

# Qualitätssicherung

## Fennec Racer

Pronzipiell sehr gut. Kleine Anmerkungen siehe unten!

Wertung QS : 5 Punkte

Autor: Team Fennec Bot  
Letzte Änderung: 16.06.2021  
Dateiname: Qualitätssicherung\_Fennec\_Bot.docx

Version: 1.0

---

## ***Inhaltsverzeichnis***

<b>1 Testplan</b>	<b>4</b>
<b>2 Testfälle</b>	<b>7</b>
Einbindung RPLIDAR mit ROS	7
Einbindung Intel Real Sense mit ROS	8
Verwendung eines Hardware Controllers für Ackerman Steuerung	9
Fernsteuerung mit einem Gamepad	10
Autonome Navigation & Umgebungserkundung	11
<b>3 Testprotokoll</b>	<b>12</b>
<b>4 Anhang</b>	<b>13</b>
Fehlerkategorien	13
Q-Kriterien ISO 9126	13
Q-Kriterien für Dokumente	15

### **Copyright**

© Team Fennec Bot

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.

### **Version Historie**

Version:	Datum:	Verantwortlich	Änderung
1.0	16.06.21	Lukas Evers	

### **Vorhandene Dokumente**

Alle für die vorliegende Spezifikation ergänzenden Unterlagen müssen hier aufgeführt werden

Dokument	Autor	Datum
Lastenheft_Fennec_Bot.pdf	Team Fennec Bot	02.05.21
Pflichtenheft_FennecBot.docx.pdf	Team Fennec Bot	19.05.21
Technische_Spezifikation_Fennec_Bot.pdf	Team Fennec Bot	16.06.21

## 1 Testplan

Test-Objekt	Qualitätskriterien	QS-Teststufe 1 "Source Code, Komponente, Funktion"			Bemerkungen
		Test-Verfahren	Zyklus	Zuständig	
Dokumentation					
Source code	Verständlichkeit Lesbarkeit, Funktionale Vollständigkeit und Korrektheit	Editorial Review Technisches Review Gegenlesen	nach jeder Änderung, Meilenstein, am Ende	Teammitglied, Anwender	Der Source-Code muss verständlich und strukturiert sein.
Source code-Dokumentation	Verständlichkeit Lesbarkeit, Funktionale Vollständigkeit und Korrektheit t	Editorial Review Technisches Review Gegenlesen	nach jeder Änderung, Meilenstein, am Ende	Teammitglied, Anwender	Die Dokumentation des Source-Codes muss ausführlich durchgeführt sein.
...					
Applikation					
Funktionalitäten					
Umgebung scannen	Richtigkeit, Zuverlässigkeit	Funktionstest, Datentest, Lasttest	am Ende	Teammitglied	Es wird eine OGM (Occupany Grid Map) erstellt
Ball autonom suchen und erkennen	Richtigkeit, Robustheit, Zuverlässigkeit	Funktionstest, Datentest, Performanztest	am Ende	Teammitglied	Der Ball ist ungefähr fußballgroß.

Automatisch abschalten	Zuverlässigkeit	Lasttest	nach jeder Änderung	Teammitglied	Bei Überhitzung muss der Schutz der Hardware gewährleistet sein
Tore erkennen und auseinanderhalten	Richtigkeit, Zuverlässigkeit, Funktionalität	Funktionstest, Datentest	am Ende	Teammitglied	Es soll nur auf das richtige Tor geschossen werden.
Fernsteuerung	Ergonomie, Zuverlässigkeit, Benutzbarkeit	Funktionstest, Ergonomietest	am Ende	Kunde, Teammitglied	Bedienung soll leicht und intuitiv sein
<b>nicht funktionale Eigenschaften / Anforderungen</b>					
Hardware ist durch Chassis vor Schäden geschützt	Robustheit	Stresstest	nach jeder Änderung	Teammitglied, Anwender	
Inbetriebnahme des Roboters ist leicht und schnell	Benutzbarkeit	Lasttest	nach jeder Änderung	Teammitglied, Anwender	Unter zwei Minuten soll der Roboter startbar sein.
Wartbarkeit des Roboters	Wartung	Ergonomietest, Funktionsst	nach jeder Änderung	Teammitglied	Ladebuchse muss zugänglich bleiben.

Test-Objekt	Qualitätskriterien	QS-Teststufe 2 "Integration / Systemtest"			Bemerkung?
		Test-Verfahren	Zyklus	Zuständig	
Funktionalitäten					
Gegen Ball fahren und schießen	Funktionalität	Funktionstest, Zuverlässigkeit, Robustheit	am Ende	Teammitglied	Bilderkennung, Lokalisierung, Navigation und Hardware Controller spielen hier zusammen
Hindernisse umfahren	Richtigkeit, Zuverlässigkeit, Effizienz	Funktionstest, Performanztest	am Ende	Teammitglied	Navigation und die Integration des Ackerman Controllers sind hier zu testen, insbesondere auch die Recovery Behaviours bei Fehlern der Sensorik/Navigation
Sensorfusion	Funktionalität, Zuverlässigkeit	Lasttest, Funktionstest	am Ende	Teammitglied	Odometrie, IMU, Laserscan und vSlam können gemeinsam zur Verbesserung der Karten und der Navigation dienen
Sensorik einbinden	Funktionalität	Funktionstest	Meilenstein	Teammitglied	Integration von Komponenten in ROS



## 2 Testfälle

### 2.1 Einbindung RPLIDAR mit ROS

Pro Testfall soll das folgende Template angewandt werden:

Testfall	Beschreibung
Testfall-Nummer	01
Teststart	Funktionstest
Zu testender Geschäftsprozess/ Zu testende Funktionsgruppe	Laserscanner
Testziel	Man kann den Laserscanner mit ROS starten und die Sensorinformationen empfangen und zur Überprüfung anzeigen lassen
Testvoraussetzungen	<ul style="list-style-type: none"><li>• Vollständige ROS Installation auf Ubuntu</li><li>• Catkin Workspace erstellt</li><li>• Verbindung mit USB Schnittstelle hergestellt</li><li>• RPLiDAR ROS Package installiert</li><li>• Workspace und ROS Installation gesourced</li></ul>
Testfalldaten	<ul style="list-style-type: none"><li>• Starten des Lasers mit <code>roslaunch rplidar_ros view_rplidar.launch</code></li></ul>
Erwartetes Verhalten	<ul style="list-style-type: none"><li>• In Rviz sollte man rote Punkte im Raum sehen welche die einzelnen Entfernungsmessungen des Lasers darstellen.</li></ul>



#### Testergebnis

Folgendes Template soll das Testergebnis jedes einzelnen Testfalls dokumentieren:

Testergebnis	<input checked="" type="checkbox"/> Bestanden <input type="checkbox"/> Nicht Bestanden	
Fehlerkategorie	<input type="checkbox"/> Leicht <input type="checkbox"/> Mittel <input type="checkbox"/> Schwer <sup>1</sup>	
Bemerkung	Der Laserscan kann mit dem Launchfile problemlos gestartet werden.	
Tester Kunde	Tester Auftragnehmer	Datum
Umut Uzunoglu	Umut Uzunoglu	09.06.21

<sup>1</sup> Die Beschreibung der Fehlerkategorien entnehmen Sie bitte dem beigegeführten Anhang

## 2.2 Einbindung Intel Real Sense mit ROS


Testfall	Beschreibung
Testfall-Nummer	02
Teststart	Funktionstest
Zu testender Geschäftsprozess/ Zu testende Funktionsgruppe	Intel RealSense Kamera
Testziel	Die Kamera kann mit ROS gestartet werden und Kamerabilder sowie die Sensorinformationen werden empfangen und zur Überprüfung werden diese visuell dargestellt
Testvoraussetzungen	<ul style="list-style-type: none"> <li>• Vollständige ROS Installation auf Ubuntu</li> <li>• Catkin Workspace erstellt</li> <li>• Verbindung mit USB Schnittstelle hergestellt</li> <li>• RealSense-ROS Package installiert</li> <li>• Workspace und ROS Installation gesourced</li> </ul>
Testfalldaten	• Starten des Lasers mit "roslaunch realsense2_camera rs_camera.launch"
Erwartetes Verhalten	<ul style="list-style-type: none"> <li>• In Rviz kann man das Live-Bild der Kamera sehen</li> <li>• In Rviz kann man die Tiefendaten der Kamera anzeigen lassen</li> </ul>

Testergebnis	<input checked="" type="checkbox"/> Bestanden <input type="checkbox"/> Nicht Bestanden	
Fehlerkategorie	<input type="checkbox"/> Leicht <input type="checkbox"/> Mittel <input type="checkbox"/> Schwerwiegend	
Bemerkung		
Tester Kunde Umut Uzunoglu	Tester Auftragnehmer Umut Uzunoglu	Datum 09.06.21





## 2.3 Verwendung eines Hardware Controllers für Ackerman Steuerung

Testfall	Beschreibung
Testfall-Nummer	<ul style="list-style-type: none"> <li>03</li> </ul>
Testart	<ul style="list-style-type: none"> <li>Integrationstest</li> </ul>
Zu testender Geschäftsprozess/ Zu testende Funktionsgruppe	<ul style="list-style-type: none"> <li>Jetracer</li> </ul>
Testziel	<ul style="list-style-type: none"> <li>Es ist möglich Twist Message in ROS zu publishen und diese werden dann vom Hardware Controller in die einzelnen Bestandteile zerlegt um die Lenk- und Antriebsachse entsprechend anzusteuern</li> </ul>
Testvoraussetzungen	<ul style="list-style-type: none"> <li>Vollständige ROS Installation auf Ubuntu</li> <li>Catkin Workspace erstellt</li> <li>Verbindung mit USB Schnittstelle hergestellt</li> <li>RealSense-ROS Package installiert</li> <li>Workspace und ROS Installation gesourced</li> <li>ROS Package Abhängigkeit geometry_msgs, std_msgs, rospy</li> </ul>
Testfalldaten	<p>cmd_vel Daten:</p> <p>cmd_vel.linear.x = 1.0 cmd_vel.linear.y = 0.0 cmd_vel.angular.z = 1.0</p> <p>cmd_vel.linear.x = 2.0 cmd_vel.linear.y = 0.0 cmd_vel.angular.z = -1</p> 
Erwartetes Verhalten	Es wird ein geschlossener Kreis gefahren. Zunächst vorwärts im Uhrzeigersinn, im Anschluss rückwärts gegen den Uhrzeigersinn


Testergebnis	<input checked="" type="checkbox"/> Bestanden <input type="checkbox"/> Nicht Bestanden		
Fehlerkategorie	<input type="checkbox"/> Leicht <input type="checkbox"/> Mittel <input type="checkbox"/> Schwerwiegend		
Bemerkung			
Tester Kunde Lukas Evers	Tester Auftragnehmer Lukas Evers	Datum 16.06.21	


## 2.4 Fernsteuerung mit einem Gamepad

Testfall	Beschreibung
Testfall-Nummer	• 04
Teststart	• Funktionstest
Zu testender Geschäftsprozess/ Zu testende Funktionsgruppe	• Fernsteuerung ( Teleoperation)
Testziel	• Der Jetracer lässt sich leicht und zuverlässig mit einem Gamepad installieren
Testvoraussetzungen	<ul style="list-style-type: none"> <li>• Vollständige ROS Installation auf Ubuntu</li> <li>• Catkin Workspace erstellt</li> <li>• Verbindung mit Bluetooth oder 2,4GHz Empfänger ist hergestellt</li> <li>• Joy ROS Package installiert</li> <li>• Workspace und ROS Installation gesourced</li> </ul>
Testfalldaten	<ul style="list-style-type: none"> <li>• Joy-Message: joy.axes[1] = 0.5 joy.axes[4] = 1.0 joy.buttons[4] = 1</li> </ul>
Erwartetes Verhalten	• Der Jetracer fährt einen geschlossenen Kreis

Testergebnis	<input checked="" type="checkbox"/> Bestanden <input type="checkbox"/> Nicht Bestanden		
Fehlerkategorie	<input type="checkbox"/> Leicht <input type="checkbox"/> Mittel <input type="checkbox"/> Schwerwiegend		
Bemerkung			
Tester Kunde Lukas Evers	Tester Auftragnehmer Lukas Evers	Datum 16.06.21	

## 2.5 Autonome Navigation & Umgebungserkundung

Testfall	Beschreibung
Testfall-Nummer	<ul style="list-style-type: none"> <li>05</li> </ul>
Teststart	<ul style="list-style-type: none"> <li>Funktionstest</li> </ul>
Zu testender Geschäftsprozess/ Zu testende Funktionsgruppe	<ul style="list-style-type: none"> <li>ROS Navigation</li> </ul>
Testziel	<ul style="list-style-type: none"> <li>Der Fennec kann selbstständig (ohne Fernsteuerung) Ziele im Raum erreichen. Außerdem kann eine Karte der Umgebung erzeugt werden die zur Pfadplanung dient. Der Roboter ist in der Lage sich auf dieser Karte zu lokalisieren.</li> </ul>
Testvoraussetzungen	<ul style="list-style-type: none"> <li>Vollständige ROS Installation auf Ubuntu</li> <li>Catkin Workspace erstellt</li> <li>ROS Navigation Package installiert</li> <li>AMCL installiert</li> <li>Gmapping installiert</li> <li>Workspace und ROS Installation gesourced</li> <li>Alle benötigten Sensoren für Gmapping fahren auf dem Roboter mit (mind. ein Laserscanner)</li> <li>Die Koordinatentransformationen aller Komponenten (Räder, Gelenke, Sensorik) sind vorhanden und werden gepublished</li> <li>Es werden Odometrie - Daten (Bewegungsdaten) vom Roboter gemessen (alternativ: kalkuliert, approximiert) und gepublished</li> <li>Der Roboter verfügt über einen Hardware Controller welche Twist-Messages entgegennehmen kann und in Bewegung umsetzt</li> </ul>
Testfalldaten	<ul style="list-style-type: none"> <li></li> </ul>
Erwartetes Verhalten	<ul style="list-style-type: none"> <li>In Rviz wird nach und nach eine Karte erstellt und der Roboter plant einen Pfad zu jedem Ziel, welches vorgegeben wird</li> </ul>

Testergebnis	<input type="checkbox"/> Bestanden <input checked="" type="checkbox"/> Nicht Bestanden	
Fehlerkategorie	<input type="checkbox"/> Leicht <input type="checkbox"/> Mittel <input checked="" type="checkbox"/> Schwerwiegend	
Bemerkung	Es ist zurzeit nicht möglich die Sensorik entsprechend der Anforderung für ROS Navigation am Roboter zu montieren. Die Kartierung und Navigation funktioniert in einer Gazebo Simulation jedoch bereits, sodass die Portierung von Simulation in die echte Welt der nächste Schritt ist. 	
Tester Kunde Lukas Evers	Tester Auftragnehmer Lukas Evers	Datum 16.06.21

### 3 Testprotokoll

TestfallNr.	Datum	Status	Schweregrad	Datum 2. Lauf	Status 2. Lauf
01	09.06.21	bestanden			
02	09.06.21	bestanden			
03	16.06.21	bestanden			
04	16.06.21	bestanden			
05	16.06.21	nicht bestanden	schwer	30.06.21	nicht begonnen

## 4 Anhang

### 4.1 Fehlerkategorien

Für die Abnahme des Systems sind folgende Fehlerklassen definiert:

- **3 = Schwerer Mangel**

Produktivsetzung nicht möglich (Nachhaltige Störung des Softwareablaufes mit daraus resultierender Funktionsuntüchtigkeit des Systems bzw. Störung von Systemteilen, die zur Störung aller Arbeitsabläufe beim Auftraggeber führt.)
  
- **2 = Mittlerer Mangel**

Produktivsetzung möglich aber mangelhafte Funktionen nicht nutzbar (Durch eine Störung treten in Teilen der Programmabläufe nicht unerhebliche Störungen auf, so dass Teile der Software nicht verwendbar sind.)
  
- **1 = Leichter Mangel**

Produktivsetzung durch Workaround mit vertretbarem Zusatzaufwand möglich (Alle anderen als die in den vorstehenden Prioritätsgraden beschriebenen Störungsbilder)

### 1.1 Q-Kriterien ISO 9126

Gruppe	Q-Kriterium	
Funktionalität		
Sind alle im Pflichtenheft aufgeführten Kriterien vorhanden und ausführbar?	Angemessenheit	Merkmale von Software, die sich auf das Vorhandensein und die Eignung einer Menge von Funktionen für spezifizierte Aufgaben beziehen.
	Richtigkeit	Merkmale von Software, die sich beziehen auf das Liefern der richtigen oder vereinbarten Ergebnisse oder Wirkungen.
	Interoperabilität	Merkmale von Software, die sich auf ihre Eignung beziehen, mit vorgegebenen Systeme zusammenzuwirken.
	Ordnungsmäßigkeit	Merkmale von Software, die bewirken, dass die Software anwendungsspezifische Normen oder Vereinbarungen oder gesetzliche Bestimmungen oder ähnliche Vorschriften erfüllt.
	Sicherheit	Merkmale von Software, die sich auf ihre Eignung beziehen, unberechtigten Zugriff, sowohl versehentlich als auch vorsätzlich, auf Programme und Daten zu verhindern.
Zuverlässigkeit		
Zu welchem Grad erfüllt die Software dauerhaft und korrekt die geforderten Funktionen?	Reife	Merkmale von Software, die sich auf die Häufigkeit von Versagen durch Fehlzustände in der Software beziehen.
	Fehlertoleranz	Merkmale von Software, die sich auf ihre Eignung beziehen, ein spezifiziertes Leistungsniveau bei Software-Fehlern oder Nicht-Einhaltung ihrer spezifizierten Schnittstelle zu bewahren.
	Wiederherstellbarkeit	Merkmale von Software, die sich beziehen auf die Möglichkeit, bei einem Versagen ihr Leistungsniveau wiederherzustellen und die direkt betroffenen Daten wiederzugewinnen, und auf die dafür benötigte Zeit und den benötigten Aufwand.
Benutzbarkeit		
Wie schnell kann man den Umgang mit der Software lernen und wie leicht ist sie zu bedienen?	Verständlichkeit	Merkmale von Software, die sich auf den Aufwand für den Benutzer beziehen , das Konzept und die Anwendung zu verstehen.

	<b>Erlernbarkeit</b>	Merkmale von Software, die sich auf den Aufwand für den Benutzer beziehen, ihre Anwendung zu erlernen. (z.B. Ablaufsteuerung, Eingabe, Ausgabe)
	<b>Bedienbarkeit</b>	Merkmale von Software, die sich auf den Aufwand für den Benutzer bei der Bedienung und Ablaufsteuerung beziehen.
<b>Effizienz</b>		
Wie sind zeitliches Verhalten und Ressourcenverbrauch bei gegebenen System-voraussetzungen?	<b>Zeitverhalten</b>	Merkmale von Software, die sich beziehen auf die Antwort- und Verarbeitungszeiten und auf den Durchsatz bei der Ausführung ihrer Funktionen.
	<b>Verbrauchsverhalten</b>	Merkmale von Software, die sich darauf beziehen, wie viele Betriebsmittel bei der Erfüllung ihrer Funktionen benötigt werden und wie lange.
<b>Änderbarkeit</b>		
Mit welchem Zeit- und Arbeitsaufwand lassen sich Änderungen sowie Fehlererkennung und -behebung durchführen?	<b>Analysierbarkeit</b>	Merkmale von Software, die sich auf den Aufwand beziehen, der notwendig ist, um Mängel oder Ursachen von Versagen zu diagnostizieren oder um änderungsbedürftige Teile zu bestimmen.
	<b>Modifizierbarkeit</b>	Merkmale von Software, die sich auf den Aufwand beziehen, der zur Ausführung von Verbesserungen, zur Fehlerbeseitigung oder zur Anpassung an Umgebungsänderungen notwendig ist.
	<b>Stabilität</b>	Merkmale von Software, die sich auf das Risiko unerwarteter Wirkungen von Änderungen beziehen.
	<b>Prüfbarkeit</b>	Merkmale von Software, die sich auf den Aufwand beziehen, der zur Prüfung der geänderten Software notwendig ist.
<b>Übertragbarkeit</b>		
Mit welchem Aufwand lässt sich die Software an geänderte/ verbesserte Systembedingungen anpassen bzw. in neuen Systemen einsetzen?	<b>Anpassbarkeit</b>	Merkmale von Software, die sich auf die Möglichkeit beziehen, sie an verschiedene festgelegte Umgebungen anzupassen, wenn nur Schritte unternommen oder Mittel eingesetzt werden, die für diesen Zweck für die betrachtete Software vorgesehen sind.
	<b>Installierbarkeit</b>	Merkmale von Software, die sich auf den Aufwand beziehen, der zur Installation der Software in einer festgelegten Umgebung notwendig ist.
	<b>Konformität</b>	Merkmale von Software, die bewirken, dass die Software Normen oder Vereinbarungen zur Übertragbarkeit erfüllt.
	<b>Austauschbarkeit</b>	Merkmale von Software, die sich beziehen auf die Möglichkeit, diese anstelle einer anderen Software in der Umgebung jener Software zu verwenden und auf den dafür notwendigen Aufwand.

## 1.2 Q-Kriterien für Dokumente

Für die Erreichung des Projektzieles, das Produkt „Dokument“ zu erzeugen, dass den fachlichen und technischen Anforderungen des Auftraggebers entspricht, ergeben sich z.B. die folgenden Qualitätsmerkmale:

Merkmal	Erläuterung	Mindest-anfordrg	Prüfmöglichkeit
<b>Eindeutigkeit</b>	Eignung von Dokumenten zur unmissverständlichen Vermittlung von Informationen für jeden Leser		Keine offenen Fragen zu den einzelnen Abschnitten (Prüfung durch Gruppeninspektion und Diskussion)
<b>Lesbarkeit</b>	Eignung von Dokumenten zur Entnahme der darin enthaltenen Informationen		Prüfung durch Einsatz eines unbedarften Testlesers, Vorhandensein eines Glossars, Erläuterung von Fachbegriffen
<b>Verständlichkeit</b>	Eignung von Dokumenten zur erfolgreichen Vermittlung der darin enthaltenen Informationen an einen sachkundigen Leser		Vorhandensein eines Glossars, Integration von Illustrationen, Diagrammen
<b>Detaillierungsgrad</b>	Vorhandensein der ausreichenden Beschreibung der fachlichen und technischen Einzelheiten im Dokument		Beschreibung der Sonder- und Ausnahmefälle, gleiche Behandlung (gleiche Detaillierung) aller Textabschnitte
<b>Funktionale Vollständigkeit</b>	Vorhandensein der für den Zweck der Dokumentation notwendigen und hinreichenden Information		Einsatz des <KUNDE>Templates gewährleistet die Vollständigkeit an notwendigen Informationen, Beschreibung der Sonder- und Ausnahmefälle
<b>Fehlerfreiheit</b>	Nichtvorhandensein von sprachlichen Fehlern, die die Informationsaufnahme beeinträchtigen		Rechtschreib- und Grammatikprüfung
<b>Widerspruchsfreiheit</b>	Nichtvorhandensein von einander entgegengesetzten Aussagen im Dokument		Unnötige Redundanzen sollen vermieden werden, Dokument soll in sich konsistent sein
<b>Aktualität</b>	Übereinstimmung der Beschreibung der Situation in Dokument und Wirklichkeit		Gespräche mit dem Auftraggeber (Kundeninspektion, Workshops)
<b>Funktionale Korrektheit</b>	Nichtvorhandensein von funktionalen Fehlern, die den fachlichen und technischen Inhalt betreffen		Wiedergabe der Anforderungen aus dem Vorgängerdokument
<b>Normenkonformität</b>	Erfüllung der für die Erstellung von Dokumenten geltenden Vorschriften und Normen		Einsatz des <KUNDE>Templates gewährleistet die formale Richtigkeit
<b>Änderbarkeit</b>	Eignung von Dokumenten zur Ermittlung aller von einer Änderung betroffenen Dokumententeile und zur Durchführung der Änderung		Einsatz des <KUNDE>Templates gewährleistet die formale Änderbarkeit, unnötige Redundanzen sollen vermieden werden