websocket.

Für eine benutzerfreundliche Webseite wird eine Single-Page-Webanwendung mit MVC-Pattern verwendet. Die Beschreibung folgt mit der Tabelle unten:

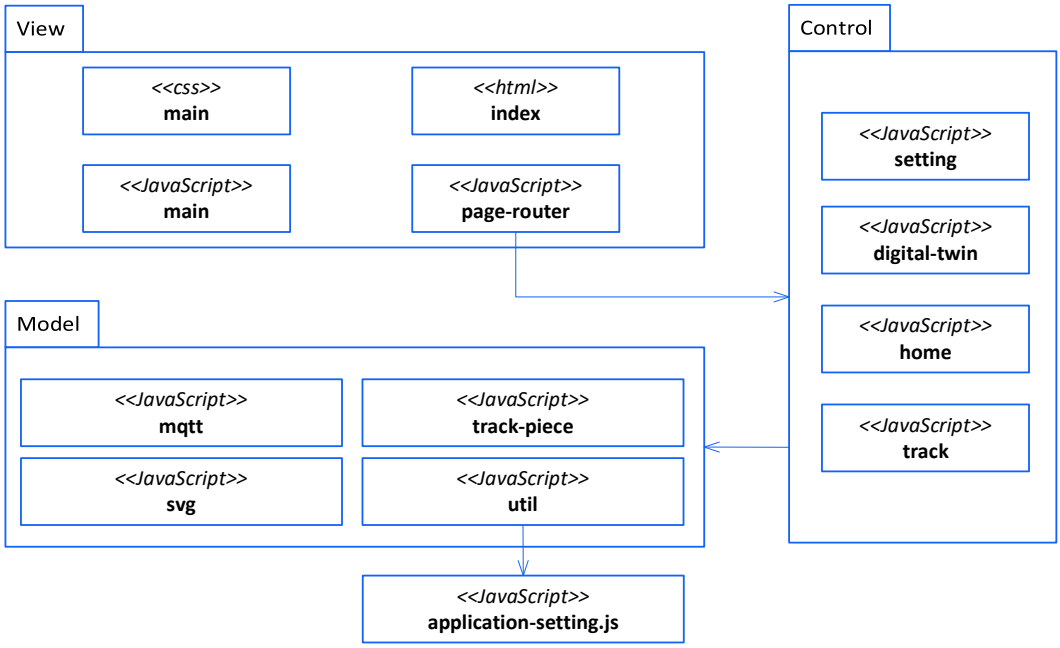


Abbildung 11: MVC-Pattern

Datei-Name	Beschreibung
index.html	Die Hauptseite der Applikation.
main.css	Die Darstellung der Applikation wurde in dieser Datei festgelegt.
main.js	Die erste auszuführende JavaScript-Datei. In der Datei wurden alle Pfade zu den Controllern bei page-router.js registriert.
page-router.js	Der Router sucht aus der ausgewählten Webpage des/der Benutzers/-in einen entsprechenden Controller aus. Der Controller generiert die Inhalte der Webseite als HTML-Elemente und gibt diese an den Router zurück. Der Router überschreibt das "<main>"-Element in index.html. Somit muss die Seite bei jeder Anfrage des/der Benutzers/-in nicht komplett neu geladen werden.
setting.js	Generiert HTML-Elemente und kontrolliert die Funktionen von der "setting"-Seite, welche dem/der Benutzer/-in den Editor Board und die MQTT-Einstellungen ermöglicht. Die weiteren Funktionen sind Export bzw. Import. Der/Die Benutzer/-in kann den Zustand der Track Editor auf dem lokalen Computer speichern bzw. wiederherstellen. Der/Die Benutzer/-in hat die Möglichkeit auch die Tracks als SVG für den Drucker herunterladen.
digital-twin.js	Generiert HTML-Elemente und kontrolliert die Funktionen von der "digital-twin"-Seite, welche dem Benutzer die physikalischen Bilder von den ausgedruckten Streckenteilen und Fahrzeugen visualisiert. Der/Die Benutzer/-in kann die Geschwindigkeit der verbundenen Fahrzeuge anpassen.
home.js	Generiert HTML-Elemente und kontrolliert die Funktionen von der Hauptseite des Track Editors.
track.js	Ist der Kontroller für das Board des Track-Editors.
mqtt.js	Ist zuständig für die MQTT-Kommunikation.
track-piece.js	Ist zuständig für die Daten von Track-Piece auf dem Board des Track-Editors.
svg.js	Ist zuständig für die Bearbeitung der SVG-Daten vom Track-Piece.
util.js	Ist zuständig für die Session-Daten, die Einstellungen und die Import- und Export-Funktionen.
application-setting.js	Bei dieser Datei können die Standardeinstellungen des Frontend festgelegt werden.

Tabelle 9: Beschreibung SPA & MVG

3.10.1 Scalable Vector Graphics (SVG)

Die skalierbare Vektorgrafik wurde von W3C als Standard für Websites empfohlen. Die «Extensible Markup Language» (XML) basierten Texte erzeugen auf dem Browser die zweidimensionale Vektorgrafik.

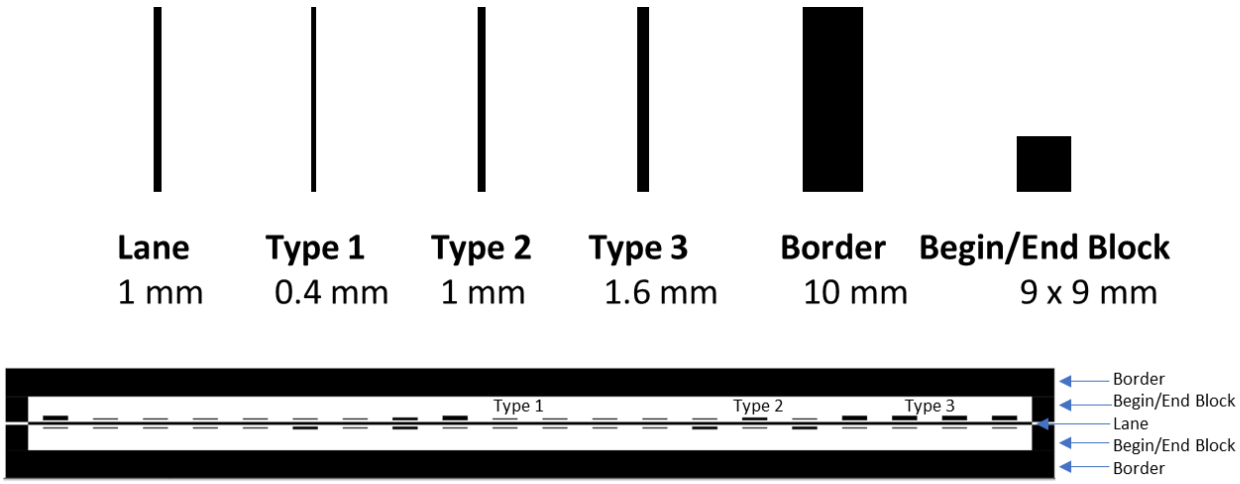


Abbildung 12: Block-Typen

Die Abbildung 12 zeigt die Hauptlinie (Lane) mit den unterschiedlich breiten Block Typen, welche für die Erzeugung des Tracks benötigt werden. Wie die Linie den Code interpretiert, wurde bereits unter Punkt 1.1 erklärt.

Das Aktivitätsdiagramm zeigt, wie ein Track erzeugt wird.

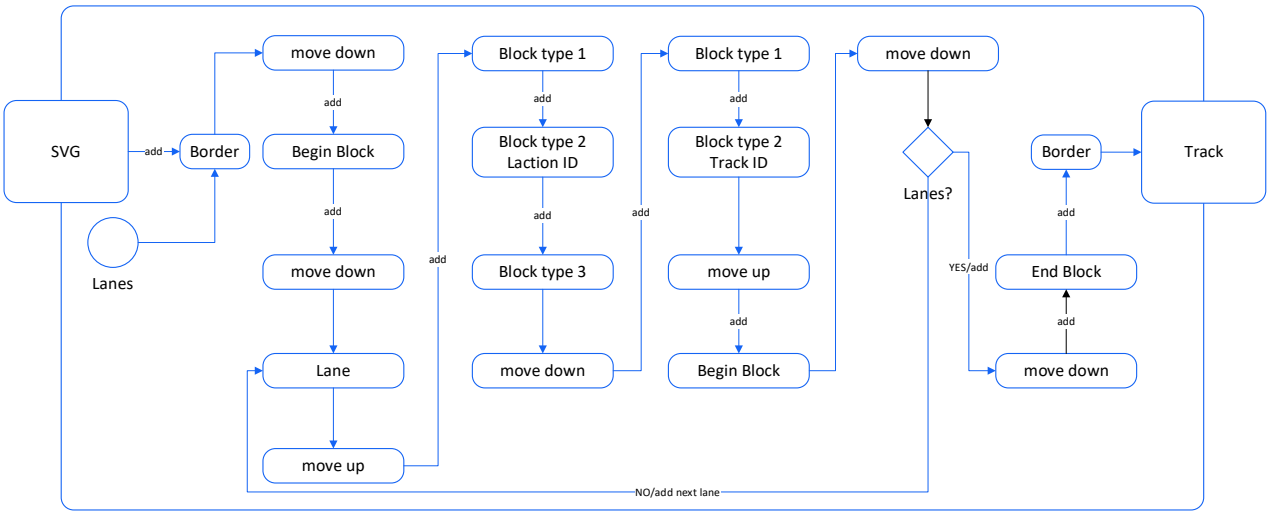


Abbildung 13: Aktivitätsdiagramm vom Track mit einem Pfad

Schritte	Beschreibung
Start	Startet die Erstellung mit der Anzahl der Spuren.
1	Ein SVG-Tag mit den benötigten Namensräumen und den Attributen werden erzeugt. Die Anfangskoordination wird berechnet.
2	Der Rand der Fahrbahn wird hinzugefügt.
3	Die Koordination wird neu berechnet.
4	Der Anfangsblock wird hinzugefügt.
5	Die Koordination wird neu berechnet.

6	Die Fahrspur wird hinzugefügt.
7	Die Koordination wird neu berechnet.
8	Der Block type 1 wird hinzugefügt.
9	Der Block type 2 wird hinzugefügt, und wird teilweise Block type 1 überschrieben.
10	Der Block type 3 wird hinzugefügt, und wird teilweise Block type 1 überschrieben.
11	Die Koordination wird neu berechnet.
12	Der Block type 1 wird hinzugefügt.
13	Der Block type 2 wird hinzugefügt und wird teilweise den Block type 1 überschreiben.
14	Die Koordination wird neu berechnet.
15	Der Anfangsblock wird hinzugefügt.
16	Die Koordination wird neu berechnet.
17	Wiederholung, bis die Anzahl der Spuren erreicht hat.
18	Die Koordination wird neu berechnet.
19	Der Anfangsblock wird hinzugefügt.
20	Die Koordination wird neu berechnet.
21	Der Rand der Fahrbahn wird hinzugefügt.
22	Track wird erzeugt.

Tabelle 10: Beschreibung des Aktivitätsdiagramms

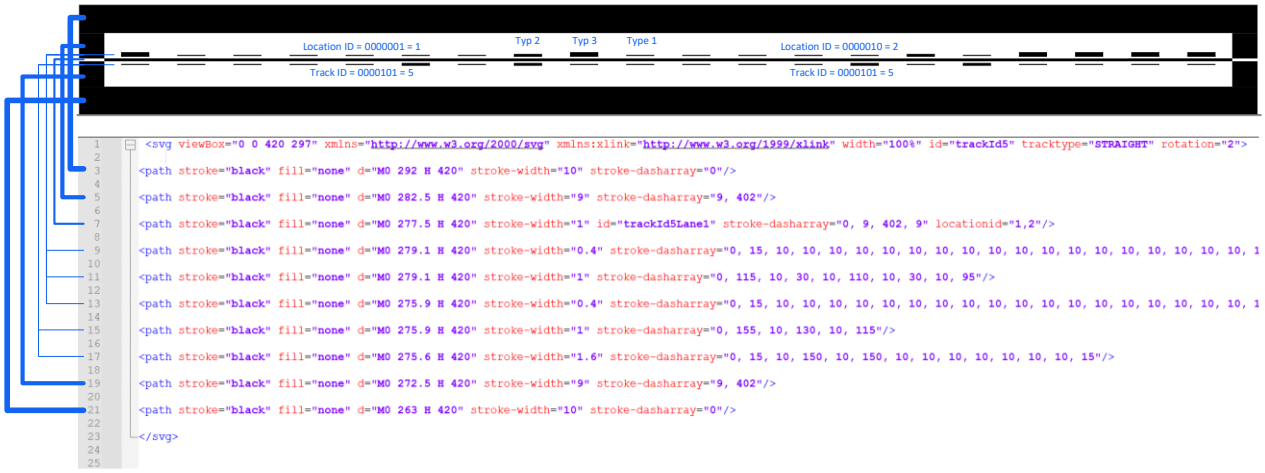


Abbildung 14: SVG-Code Beispiel

Die Abbildung 14 zeigt ein Beispiel von einer geraden Track-Piece mit einer Fahrspur. Alle Block-Typen werden als Path-Element in der SVG hinzugefügt.
Die genaue Beschreibung folgt mit der untenstehenden Tabelle.

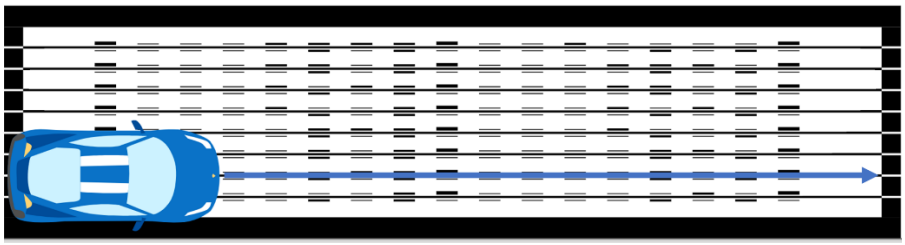
Zeile / Tag	Beschreibung
1 SVG-Tag	Das SVG-Tag muss normalerweise in zwei Namensräume eingebunden werden. **xmlns**="http://www.w3.org/2000/svg" ist der Standardnamensraum für SVG **xmlns**:xlink=http://www.w3.org/1999/xlink. Das **viewBox**-Attribut legt die Anfangskoordination und die Grunddimension der Grafik fest (x-Koordination, y-Koordination, Breite, Höhe). Das **width**-Attribut legt die Breite fest, wie die SVG im Verhältnis zu seinem Eltern-Objekt dargestellt wird. Die **id**-, **tracktype**- , **rotation**-Attribute sind für die Funktionen des Frontends z.B. die Animation, die Rotation des Track Pieces usw. bestimmt.

3 bis 21 Path-Tag	Das **stroke**-Attribut legt die Farbe für den Pfad fest. Das **fill**-Attribut legt die Füllfarbe für den Pfad fest. Das **d**-Attribut legt die Koordinaten und Kommandos für Pfade fest. Das **stroke-width**-Attribut legt die Breite der Linie fest. Das **stroke-dasharray**-Attribut legt ein Strichmuster für die Linie fest, welches der wichtigste Teil des Pfades ist. Die Track ID und die Location IDs werden mit diesem Attribut erzeugt. Die **id**-, **locationid**-Attribute sind für die Funktionen des Frontends z.B. die Animation, die Rotation des Track Pieces usw. bestimmt.

Tabelle 11: SVG Attribute Beschreibung

Sämtliche Erklärungen der SVG-Attribute befinden sich auf https://wiki.selfhtml.org/wiki/SVG/Attribute.[3]

3.10.2 SVG Animation



```
▼<image href="src/pics/bfh_car.svg" width="100" height="100" id="d205effe02cb" transform="scale(-1,1) translate(-60, -60)">
    <animateMotion dur="2572ms" begin="0s" repeatCount="1" path="M0 277.5 H 420" rotation="auto"></animateMotion>
  </image>
</svg>
```

Abbildung 15: SVG Animation Code

Für die Animation ist es einfacher. Es wird eine weitere SVG der existierenden SVG hinzugefügt. Die zu animierende SVG bekommt ein «animationMotion»-Tag. Die Erklärung der Attribute folgt mit der folgenden Tabelle.

Arttribut	Beschreibung
dur	Legt die Zeit fest, wie viele Millisekunden die Animation dauert. Dieser Wert wird von der aktuellen Geschwindigkeit des Fahrzeuges berechnet. $$\frac{Pfadlänge * 1000}{Geschwindigkeit}$$
begin	Legt die Zeit fest, wann die Animation anfangen soll.
repeatCount	Legt die Anzahl der Animationswiederholung fest.
path	Ist der Pfad für die Animation. Dieser Pfad ist das **d**-Attribut der Spur, welche das Animations-Auto darauf animieren soll.
rotation	Der Wert «auto» legt fest, ob das Animations-Fahrzeug automatisch parallel zu dem Pfad rotieren soll.

Tabelle 12: SVG Animation Attribute Beschreibung

3.10.3 SVG Track-Piece Type

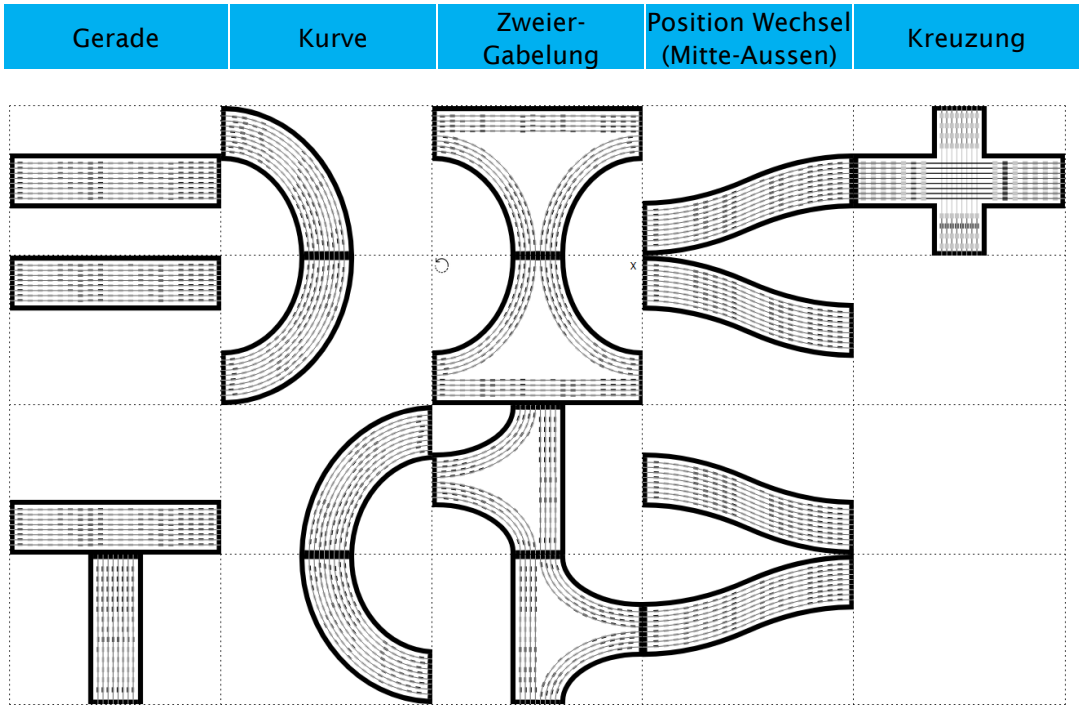


Abbildung 16: Track-Piece Typen

Der Track-Editor hat insgesamt 5 verschiedene Track-Piece Typen zur Auswahl:

- Gerade mit 4 Rotationsmöglichkeiten.
- Kurve mit 4 Rotationsmöglichkeiten.
- Zweier-Gabelung mit 4 Rotationsmöglichkeiten.
- Kreuzung.
- Position Wechsel (Mitte-Aussen) mit 4 Rotationsmöglichkeiten.

Das Track-Piece «Mitte-Aussen» und «Vertikale Gerade» wurde nach den Anforderungen neu entwickelt. Das Track-Piece «Zweier-Gabelung» wurde mit den graden Linien verbessert, ebenfalls wurde dessen Rotationsmöglichkeiten neu hinzugefügt.

Die Dimension des Track Pieces war bei der vorhergehenden Thesis quadratisch gewesen. Bei dieser Thesis kann die Grösse des Track-Piece nun auf A3 Papiergrösse angepasst werden.

Somit ist auch das Ausdrucken der Tracks auf einem Standard A3 Drucker einfacher, da keine Skalierung erfolgen muss. Anmerkung: Viele Drucker passen die Skalierung beim Drucken automatisch an, was dazu führt das die Tracks nicht aufeinanderpassen.

3.10.4 MQTT-Topics

Die folgenden Tabellen zeigen die relevanten MQTT-Topics für das Frontend. Die vollständige Topic Struktur wurde in einer Bachelorthesis "Anki Overdrive Track Editor" von Herrn Dominique Marc Hofmann [2] erläutert.

Die Topic Struktur wird grundsätzlich zwischen drei verschiedenen Arten von Topics unterschieden:

- **Intent** Ein Befehl.
- **Status** Zustand des Fahrzeugkontrollers und Frontend.
- **Event** Event wird ausgelöst, wenn neue Informationen vorhanden sind.

Intent
Anki/Host/{Host Name}/I Befehl, dass der Fahrzeugkontroller die Verbindung mit den Fahrzeugen herstellen soll.
Anki/{Fahrzeug ID}/I Befehl, der an den Fahrzeugkontroller gesendet wird und entsprechende Handlung beim Fahrzeug ausführt: z.B. Spurwechsel, Geschwindigkeitsänderung usw.

Tabelle 13: MQTT Intent

Status
Anki/Host/{Host Name}/S/HostStatus Bei diesem Topic wird der Status des Fahrzeugkontrollers übermittelt.
Anki/Host/{Host Name}/S/{Fahrzeug ID} Bei diesem Topic wird der Status des Fahrzeuges übermittelt.

Tabelle 14: MQTT Status

Event
Anki/{Fahrzeug ID}/E/track_piece_id Bei diesem Topic wird die Track-Piece ID des Fahrzeugs übermittelt.
Anki/{Fahrzeug ID}/E/track_location_id Bei diesem Topic wird die Track Location-ID des Fahrzeugs übermittelt.
Anki/{Fahrzeug ID}/E/E/speed Bei diesem Topic wird die aktuelle Geschwindigkeit des Fahrzeugs übermittelt.
Anki/{Fahrzeug ID}/E/Messages/ **ANKI_VEHICLE_MSG_V2C_LOCALIZATION_TRANSITION_UPDATE** Bei diesem Topic wird die aktuelle Geschwindigkeit des Fahrzeugs übermittelt.
Anki/WebClient/E/Share Bei diesem Topic wird die Eigenschaften des Tracks für die kollaborative Funktion übermittelt.

Tabelle 15: MQTT-Event

3.10.5 Track Sharing – SVG über MQTT

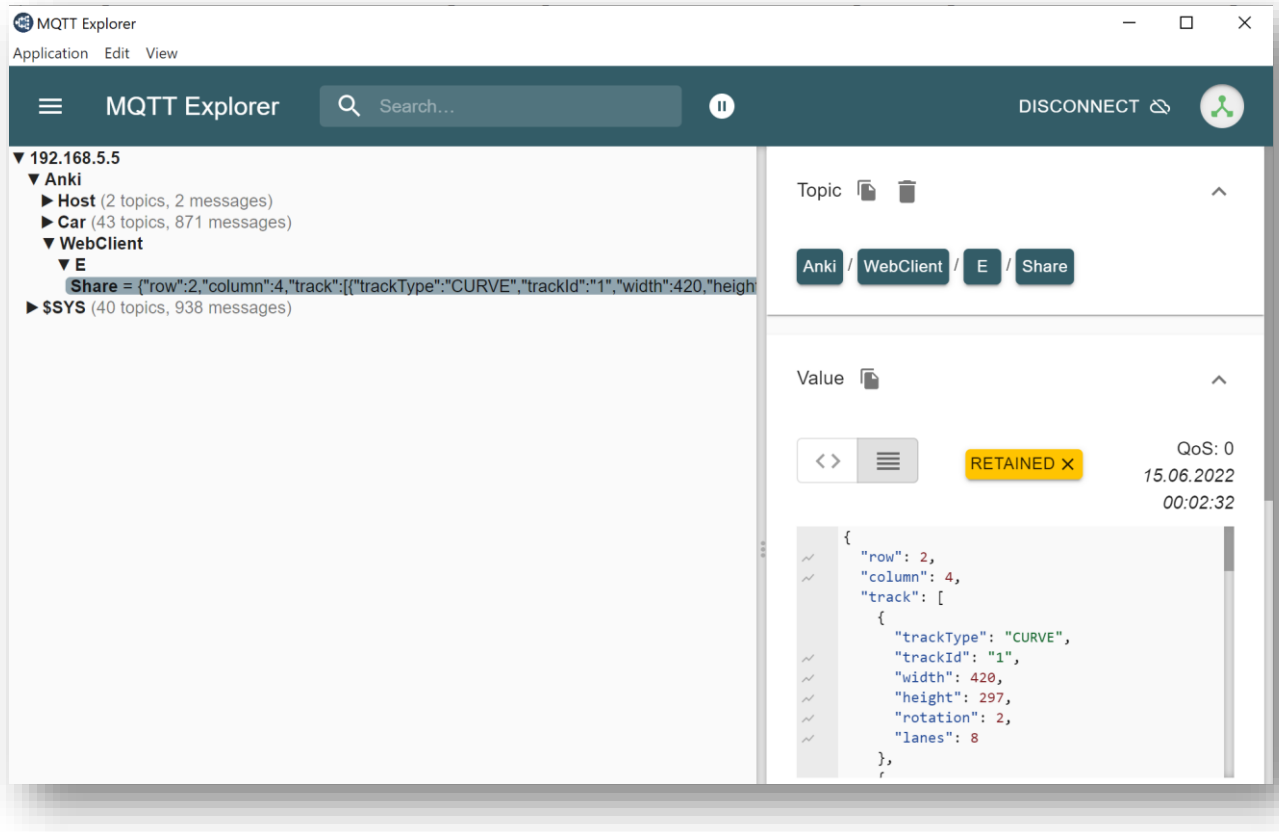


Abbildung 17: Track Sharing Beispiel

Das System ist kollaborativ mit weiteren Systemen. Alle Systeme sollen einen gleichen Track haben. Das Frontend wird entwickelt, um die Eigenschaften der Tracks mit einer MQTT-Nachricht senden zu können. Der neue Bridge-Client bekommt diese Nachricht und kann sich den Track im Editor oder in dem Digitalzwilling zeigen lassen.

Bei dieser MQTT-Nachricht wird das «retained flag» gesetzt. Mit Hilfe von <retained flag» erhalten neu angemeldete Clients die letzte Nachricht unmittelbar, nachdem sie das MQTT-Topic abonniert haben.

Die erste Idee war, das ganze SVG mit der MQTT-Message zu senden, weil das Protokoll die Message bis zu einer Grösse von 260 Megabyte [4] übertragen kann. Beim Test wurde festgestellt, dass alles über einer Nachrichtlänge von 20'000 Zeichen (20KB) abgeschnitten wurde. Das Problem liegt an der Programmbibliothek von mqtt.js.

Aus diesem Grund wird das SVG nicht direkt mit der MQTT-Nachricht verteilt.

4 Ergebnisse

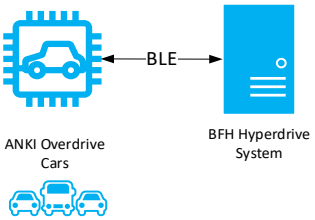


Abbildung 18: BFH Hyperdrive Ergebnis

Ergebnisse des Systems sind:

- Ein Standalone-System, welches nach dem Start des Computers sofort bereit ist.
- Der Track Editor kann die Streckenbilder mit dem SVG Format erzeugen.
- Im Gegensatz zu PNG ist die SVG-Dateigrösse um bis zu 40mal kleiner geworden.
- Die Qualität der ausgedruckten Strecken ist sehr gut.
- Die Animation des Digitalzwillings ist fliessend und präziser geworden.
- Optional kann es im Netzwerk mit weiteren «BFH Hyperdrive» kollaborativ arbeiten.

4.1 Frontend

4.1.1 Track Editor

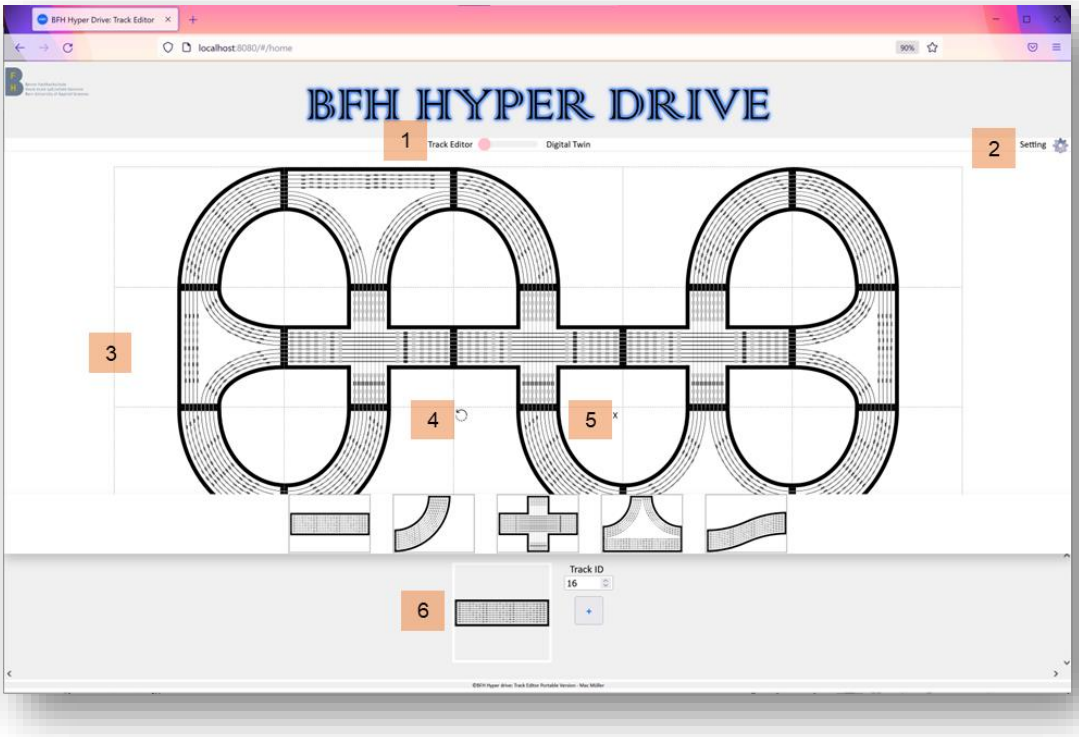


Abbildung 19: Frontend Track Editor

Auf der Hauptseite des Frontends findet der/die Benutzer/-in den Track Editor wie es die Abbildung 19 zeigt. Die numerierten Komponenten werden wie folgt beschrieben:

Nr.	Beschreibung
1	Navigation Switch zwischen Track Editor und Digitalzwilling.
2	Navigation zur Einstellung.
3	Das Board des Track Editors.
4	Die Rotation-Funktion.
5	Die Löschfunktion.
6	Board-Kontroller: Ein neues Track-Piece hinzufügen.

Tabelle 16: Track Editor Komponenten

4.1.2 Digitalzwilling

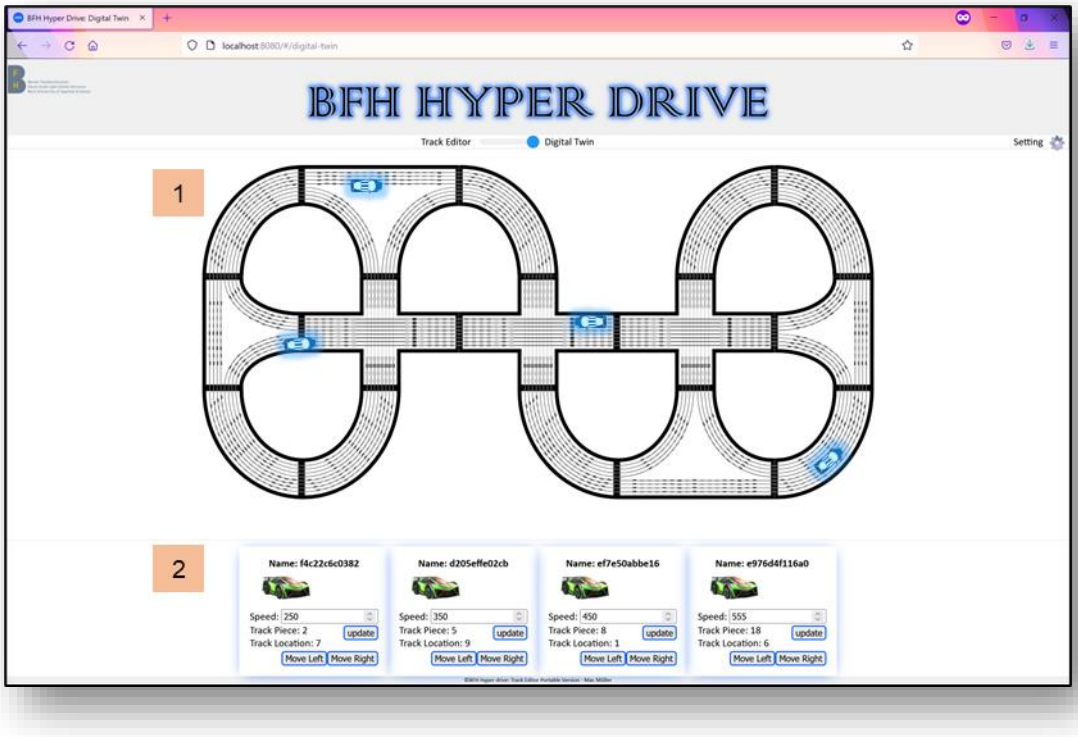


Abbildung 20: Frontend Digitalzwilling

Auf der Digitalzwilling Seite findet der/die Benutzer/-in die Darstellung wie es die Abbildung 20 zeigt. Die numerierten Komponenten werden wie folgt beschrieben:

Nr.	Beschreibung
1	Digitalzwilling Board.
2	Fahrzeugkontroller.

Tabelle 17: Digitalzwilling Komponenten

Folgende Funktionen stehen beim Fahrzeugkontroller zur Verfügung:

- Geschwindigkeitsbestimmung.
- Spurwechsel nach links oder rechts.
- Anzeige der Positionswerte.

4.1.3 Setting

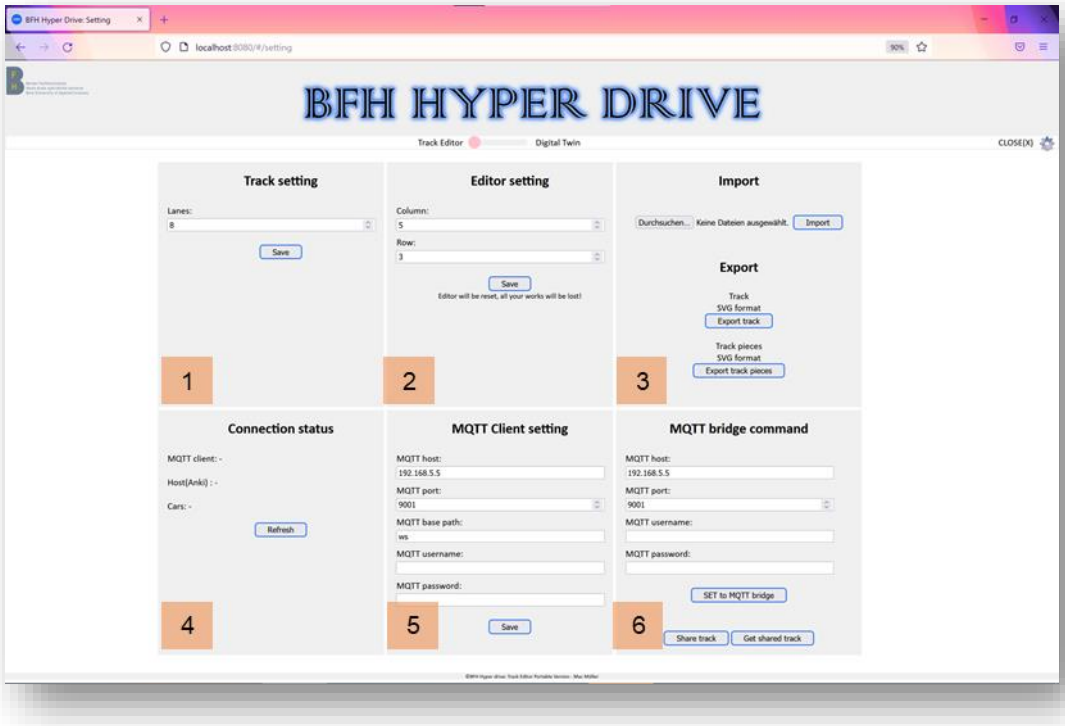


Abbildung 21: Frontend Setting

Auf der Setting-Seite findet der/die Benutzer/-in die Einstellungsmöglichkeiten wie es die Abbildung 21 zeigt. Die numerierten Komponenten werden wie folgt beschrieben:

Nr.	Beschreibung
1	Einstellung für die Anzahl der Spuren.
2	Einstellung des Editor Boards.
3	Funktionen: SVG Import/Export. Beim Import wird das Frontend anhand der SVG-Tags automatisch erkennen, ob es sich um einen Track oder ein Track-Piece handelt.
4	Testfunktion für die Verbindung zum Anki Overdrive Host und Fahrzeuge.
5	Einstellungen für den lokalen MQTT Broker.
6	MQTT-Bridge Einstellung: Funktion: Track über MQTT teilen. Funktion: Geteilter Track von MQTT auf das Editor Board laden.

Tabelle 18: Einstellung Komponenten

4.2 System Kollaborativ

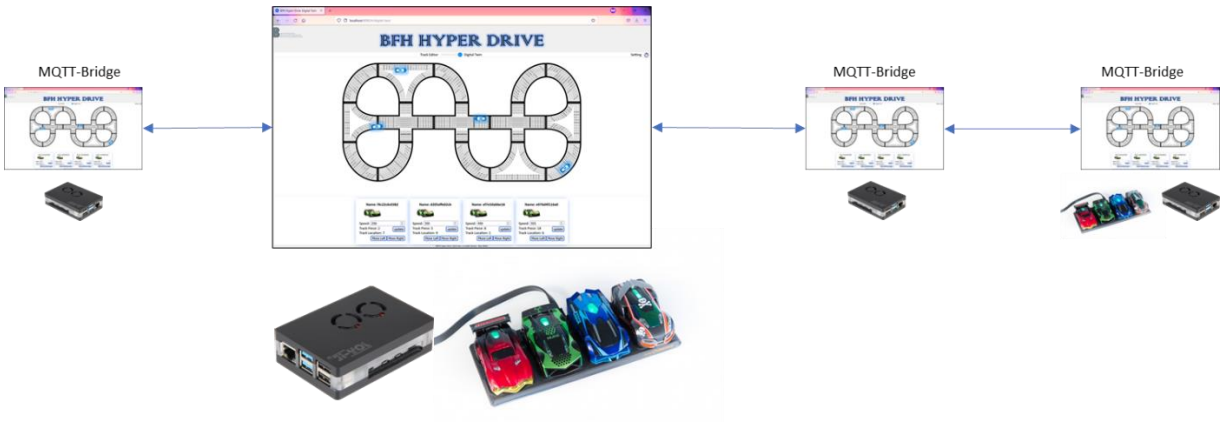
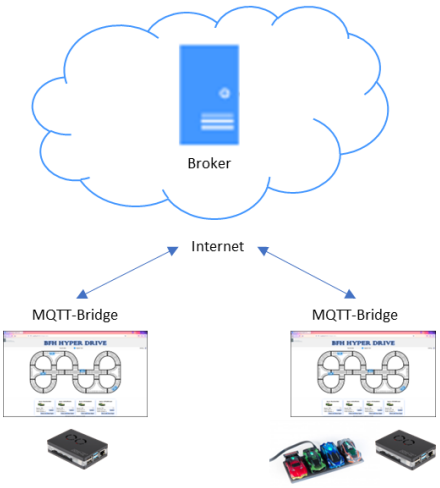


Abbildung 22: System Peer-to-Peer Bridge

Für die kollaborative Arbeit mit anderen BFH-Hyperdrive Systemen soll ein System als Bridge definiert werden. Dieses System wird die Konfiguration seines MQTT-Brokers anpassen. Der Bridge-Broker funktioniert wie eine Kopie des verbindenden Brokers.

Die Systeme können eine Peer-to-Peer Bridge-Verbindung (Abbildung 22) herstellen. Optional könnten die Systeme auch mit einem zusätzlichen zentralen Broker operieren. Wie die folgende Abbildung 23 zeigt:



Um eine Bridge-Verbindung herzustellen, müssen die BFH Hyperdrive Geräte über eine Netzwerk-Verbindung erreichbar sein.

Der/Die Benutzer/-in muss ausreichende Kenntnisse haben, um Netzwerk-Einstellung im Betriebssystem ändern zu können [5].

Abbildung 23: Optional mit einem zentralen Broker

4.3 Test

Die folgende Tabelle zeigt welche Funktionen getestet wurden:

Nr.	Beschreibung	Ergebnis
1	**Frontend**	
1.1	**Track Editor**	
1.1.1	Navigation Switch zwischen Track Editor und Digitalzwilling.	bestanden
1.1.2	Navigation zur Einstellung und zurück zum Track Editor/Digitalzwilling.	bestanden
1.1.3	Ein Track-Piece hinzufügen, löschen, rotieren.	bestanden
1.1.4	Drag und Drop Funktion.	bestanden
1.2	**Digitalzwilling Board**	
1.2.1	Geschwindigkeit des Fahrzeugs ändern.	bestanden
1.2.2	Spurwechseln nach Links und Rechts.	bestanden
1.2.3	Anzeige der Positionswerte mit MQTT-Nachricht vergleichen.	bestanden
1.2.4	Korrektheit des Digitalzwillings - Abbildung 24.	bestanden
1.3	**Setting**	
1.3.1	Einstellung zur Anzahl der Spuren.	bestanden
1.3.2	Einstellung des Editor Boards.	bestanden
1.3.3	Funktionen: SVG Import/Export.	bestanden
1.3.4	Funktion: Verbindungstest.	bestanden
1.3.5	Einstellung für den lokalen MQTT Broker werden übernommen.	bestanden
1.3.6	MQTT-Bridge Einstellungen.	bestanden
1.3.7	Funktion: Track über MQTT teilen.	bestanden
1.3.8	Funktion: Geteilter Track von MQTT auf das Editor Board laden.	bestanden
2	**Standalone-System**	
2.1	MQTT-Service, Bridge Verfügbarkeit.	bestanden
2.2	Kontroller-Service Verfügbarkeit.	bestanden

Tabelle 19: Test

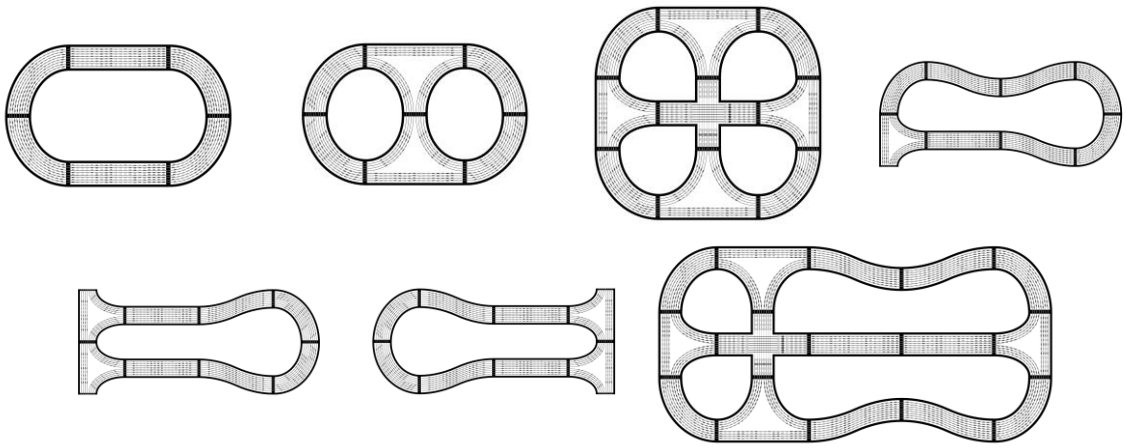


Abbildung 24: Test-Tracks

Für die Tests bei der Nr. 1.2.4, wurden die Tracks wie in der Abbildung 24 verwendet.

4.4 Installationsanleitung

Das System kann mit den folgenden Anweisungen als ein neues BFH Hyperdrive aufgebaut werden:

Voraussetzung Computer

- Linux Betriebssystem
- Eclipse Mosquitto MQTT-Broker
- Node.Js 10.13.x oder höher
- Chromium Browser 98.0 oder höher

Auf dem Computer sollte eine statische IP-Adresse für die Ethernet-Schnittstelle festgelegt werden. z.B.192.168.5.5/24.

Den Code von BFH Hyperdrive aus dem Gitlab-Repository herunterladen.
https://gitlab.ti.bfh.ch/mullk8/Bachelorthesis_BFH_Hyperdrive
Das File «CarsController.zip» muss entpackt werden.

Der MQTT-Broker soll wie folgt konfiguriert werden.
Die Konfigurationsdatei befindet sich auf dem «/etc/mosquitto/mosquitto.conf» Verzeichnis.

```
listener 1883
protocol mqtt
allow_anonymous true

listener 9001
protocol websockets
http_dir {Paht des Frontends-Verzeichnis}
allow_anonymous true

#######################
##### Bridge Config #####
```

Um den Fahrzeugkontroller als System-Service zu bestimmen, soll folgende Textdatei im Ordner «/etc/systemd/system/anki_car_controller.service» **erstellt werden.**

```
[Unit]
Description=Anki Overdrive Car Controller

Wants=network.target
After=syslog.target network-online.target

[Service]
Type=simple
ExecStart=node {Path des Fahrzeugkontrollers}
Restart=always
RestartSec=5
KillMode=process

[Install]
WantedBy=multi-user.target
```

Die folgenden Services aktivieren und starten mit den folgenden Befehlen:

- sudo systemctl enable mosquitto.service
- sudo systemctl enable anki_car_controller.service
- sudo systemctl restart mosquitto.service
- sudo systemctl restart anki_car_controller.service

Das Frontend kann nun direkt über den Browser des Systems oder über die IP-Adresse im LAN z.B. http://192.168.5.5:9001 aufgerufen werden.

5 Projekt Management

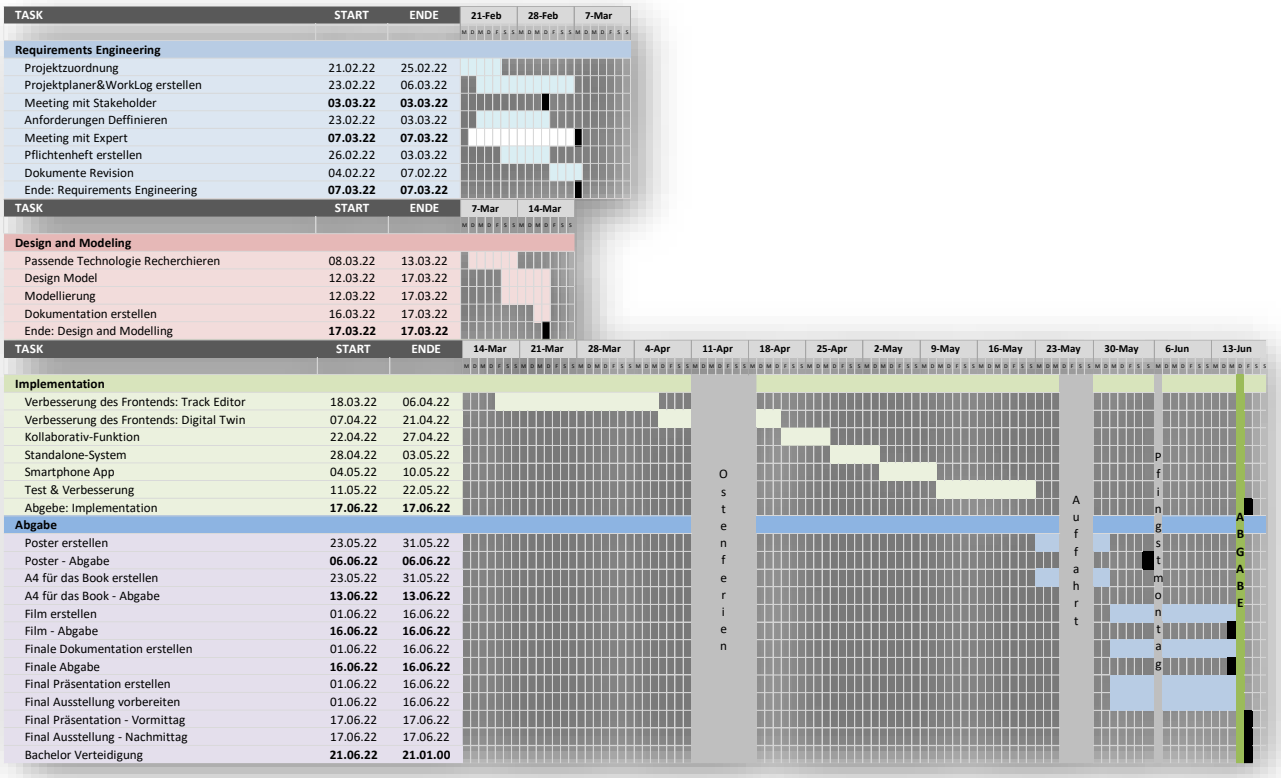
Zu Beginn des Projektes wurden folgende Arbeitsabläufe festgelegt:

- Jede zweite Woche, ein bis zwei Meetings.
- Code aus der Entwicklung wird im GitLab-Repository publiziert.

Während der Sprints gab es zwei Fragen, die beantwortet werden mussten, bevor der neue Sprint beginnen konnte.

- Was soll im kommenden Sprint entwickelt werden?
- Wie wird diese Arbeit erledigt oder das Ziel erreicht?

Die Termin-Planung erfolgte mit einem Excel Sheet:



TASK	START	ENDE
Requirements Engineering		
Projektzuordnung	21.02.22	25.02.22
Projektplaner&WorkLog erstellen	23.02.22	06.03.22
Meeting mit Stakeholder	**03.03.22**	**03.03.22**
Anforderungen Deffinieren	23.02.22	03.03.22
Meeting mit Expert	**07.03.22**	**07.03.22**
Pflichtenheft erstellen	26.02.22	03.03.22
Dokumente Revision	04.02.22	07.02.22
Ende: Requirements Engineering	**07.03.22**	**07.03.22**
Design and Modeling		
Passende Technologie Recherchieren	08.03.22	13.03.22
Design Model	12.03.22	17.03.22
Modellierung	12.03.22	17.03.22
Dokumentation erstellen	16.03.22	17.03.22
Ende: Design and Modelling	**17.03.22**	**17.03.22**
Implementation		
Verbesserung des Frontends: Track Editor	18.03.22	06.04.22
Verbesserung des Frontends: Digital Twin	07.04.22	21.04.22
Kollaborativ-Funktion	22.04.22	27.04.22
Standalone-System	28.04.22	03.05.22
Smartphone App	04.05.22	10.05.22
Test & Verbesserung	11.05.22	22.05.22
Abgebe: Implementation	**17.06.22**	**17.06.22**
Abgabe		
Poster erstellen	23.05.22	31.05.22
Poster - Abgabe	**06.06.22**	**06.06.22**
A4 für das Book erstellen	23.05.22	31.05.22
A4 für das Book - Abgabe	**13.06.22**	**13.06.22**
Film erstellen	01.06.22	16.06.22
Film - Abgabe	**16.06.22**	**16.06.22**
Finale Dokumentation erstellen	01.06.22	16.06.22
Finale Abgabe	**16.06.22**	**16.06.22**
Final Präsentation erstellen	01.06.22	16.06.22
Final Ausstellung vorbereiten	01.06.22	16.06.22
Final Präsentation - Vormittag	17.06.22	17.06.22
Final Ausstellung - Nachmittag	17.06.22	17.06.22
Bachelor Verteidigung	**21.06.22**	**21.01.00**

Abbildung 25: Terminplan

Die folgenden Tabellen zeigen, die «Sprint» Planungen und welche Aufgaben in diesen durchgeführt worden sind:

Aufgabe	Plan Sprint	Implem. Sprint	Plan Stunden	Aufgewendete Stunden
Sprint 1: Requirement Engineering				
Projektzuordnung	1	1	4	4
Pflichtenheft erstellen	1	1	28	26
Projektplaner & WorkLog erstellen	1	1	6	6
Sprint 2: Design and Modelling				
Passende Technologien recherchieren	**2**	2	8	7
System Design	2	2	4	4
Modellierung (UML)	2	2	4	4
Dokumentation erstellen	**2**	2	4	4
Sprint 3: Verbesserung des Frontends: Track Editor				
Track Piece: Gerade	3	3	5	5
Track Piece: Kurve	3	3	8	12
Track Piece: Kreuzung	3	3	7	7
Track Piece: Zweier-Gabelung	3	3	8	4
Track Piece: Neu, Mitte-Aussen, Design	3	3	12	12
Integration alle Track Pieces ins Frontend	3	3	8	8
Track Editor Test	3	3	4	4
Verbesserung des Track Editors nach dem Test	3	3	4	4
Sprint 4: Verbesserung des Frontends: Digitalzwilling				
Digitalzwilling: Fahrzeuge	4	4	17	20
Pfad Animation	4	4	8	12
Funktion: Spur-Wechseln	4	4, 5	8	16
Digitalzwilling: Track Piece	4	4	10	12
Smartphone App	4	-		
Sprint 5: Kollaborativ & Standalone-System				
Funktion: Spur-Wechseln Probleme	4	5	8	10
Funktion: Kollaborativ	5	5	8	12
Standalone: Frontend	5	5	6	6
Standalone: Backend	5	5	12	10
Sprint 6: Test & Verbesserung				
Test: Track Editor	6	6	8	15
Test: Digitalzwilling	6	6	12	20
Test: Setting	6	6	8	20
Test: Standalone System	6	6	8	4
Code Verbessrung & Debug	6	6	24	32
Sprint 7: Code Cleaning				
Code kontrollieren	7	7	8	10
Summe			**259**	**310**

Tabelle 20: PM Planung

Administrative Aufgaben

Aufgabe	Plan Stunden	Aufgewendete Stunden
Dokumentationen für die Abgabe (BookTool, Thesis)	48	58
Präsentationsvorbereitung	24	23
Videoerstellung	16	16
Besprechungen (Vorbereitung, Betreuer, Experte)	15	8
Summe	**103**	**105**

Tabelle 21: Administratives

Projekt

Aufgabe	Plan Stunden	Aufgewendete Stunden
Summe: Sprints	259	310
Summe: Administrative	103	105
Summe: Projekt	**361**	**415**

Tabelle 22: Summe: Projekt

5.1 Probleme und Herausforderungen

5.1.1 Nicht lesbarer Code

Bis zu dieser Thesis wird der Track entweder mit der kurvigen Spur oder mit der geraden Spur zusammengestellt. Die Schlangen-Spur (Positionswechsel-Spur) wurde bei der aktuellen Thesis entwickelt. Am Anfang konnte das Fahrzeug den Code auf dem neuen Track nicht lesen. Nach einigen Experimenten wurde festgestellt, dass die Grafik in der X-Koordinate, je nach Rotation um +/- 1 Millimeter angepasst werden musste.
Dies wurde auch im Code des Frontend entsprechenden angepasst.

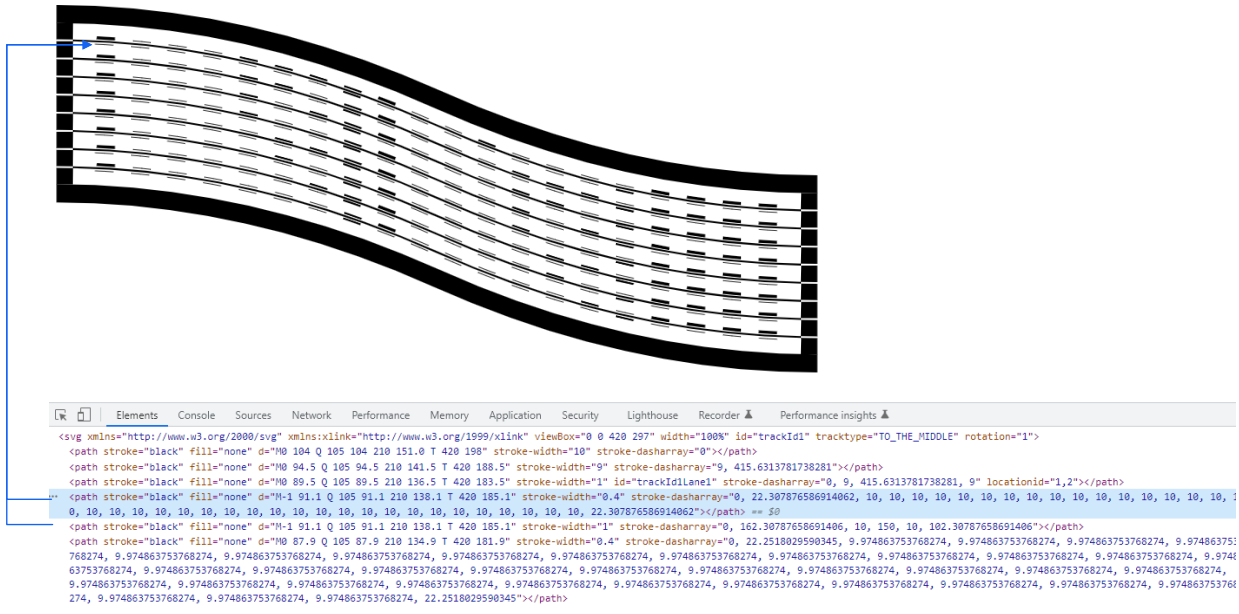


Abbildung 26: Positionswechsel-Spur

Die Abbildung 26 zeigt, dass bei den X-Koordinaten von 2 Pfaden (Block Type 1&2), das **d**-Attribut um minus 1mm korrigiert werden musste.

5.1.2 Unzuverlässigkeit des Fahrzeugs

Für die Animation benötigt die Digitalzwilling-Funktion, die abgelesenen Positionsdaten des Fahrzeugs. Das Fahrzeug liefert jedoch oft keine Positionsdaten zum Fahrzeugkontroller. Dieses Problem tritt auch bei dem originalen Track von Anki Overdrive auf. Bisher konnte der Grund für das Problem nicht evaluiert werden. Das Fahrzeug meldet sich aber trotzdem zuverlässig, wenn es ein Track-Piece verlässt. Um das Projekt im Terminplan nicht zu gefährden, wurde dieses Verhalten der «Blackbox» nicht weiterverfolgt.

Um diesen Fehler zu übergehen/tolerieren, wurde die Digitalzwilling-Funktion entwickelt. In der Digitalzwilling-Funktion werden deshalb die vor- und nachfolgenden Track Pieces erfasst und als ein Objekt (Abbildung 27) gespeichert. Falls das Fahrzeug ein Track-Piece verlässt und keine Positionsdaten vorhanden sind, wird die Visualisierung anhand des gespeicherten Objektes die realistischen Track-Pieces anzeigen.

Des weiteren wird die Funktion mit dem Objekt-Datentype optimiert, welcher die schnellste Lese-/Schreibgeschwindigkeit von Big O Notation [6] O (1) bringt. Der Array-Datentype und die Schleife wurden vermieden, weil die Lesegeschwindigkeit linear bis O(n) wachsen kann. Dies kann jedoch problematisch sein, wenn eine Funktion alle Pfade aller Track Pieces durchsuchen muss, welche über 1000 Schleifen und Parsen sein kann.

Abbildung 27: JavaScript Objekt als Text

5.1.3 Spurwechsel Funktion

Laut Angaben des Herstellers kann das Fahrzeug nach einem gewissen Befehl die Fahrspur wechseln. Trotzdem wurde dies beim Testen nie erreicht. Auf die konkrete Anfrage beim Entwickler des Fahrzeugkontrollers, wurde bis heute kein Fehler im Code gefunden.

Durch Zufall wurde die Lösung des Problems doch noch gefunden. Der Grund war, dass das Fahrzeug die Spur bei der empfohlenen Mindestgeschwindigkeit von 250 mm/s [7] nicht wechselt. Die Mindestgeschwindigkeit für den Spurwechsel liegt bei 300 mm/s.

6 Fazit & Ausblick

Die Hauptziele und Nebenziele der Bachelorthesis wurden erreicht. Die aufgetretenen Probleme bei den Neu-Entwicklungen waren sehr komplex und nur sehr zeitintensiv zu lösen. Eine Smartphone Applikation «FA9» zu entwickeln, konnte deshalb als einziges Ziel nicht erreicht werden. Dank der Hilfe von Herr Dr. Prof. Reto Koenig konnte das Unterziel «UZ1 Statischer Webserver für den Track Editor» ohne zusätzliche Software [3.9.2] einfach realisiert werden. Das ganze System benötigt somit nur zwei Applikationen: Fahrzeugkontroller und den MQTT-Broker.

Die Struktur der SVG ist sehr komplex. Für die Erzeugung einer Grafik benötigen man viel mathematisches Basiswissen (z.B. Vektorrechnungen und Bezier-Kurven [8]). Die Dateigrösse der SVG ist optimal für die Digitalisierung. Auch die Dateistruktur ist für die Software-Entwicklung geeignet. Der Digitalzwilling ist dank SVG fliessend und präziser geworden.

Die Probleme dieser Thesis kamen ausschliesslich von dem Anki Overdrive Fahrzeug, welches einer Black-Box gleicht. Die Logik des Systems im Fahrzeug, konnte man nicht lückenlos nachforschen. Die Gründe der aufgetauchten Fehler, können nur schwierig eruiert werden. Der Hersteller hat den Code für die Kommunikation mit dem BLE-Protokoll und dem SDK veröffentlicht. Das System des Fahrzeugs und wie die Sensoren funktionieren, bleiben jedoch geheim.

Zusammenfassend kann ich sagen, dass die Umsetzung dieser Thesis interessant, leidenschaftlich und spannend war. Von der skalierbaren Vektorgrafik hatte ich zuvor keine Kenntnisse. Nach diesem Projekt kann ich SVG in Zukunft problemlos bei der Digitalisierung einsetzen. Die Vorteile von SVG sind erstaunlich. Ich bin überzeugt, dass SVG in Zukunft in der Digitalisierung weitverbreitet sein wird.

Das Anki Overdrive Fahrzeug ist ein gutes Beispiel für die Entwicklung von Software-Lösungen im Berufsleben. Nicht immer erkennt man die Black-Box-Probleme gleich zu Beginn eines Projektes. Die Umsetzung erforderte mehr Aufwand als erwartet. Zusätzliche Konsultationen mit anderen Entwicklern, um bei der Umsetzung weiterzukommen, waren notwendig. Das minimale System bzw. die Hauptziele wurden angestrebt, um die Thesis aufrecht zu halten.

Ich habe mich ab dem Ergebnis sehr gefreut. Die Animation des Digitalzwillings macht das Frontend viel attraktiver.

Bei der Entwicklung des Frontend konnte ich meine Programmierungskenntnisse mit JavaScript enorm erweitern. Das Wissen meiner Studienvertiefung über dem MQTT-Protokoll und dem BLE hat wesentlich zum Erfolg dieser Thesis beigetragen. Ich würde gerne die erworbenen Kenntnisse in meiner weiteren beruflichen Zukunft anwenden.

Das neu entstandene System könnte natürlich noch weiterentwickelt werden, da das Fahrzeug einige Funktionen mehr zur Verfügung stellt. Es wäre spannend zu sehen, wenn das Fahrzeug vollautomatisch und autonom Fahren könnte (z.B. mit Kollisionsschutz).

Abschliessend möchte ich meinem Betreuer Herr Dr. Prof. Reto Koenig für die Unterstützung herzlich danken. Der Firma inroi ag gilt ebenfalls einen Dank, für das zur Verfügung stellen von diversen Infrastrukturen.

7 Abbildungsverzeichnis

8 Tabellenverzeichnis

9 Glossar

Begriff	Beschreibung
Standalone-System	Eigenständiger Computer, welcher nicht an ein anderes Computersystem angeschlossen ist.
PNG	Portable Network Graphics: ein Rastergrafikformat mit verlustfreier Datenkompression.
MQTT	Das MQTT-Protokoll bietet eine leichtgewichtige Methode zur Durchführung von Nachrichtenübermittlung unter Verwendung eines Publish/Subscribe-Modells.
MQTT-Broker	Ein MQTT-Broker ist eine Vermittler-Software, die MQTT-Clients die Kommunikation ermöglicht.
BLE	Die Bluetooth Low Energy ist eine energiesparende Variante der Funktechnikfamilie Bluetooth.
XML	Eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Textdaten.
GNU/Linux	GNU/Linux ist ein Unix-ähnliches Betriebssystem.
TCP	Transmission Control Protocol: Protokoll zur Kontrolle von Übertragungen.
Software Development Kit (SDK)	Eine Software, die bei der Entwicklung hilft. Die Softwareentwickler können die verfügbaren Dienste nutzen.
Framework	Eine Vorprogrammierung, welche die verschiedenen Funktionen zur Verfügung stellt.
AngularJS	Ein clientseitiges JavaScript-Webframework zur Erstellung von Webseiten.
Node.js	Ist eine plattformübergreifende JavaScript-Laufzeitumgebung.
Frontend	Schichteneinteilung bei der Softwareentwicklung: Nutzeroberfläche.
Backend	Schichteneinteilung bei der Softwareentwicklung: Teil eines Systems, dass sich mit der Datenverarbeitung im Hintergrund beschäftigt.
Digitalzwilling	Ein digitaler Zwilling ist eine digitale Repräsentanz eines materiellen oder immateriellen Objekts aus der realen Welt in der digitalen Welt.[9]
MVC-Pattern	Model View Controller (MVC): Ein Architekturmuster für die Softwareentwicklung.
IP-Adresse	Internetprotokoll-Adresse: Eine eindeutige Identifizierung und Adressierung eines Gerätes im Netzwerk.

10 Literaturverzeichnis

[1] Anki, Inc. anki/drive-sdk. Zugriff am: 12.01.2021. https://github.com/anki/drive-sdk	4
[2] Hofmann, Dominique Marc. Bachelor-Thesis: Anki Overdrive Track Editor. 16.06.2021	5,19
[3] SELFHTML-Wikis. Zugriff am: 12.06.2022. https://wiki.selfhtml.org/wiki/SVG/Attribute	17
[4] Cope, Steve. MQTT Client and Mosquitto Broker Message Restrictions With Examples Zugriff am: 12.06.2022. http://www.steves-internet-guide.com/mqtt-broker-message-restrictions/	20
[5] Patrick Schnabel. Zugriff am: 12.06.2022. https://www.elektronik-kompendium.de/sites/raspberry-pi/1912151.htm	24
[6] Wikipedia. Zugriff am: 12.06.2022. https://en.wikipedia.org/wiki/Big_O_notation	30
[7] Apache 2.0. Zugriff am: 12.06.2022. https://github.com/anki/drive-sdk/blob/410352617b8660338746a6bb2b2dd173afb1b91a/include/ankidrive/protocol.h#L118	30
[8] MDN contributors. Zugriff am: 12.06.2022. https://developer.mozilla.org/de/docs/Web/SVG/Tutorial/Paths	31
[9] Wikipedia. Zugriff am: 12.06.2022. https://de.wikipedia.org/wiki/Digitaler_Zwilling	33

11 Quellen

Abbildung 1: https://m.media-amazon.com/images/I/71Iaq5Aa7uL._SL1500_.jpg
Abbildung 2,3: Bachelor-Thesis: Anki Overdrive Track Editor, 16.06.2021

12 Erklärung der Diplomandinnen und Diplomanden

Selbständige Arbeit

Ich bestätige mit meiner Unterschrift, dass ich meine vorliegende Bachelor-Thesis selbständig durchgeführt habe. Alle Informationsquellen (Fachliteratur, Besprechungen mit Fachleuten, usw.) und anderen Hilfsmittel, die wesentlich zu meiner Arbeit beigetragen haben, sind in meinem Arbeitsbericht im Anhang vollständig aufgeführt. Sämtliche Inhalte, die nicht von mir stammen, sind mit dem genauen Hinweis auf ihre Quelle gekennzeichnet.

Name, Vorname	Müller, Mac
Datum	15 Juni 2022
Unterschrift	