

# Pflichtenheft

FennecBot

Autor: Team FennecBot  
Letzte Änderung: 19. Mai 2021  
Dateiname: Pflichtenheft\_FennecBot.docx

Version: 1.0

---

## ***Inhaltsverzeichnis***

<b>Versionshistorie</b>	<b>4</b>
<b>Vorhandene Dokumente</b>	<b>5</b>
<b>Überblick</b>	<b>5</b>
<b>Annahmen und Abgrenzungen</b>	<b>6</b>
<b>Workflow</b>	<b>6</b>
<b>Funktionalität</b>	<b>8</b>
Überblick der Funktionalität	8
Funktion 1: Die Umgebung scannen	8
Funktion 2: Einen Ball autonom suchen und erkennen	10
Funktion 3: Gegen den Ball fahren und schießen	12
Funktion 4: Hindernisse umfahren	13
Funktion 5: Automatisch abschalten	14
Funktion 6: Tore erkennen und auseinanderhalten	15
Funktion 7: Fernsteuerung	16
<b>Verwendete Hardware</b>	<b>17</b>

---

## ***Tabellenverzeichnis***

Tabelle 1: Relevante Unterlagen	5
Tabelle 2: Die Hauptziele, basierend auf dem Lastenheft	5
Tabelle 3: Fachliche und technische Projektannahmen	6
Tabelle 4: Projektabgrenzungen	6
Tabelle 5: Beschreibung der Funktion 1	9
Tabelle 6: Beschreibung der Funktion 2	10
Tabelle 7: Beschreibung der Funktion 3	12
Tabelle 8: Beschreibung der Funktion 4	13
Tabelle 9: Beschreibung der Funktion 5	14
Tabelle 10: Beschreibung der Funktion 6	15
Tabelle 11: Beschreibung der Funktion 7	16
Tabelle 12: Beschreibung der verwendeten Hardware	17

## ***Abbildungsverzeichnis***

Abbildung 1: Aktivitätsdiagramm des Fennec-Bot-Systems	7
Abbildung 2: Use-Case-Diagramm des Fennec-Bot-Systems	8
Abbildung 3: Aktivitätsdiagramm für die Funktion 2	11

---

### **Copyright**

© Team FennecBot

Die Weitergabe, Vervielfältigung oder anderweitige Nutzung dieses Dokumentes oder Teile davon ist unabhängig vom Zweck oder in welcher Form untersagt, es sei denn, die Rechteinhaber/In hat ihre ausdrückliche schriftliche Genehmigung erteilt.

### **Versionshistorie**

<i>Version:</i>	<i>Datum:</i>	<i>Verantwortlich</i>	<i>Änderung</i>
0.1	04.05.2021	Lukas Evers	Initiale Dokumenterstellung
0.2	10.05.2021	Son Khue Nguyen	Erweiterungen
0.3	11.05.2021	Umut Uzunoglu	Erweiterungen
0.4	12.05.2021	Hien Anh Nguyen Manh	Erweiterungen
0.5	16.05.2021	Lukas Evers, Son Khue Nguyen, Hien Anh Nguyen Manh, Umut Uzunoglu	Erweiterungen
0.8	18.05.2021	Lukas Evers, Son Khue Nguyen, Hien Anh Nguyen Manh, Umut Uzunoglu	Erweiterungen und Korrekturen
1.0	19.05.2021	Lukas Evers	Korrektur

## 1 Vorhandene Dokumente

Tabelle 1: Relevante Unterlagen

Dokument	Autor	Datum
Lastenheft	Team FennecBot	12.05.2021
Technische Spezifikation <a href="#">JetRacer AI Kit</a>	Waveshare	01.02.2021

## 2 Überblick

Für das Projekt soll auf Basis eines Waveshare JetRacers ein autonom fahrendes Auto erstellt werden. Die dazu zu erstellenden Teile bestehen aus einem 3D - gedruckten Chassis und der Software zur Verarbeitung der Sensordaten, um die Motoren des JetRacers entsprechend anzusteuern. Die Software wird auf einem Nvidia Jetson Nano ausgeführt. Die Software soll mittels Bilderkennung einen kleinen Fußball erkennen und dann den JetRacer autonom zum Ball fahren lassen um ihn zu schießen oder ihn zu dribbeln. Neben dem autonomen Modus kann man das Auto auch manuell mit einem Controller steuern.

Die Hauptziele des Projektes sind in der folgenden Tabelle dargestellt.

Tabelle 2: Die Hauptziele, basierend auf dem Lastenheft

#	Ziel	Beschreibung der Implementation
1	Erkennung eines Balles	ROS Package über Intel Realsense (Tiefenkamera) und LiDAR (Laserscanner)
2	Erkennung einer Linie auf dem Boden	ROS Package über Intel Realsense
3	Autonomes Verfolgen und Abfahren einer Linie	ROS Package für Erkennung mit der Intel Realsense und LiDAR, Abfahren über ROS Navigation
4	Autonomes Erkunden und Suche nach einem Ball falls nicht im Sichtfeld des Roboters	ROS Navigation mit Intel Realsense und LIDAR für SLAM-Funktionalität
5	Steuerung FennecBot mit einem Remote Controller	Ferngesteuert mit ROS Teleop
6	Autonomes Vermeiden von Hindernissen welches nicht Bälle sind	ROS Navigation mit Tiefenkamera und LIDAR
7	Erkennung eines Tores für den Ball	ROS Package über Intel Realsense
8	Beförderung des Balles in das richtige Tor	ROS Navigation, Intel Realsense, LIDAR
9	Variierung der Fahrgeschwindigkeit für Anpassung an verschiedene Situation (viele, wenig Hindernisse, Erkundung, Schießen, Dribbeln)	Phasenerkennung und Übermittlung der Fahrgeschwindigkeit an den ROS Controller (Steuerung über PWM)
10	Rückwärtsfahren und autonom rangieren	ROS Navigation

### **3 Annahmen und Abgrenzungen**

*Tabelle 3: Fachliche und technische Projektannahmen*

#	Annahmen
1	Verwendung von Open-Source-Code ist ohne Einschränkungen möglich
2	Verwendung eines Waveshare JetRacers
3	Einsatz von ROS
4	Bilderkennung über Intel Realsense

*Tabelle 4: Projektabgrenzungen*

#	Abgrenzungen
1	Genaue Implementierung der Funktionalität des eigentlichen "Fennecs" (z.B. Springen)
2	Alles, was über die Möglichkeiten des JetRacer hinausgeht

## 4 Workflow

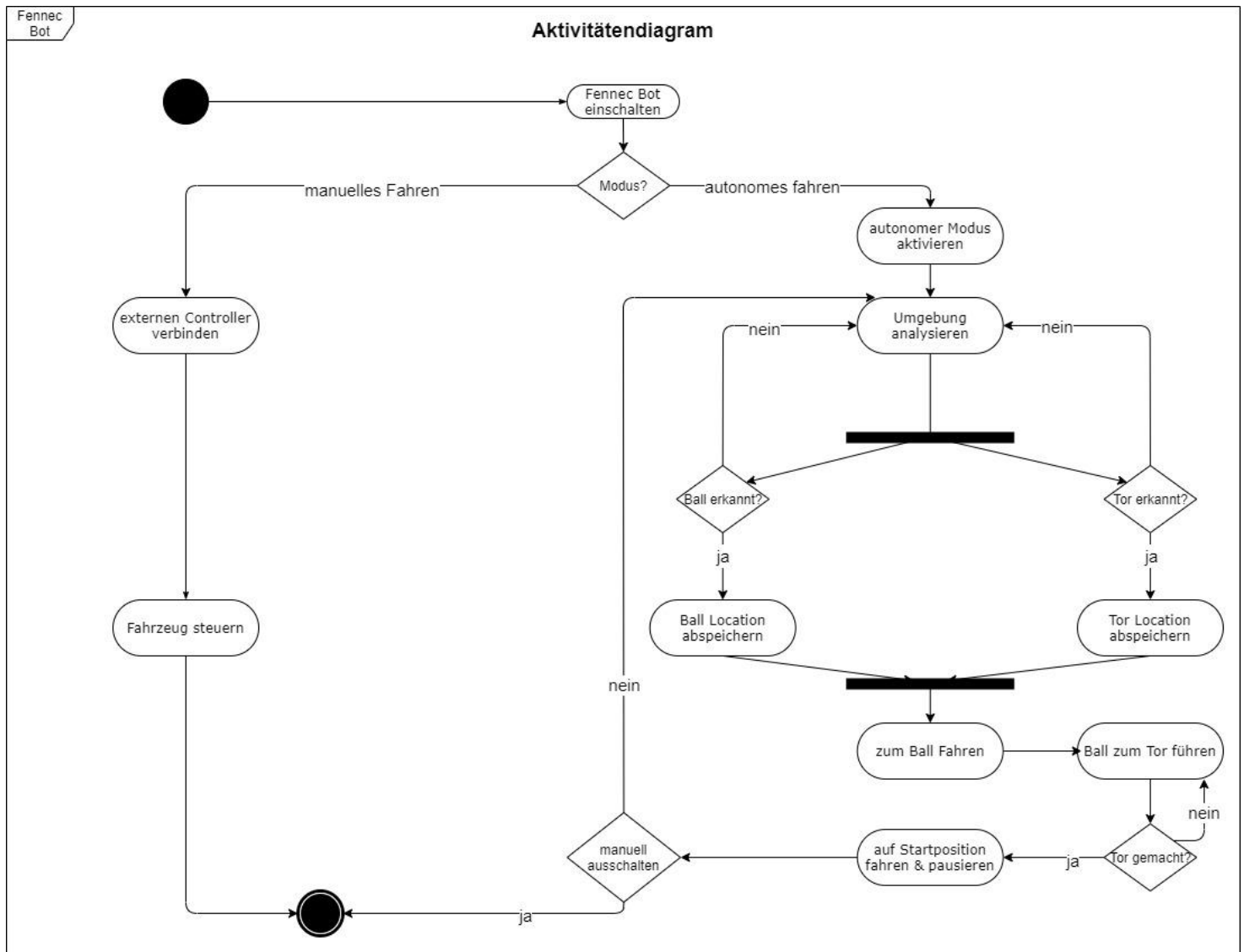


Abbildung 1: Aktivitätsdiagramm des Fennec-Bot-Systems

## 5 Funktionalität

### 5.1 Überblick der Funktionalität

Die Funktionalität wird durch das folgende Use-Case-Diagramm dargestellt, dessen Funktionen in den folgenden Tabellen detailliert beschrieben werden.

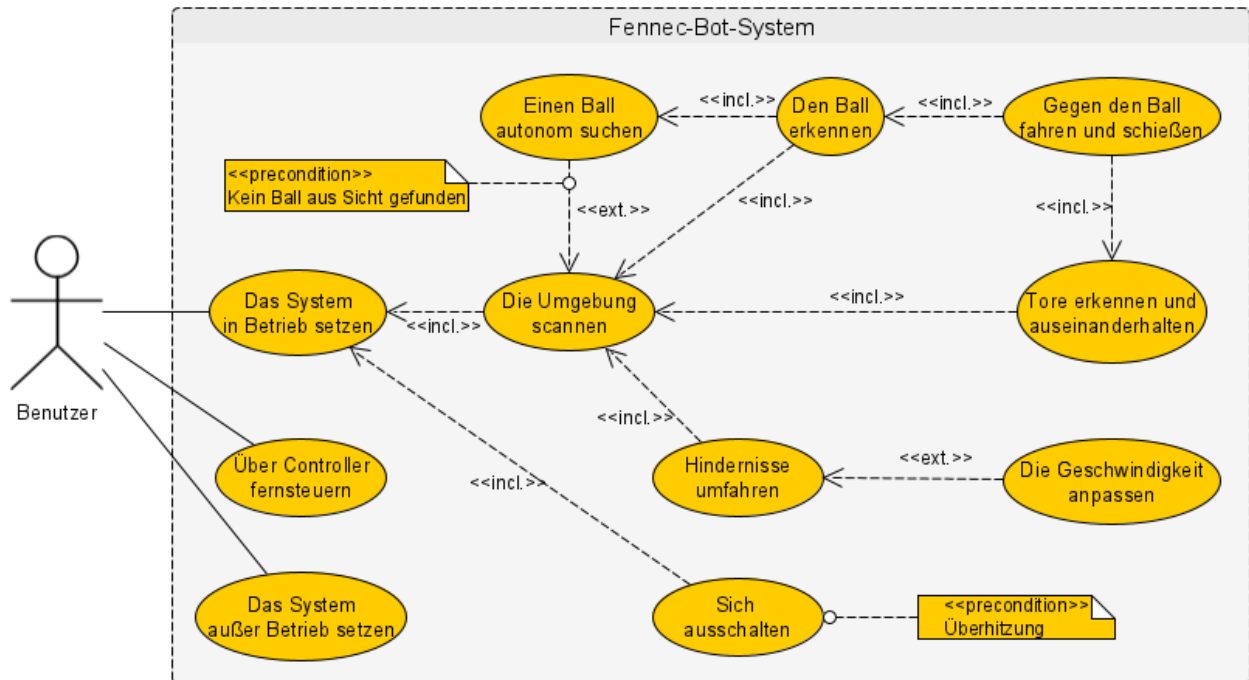


Abbildung 2: Use-Case-Diagramm des Fennec-Bot-Systems



## 5.2 Funktion 1: Die Umgebung scannen

*Tabelle 5: Beschreibung der Funktion 1*

Zweck/Ziel	Der Fennec kann mit der Intel Realsense und dem Laserscanner (LiDAR) eine Karte von der Umgebung erzeugen und autonom erkunden
Akteur/Auslöser	LiDAR/Intel Realsense
Vorbedingung	keine
Daten-Input	Tiefendaten/-bilder, Farbbilder, Laserscandaten
Verarbeitungsschritte	<ol style="list-style-type: none"> <li>1. Intel Realsense/LiDAR nehmen die Umgebung auf (Abstand und Position) und geben die Tiefendaten/Laserscandaten weiter an ROS</li> <li>2. In einen SLAM-Algorithmus (Simultaneous Localization and Mapping) werden Laserscan- und Odometriedaten (Bewegungsdaten) verarbeitet.</li> <li>3. Der SLAM-Algorithmus liefert eine stetig erweiterbare Karte und die aktuelle Schätzung der Position in dieser neu erstellten Karte basierend auf einem probabilistischen Algorithmus zur Posenbestimmung</li> </ol>
Ergebnis	Der Fennec besitzt eine genaue Repräsentation seiner Umgebung und kann basierend auf dieser Information autonom einen Pfad zu seinem Ziel abfahren.
Plausibilitäten	<p>Zu überprüfen:</p> <ul style="list-style-type: none"> <li>• Die Reichweite und die Aktualisierungsgeschwindigkeit der gescannten Karte, wenn sich der Fennec mit unterschiedlicher Tempo bewegt</li> <li>• Die Reaktionsgeschwindigkeit bei der Begegnung mit verschiedenen Objekten (mehr dazu in Plausibilitätsprüfungen Funktionen 2, 4, 6)</li> </ul>
Fehlerhandling	
Anforderung	Funktionale Anforderung 1
Test Cases	<ul style="list-style-type: none"> <li>• Erkennen und eingrenzen des Spielfeldes</li> <li>• Hindernisse, die die Fahrt beeinflussen umgehen</li> </ul>

## 5.3 Funktion 2: Einen Ball autonom suchen und erkennen

*Tabelle 6: Beschreibung der Funktion 2*

Zweck/Ziel	Der Fennec kann einen Ball autonom suchen und erkennen, macht somit sein Ziel und die Fahrtrichtung fest und fährt in Richtung Ball
Akteur/Auslöser	Intel Realsense/LiDAR, Motoren, Achse, ROS
Vorbedingung	Funktion 1 muss erfüllt sein
Daten-Input	Tiefendaten/-bilder, Farbbilder
Verarbeitungsschritte	<ol style="list-style-type: none"> <li>1. Der Ball, welcher einen Umfang von 64cm - 66cm wird mit der Intel RealSense Kamera aus zahlreichen Positionen fotografiert um eine robuste Erkennung des Spielballs zu ermöglichen.</li> <li>2. Während der Erkundung der Umgebung wird ständig mit der Farbkamera analysiert, ob ein Ball im Frame zu sehen ist. Abstand zum Ball bzw. die Position Balles auf der Karte wird gespeichert.</li> <li>3. Wurde der Ball nicht gefunden, soll der Fennec sich im Spielfeld/Raum bewegen und den Ball suchen.</li> </ol>
Ergebnis	Der Ball wurde gefunden und der Fennec fährt auf den Ball zu. Darauf folgt das Führen/Schießen des Balles
Plausibilitäten	<p>Zu überprüfen:</p> <ul style="list-style-type: none"> <li>• Die maximale und minimale Entfernung, in der der Ball erkannt werden kann</li> </ul>
Fehlerhandling	<ul style="list-style-type: none"> <li>• Ball wird nicht aus verschiedenen Perspektiven erkannt → mehr Bilder als Input für die Erkennung</li> <li>• Fennec bewegt sich zu schnell und erkennt den Ball nicht → Mehr FPS für die Bilderkennung (max. 60 FPS)</li> </ul> <p>FPS = Frames per Second</p>
Anforderung	Funktionale Anforderung 1.1, 1.4
Test Cases	<ul style="list-style-type: none"> <li>• Rollenden Ball erkennen</li> <li>• Stillen Ball erkennen</li> </ul>

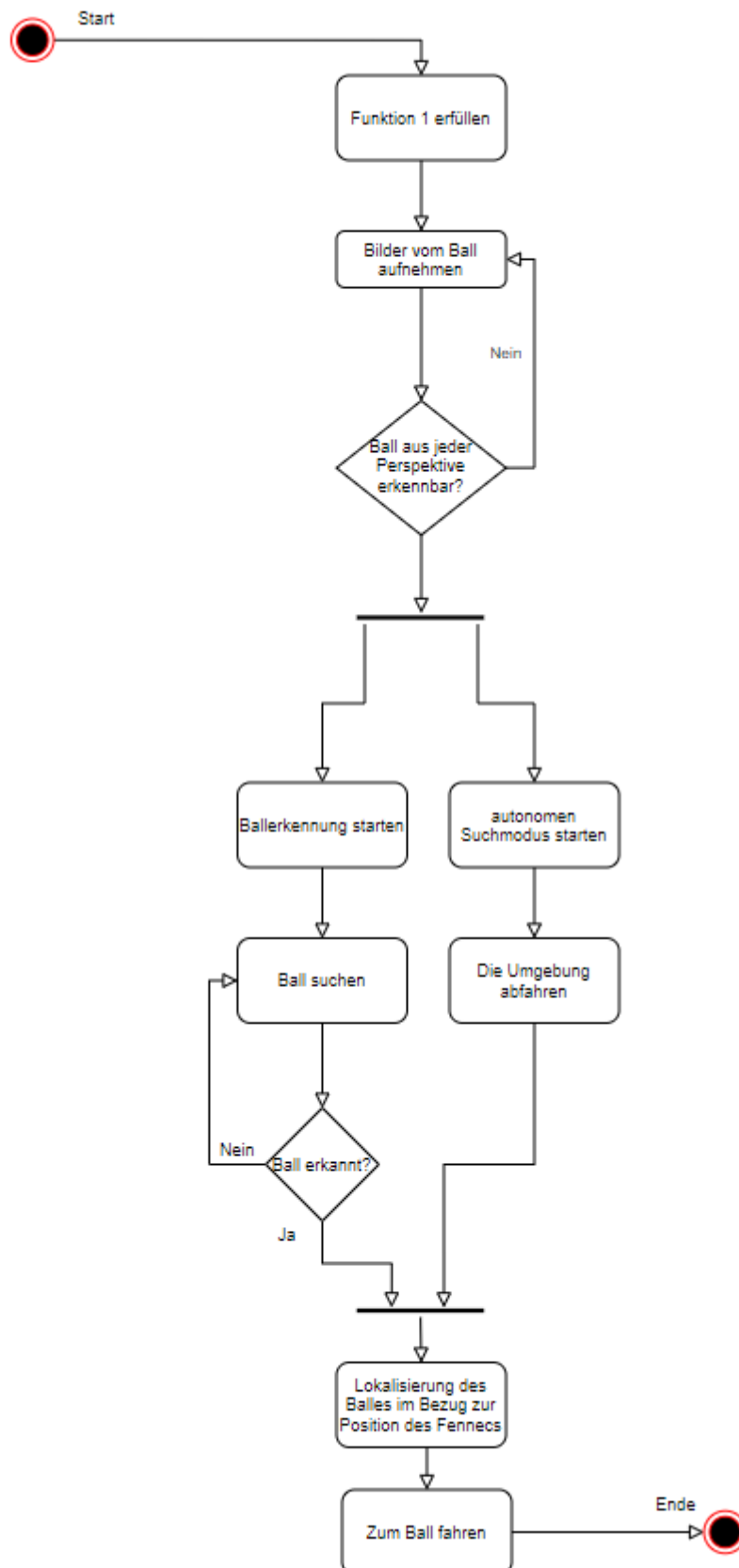


Abbildung 3: Aktivitätsdiagramm für die Funktion 2

## 5.4 Funktion 3: Gegen den Ball fahren und schießen

*Tabelle 7: Beschreibung der Funktion 3*

Zweck/Ziel	Der Fennec kann mit maximaler Geschwindigkeit gegen den Ball fahren und schießen, um einen Treffer zu erzielen/den Ball vorzulegen und hinterher zu fahren.
Akteur/Auslöser	Motoren, Achse, ROS
Vorbedingung	Funktion 2 und Funktion 6 müssen erfüllt sein und der Ball erkannt.
Daten-Input	Entfernung vom Ball zum Tor über Algebra
Verarbeitungsschritte	<ol style="list-style-type: none"> <li>1. Nach der Erkennung des Balles und eines Tores wird die Geschwindigkeit erhöht und zur Zielpose navigiert. Eine Pose ist in diesem Fall definiert als eine Position mit zugehöriger Orientierung im Raum. Die Zielpose wird errechnet aus der Position des Balles und der Position des Tores.</li> <li>2. Zum Zeitpunkt des Schießen werden die Motoren direkt angesprochen und nicht über ROS Navigation, da der Roboter sonst einem Hindernis autonom ausweichen würde.</li> <li>3. Je nach Entfernung zum Tor wird der Ball geführt oder möglichst hart geschossen. (Große Distanz -&gt; Ball führen, kurze Distanz -&gt; direkter Schuss)</li> <li>4. Um die Entfernung zu berechnen, identifiziert die Tiefenkamera beim Erkunden den Ball und das Tor, wobei beide Objekte als Punkte auf der erstellten Karte gespeichert werden.</li> </ol>
Ergebnis	Der Ball und das Tor wurden gefunden und der Fennec fährt auf den Ball zu. Darauf folgt das Führen/Schießen des Balles in Richtung des Tores
Plausibilitäten	<p>Zu überprüfen:</p> <ul style="list-style-type: none"> <li>• Die größte Kraft, die auf den Ball ausgeübt werden kann, ohne dass sich der Fennec selbst schadet</li> <li>• Die geringste Kraft, die auf den Ball ausgeübt werden kann, um ihn zu bewegen</li> </ul>
Fehlerhandling	Der Ball wird aufgrund des Geländes oder des Winkels der gerichteten Kraft in eine unerwartete Richtung bewegt → Falls Ball verloren → suchen, falls nicht → Zum Ball fahren und korrigieren
Anforderung	Funktionale Anforderung 1.2, 1.3
Test Cases	<ul style="list-style-type: none"> <li>• Gegen stillliegenden Ball fahren und schießen.</li> <li>• Gegen leicht rollenden Ball in einer gewissen Entfernung fahren und schießen.</li> </ul>

## 5.5 Funktion 4: Hindernisse umfahren

*Tabelle 8: Beschreibung der Funktion 4*

Zweck/Ziel	Der Fennec kann Hindernisse erkennen und seine Route entsprechend anpassen, um Kollisionen zu vermeiden
Akteur/Auslöser	Motoren, Achse, ROS Navigation, LiDAR, Intel RealSense
Vorbedingung	Funktion 1 muss erfüllt sein
Daten-Input	Tiefen-/Laserscandaten
Verarbeitungsschritte	<ol style="list-style-type: none"> <li>1. Der Fennec erhält permanent Tiefen-/Laserscandaten über die Intel Realsense/LiDAR, diese werden über ROS in Koordinaten im Raum transformiert → der Fennec weiß, wo ein Hindernis ist</li> <li>2. Falls der Fennec sich einem Hindernis nähert, ermittelt der Fennec über Tiefendaten der Intel RealSense die Entfernung zum Hindernis. Falls die Entfernung zu nah wird, wird eine optimierte Route berechnet und der Fennec dreht seine Achse entsprechend</li> <li>3. Der Fennec folgt der neuen Route und kehrt zur normalen Geschwindigkeit zurück</li> </ol>
Ergebnis	Der Fennec meidet unerwartete Hindernisse und erreicht sein Ziel auf einer kollisions freien Route.
Plausibilitäten	<p>Zu überprüfen:</p> <ul style="list-style-type: none"> <li>• Die benötigte Zeit für die Berechnung einer angepassten Route, die der Fennec braucht, um große Hindernisse zu umfahren</li> <li>• Die Größe und Position der toten Winkel bei der Bewegung mit dem Ball</li> </ul>
Fehlerhandling	<ul style="list-style-type: none"> <li>• Der Fennec steht vor einer Sackgasse → Bei der Berechnung der neuen Route muss der Wendekreis berücksichtigt werden (Joint Limits).</li> </ul>
Anforderung	Funktionale Anforderung 1.8, 1.9
Test Cases	<ul style="list-style-type: none"> <li>• Hindernisse unterschiedlicher Größe werden in unterschiedliche Richtungen und Winkel gesetzt.</li> <li>• Plötzlich auftauchte Hindernisse</li> </ul>

## 5.6 Funktion 5: Automatisch abschalten

*Tabelle 9: Beschreibung der Funktion 5*

Zweck/Ziel	Der Fennec kann langsamer werden und sich selbst abschalten, nachdem bestimmte Bedingungen erfüllt sind.
Akteur/Auslöser	Motoren, Temperatursensoren integriert in der CPU
Vorbedingung	Die Innentemperatur erreicht eine bestimmte Schwelle (ca. 90°C).
Daten-Input	Temperatur (in °C)
Verarbeitungsschritte	<ol style="list-style-type: none"><li>1. Die CPU wird durch lange Nutzung aufgeheizt, bis zu einer vorbestimmten Temperatur</li><li>2. Der Fennec wird langsamer</li><li>3. Der Fennec schaltet sich automatisch ab</li></ol>
Ergebnis	Der Fennec ist ausgeschaltet und die CPU muss vor dem erneuten Einschalten abgekühlt werden.
Plausibilitäten	Zu überprüfen: Die höchste Temperatur, die die CPU aushalten kann
Fehlerhandling	Mögliche Kollision durch Herunterfahren → Mehr Zeit zum Manövrieren vor dem Herunterfahren
Anforderung	Nicht-Funktionale Anforderung 3.2
Test Cases	Die Abgrenzung der CPU beim langen Fahrt testen

## 5.7 Funktion 6: Tore erkennen und auseinanderhalten

*Tabelle 10: Beschreibung der Funktion 6*

Zweck/Ziel	Der Fennec soll wissen, was er mit dem Ball machen soll, heißt er muss die Tore als Ziel definieren können, damit er den Ball auch ins Tor führen/schießen kann.
Akteur/Auslöser	Intel Realsense (Tiefen- und Farbkamera)/ LiDAR (Laserscanner)
Vorbedingung	Funktion 1 muss erfüllt sein
Daten-Input	Farbbilder, Tiefendaten
Verarbeitungsschritte	<ol style="list-style-type: none"> <li>1. Über Laserscandaten von der Funktion 1 können die Tore als Ziel festgelegt werden. Da die Laserscandaten gespeichert werden und sich die Tore nicht bewegen werden, muss die Erfassung der Tore nicht in Echtzeit erfolgen.</li> <li>2. Durch die Kartierung der Umgebung sind die Abmessungen des Spielfeldes bekannt und ein Platzieren der Tore in dieser Karte ist autonom möglich.</li> <li>3. Außerdem wird über die Farbkamera der Intel Realsense das Tor erkannt. Dafür werden die Tore mit Muster oder QR-Codes markiert. Dies vereinfacht die Erkennung von Features bzw. Unterscheidung Dieser. Dies macht das Unterscheiden der Tore erheblich einfacher.</li> </ol> <p>Features sind Eigenschaften, die die Bilderkennung erleichtert. Beispielsweise Kanten/Ecken.</p>
Ergebnis	Der Fennec kann mit den gegebenen Informationen den Ball ins Tor führen/schießen
Plausibilitäten	<p>Zu überprüfen:</p> <ul style="list-style-type: none"> <li>• Die Entfernung, aus der der Fennec ein Tor unterscheiden kann</li> <li>• Die mögliche Art der Kennzeichnung des Tores (z.B. farbige Klebestreifen als Muster oder durch QR-Codes)</li> </ul>
Fehlerhandling	<ul style="list-style-type: none"> <li>• Das Tor ist blockiert → Der Fennec muss in der Lage sein, in einen anderen Winkel umzuleiten</li> <li>• Die Kennzeichnung auf dem Tor wird getauscht → die Positionen der Toren im Speicher müssen flexibel austauschbar sein</li> </ul>
Anforderung	Funktionale Anforderung 1.7
Test Cases	<ul style="list-style-type: none"> <li>• Fennec vor dem Tor mit/ohne Ball platzieren und das Tor erkennen lassen</li> <li>• Fennec während Bewegung mit/ohne Ball das Tor erkennen lassen</li> <li>• Aus verschiedenen Perspektiven das Tor erkennen lassen</li> </ul>

## 5.8 Funktion 7: Fernsteuerung

*Tabelle 11: Beschreibung der Funktion 7*


Zweck/Ziel	Der Benutzer kann einen Controller verwenden, um den Fennec fernzusteuern
Akteur/Auslöser	Bluetooth-Empfänger (Wireless Controller), Motoren, Achsen
Vorbedingung	Fernbedienungsmodus wurde aktiviert
Daten-Input	Controller-Input
Verarbeitungsschritte	<ol style="list-style-type: none"><li>1. Der Fennec empfängt über Bluetooth die Benutzereingaben auf dem Controller</li><li>2. Der Fennec bewegt sich entsprechend</li></ol>
Ergebnis	Der Benutzer steuert den Fennec nach Belieben
Plausibilitäten	Das sofortige Umschalten des Modus wird überprüft (z.B während der Hochgeschwindigkeitsfahrt)
Fehlerhandling	keine
Anforderung	Funktionale Anforderung 1.5, Nicht-Funktionale Anforderung 2.3
Test Cases	Die Konnektivität in unterschiedlicher Entfernung testen



## 6 Verwendete Hardware

Tabelle 12: Beschreibung der verwendeten Hardware

Name	Funktionen/Schnittstellen	Bild
Waveshare Jetson Ranger	Stromversorgung, Motorsteuerung, Lenkung	
Intel RealSense D415	RGB-3D Tiefenkamera, USB-Schnittstelle	
Nvidia Jetson Nano	Leistungsstarker Mini-Computer für KI und Robotik, HDMI-, Ethernet-, USB-Schnittstellen, GPIO-Ports, I <sup>2</sup> C, SPI, UART	
RPLiDAR A1	2D-Laserscan im Bereich von 30cm - 6m, USB-Schnittstelle	
Intel Model 8265NGW	Wifi Communication Chip	

Ultimaker 2+	3D-Drucker zur Chassis-Erstellung	
--------------	--------------------------------------	---