Friedrich-Alexander-Universität Erlangen-Nürnberg
Technische Fakultät, Department Informatik

Fabian Justi
MASTERARBEIT

Moderne Datenanalyse in der Cloud basierend auf Open Source

Eingereicht am 2. Mai 2024

Betreuer: Prof. Dr.-Ing. Bernd Hindel
Prof. Dr. Dirk Riehle, M.B.A.
Professur für Open-Source-Software
Department Informatik, Technische Fakultät
Friedrich-Alexander Universität Erlangen-Nürnberg

Versicherung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

_____

Erlangen, den 02 Mai 2024

Abstract

Data analysis tools are used daily in companies, universities, and private households all over the world, with many different requirement demands on use cases, goals, performance, data types, and data set sizes. There are commercial and open source tools like Excel, Power BI, DIAdem, Python with libraries, and many more, which are used very widely in the tech industry. Even with Excel you can do image data analysis, but this is very unaesthetic, impractical, and also performs terribly. The goal of this thesis is to use nearly only OSS and OSH. There is so much efficient and proven OSS software out there. We just have to combine, configure, and extend it to have a flexible solution that can do a lot more than the tailored commercial tools. In this thesis, a cloud based data analysis platform and its infrastructure based on open source are to be advanced and used to analyze large amounts of data from various data sources. Furthermore, the aim of this thesis is to create prototypes and demonstrators that collect data, analyze, and present their results to create interest in using the state of the art data analytics. One demonstrator collects data from an electric vehicle over the OBD 2 port which is open by default. With the collected data we can analyse the EV Battery characteristics. The other demonstrator extracts and collects data from and with a Carrera race track, which is then analyzed based on the rounds and races.

Zusammenfassung

Datenanalysewerkzeuge werden täglich in Unternehmen, Universitäten und Privathaushalten auf der ganzen Welt eingesetzt, wobei die Anforderungen an Anwendungsfälle, Ziele, Leistung, Datentypen und Datensatzgrößen sehr unterschiedlich sind. Es gibt kommerzielle und OSS Werkzeuge wie Excel, Power BI, DIAdem, Python mit Bibliotheken und viele mehr, die in der Tech-Industrie sehr weit verbreitet sind. Sogar mit Excel kann man Bilddaten analysieren, wobei dies sehr unästhetisch und unpraktisch ist und eine wesentlich längere Ausführungs- und Analysezeit aufweist. Das Ziel dieser Arbeit ist es, fast nur OSS und OSH zu verwenden. Es gibt so viel effiziente und bewährte OSS-Software, dass wir sie nur noch kombinieren, konfigurieren und erweitern müssen um eine flexible Lösung zu haben, die viel mehr kann als die maßgeschneiderten kommerziellen Tools. In dieser Arbeit soll eine Cloud-basierte DAP und deren Infrastruktur auf Basis von Open Source weiterentwickelt und genutzt werden, um große Datenmengen aus verschiedenen Datenquellen zu analysieren. Darüber hinaus ist es das Ziel dieser Arbeit, Prototypen und Demonstratoren zu erstellen, die Daten sammeln, analysieren und ihre Ergebnisse präsentieren, um das Interesse an der Nutzung des Stands der Technik der Datenanalyse zu wecken. Ein Demonstrator sammelt Daten von einem Elektrofahrzeug über den OBD 2-Anschluss, der standardmäßig offen ist. Mit den gesammelten Daten können wir die Eigenschaften der Fahrzeugbatterie analysieren. Der andere Demonstrator extrahiert und sammelt Daten von und mit einer Carrera-Rennbahn, die dann anhand der Runden und Rennen analysiert werden.

Inhaltsverzeichnis

1 **Einleitung**....10
1.1 Statistik und Datenanalyse....10
1.2 Der Weg zu Kubernetes....10
1.3 Was ist Kubernetes?....10
1.4 Kubernetes Namespaces....11
1.5 Vorteile von Kubernetes....12
1.6 Zielsetzung....12
2 **Hintergrund**....14
3 **Alternative Werkzeuge**....15
4 **Anforderungen**....16
4.1 Funktionale Anforderungen OSS DAP....16
4.2 Nicht-funktionale Anforderungen OSS DAP....17
4.3 Funktionale Anforderungen der Demonstratoren....18
4.4 Nicht-funktionale Anforderungen der Demonstratoren....18
5 **Architektur**....19
5.1 OSS basierte DAP....19
5.1.1 Grundlegende Architekturentscheidungen....20
5.1.2 Überwachung und Verwaltung des Cluster....21
5.1.3 Cluster Erweiterungs- und Anpassungsdiagramm....21
5.1.4 Bausteinsicht....22
5.1.5 JupyterLab Rechenlastauslagerung....22
5.2 Carrera Demonstrator....22
5.2.1 Anpassungen ans Cluster....22
5.2.2 Systemarchitektur....22
5.2.3 Datenfluss....25
6 **Design, Implementierung, Datenauswertung und Verwendung**....26
6.1 OSS DAP....26
6.1.1 Kubernetes....26
6.1.2 AWS-Cloud....26
6.1.3 Cluster-Verwaltung....27
6.1.4 GitOps....27
6.1.5 Benutzerverwaltung....27
6.1.6 Private Cloud Lösung....27
6.1.7 Weitere verwendete OSS Lösungen und Werkzeuge....27
6.2 Carrera Demonstrator....30
6.2.1 Verwendete Hardware und Software....30
6.2.2 Demonstrationsablauf....30
6.2.3 Netzwerk und Kommunikation des Demonstrators....30
6.2.4 Aufbau der Bahn....31
6.2.5 Hardware Aufbau....32
6.2.6 Auslesen der Streckeninformationen....33
6.2.7 Modifikationen an den Carrera Autos....35
6.2.8 Sensorgenauigkeit des LSM9ds1....36
6.2.9 Sensorbefestigung....37
6.2.10 Videodaten für die Fahrzeugverfolgung....38
6.2.11 Renn-Manager....39
6.2.12 Datenaufzeichnung....39
6.2.13 Datenanalyse und Visualisierung....40
6.2.14 Automatische Carrera Autosteuerung....47
6.3 Elektroauto OBD Demonstrator....47
6.3.1 Torque Pro Analyse....47
6.3.2 OVMS Datenaggregation....48
6.3.3 Beschreibung der Demonstrators....49
7 **Evaluation**....51
7.1 Funktionale Anforderungen....51
7.2 Nicht-funktionale Anforderungen....52
7.3 Funktionale Anforderungen der Demonstratoren....53

7.4 Nicht Funktionale Anforderungen der Demonstratoren....53
7.5 OSS DAP im Vergleich....54
7.5.1 OSS DAP vs. Power BI, Qlik oder Tableau....54
7.5.2 OSS DAP vs. Anaconda Cloud oder Google Colab....54
7.5.3 OSS DAP vs. Data iku....54
7.5.4 OSS DAP vs. Diadem....54
7.5.5 OSS DAP vs. Excel....54
8 **Fazit**....55
Appendix A **Beispiel Notebook**....56
Literaturverzeichnis....58

Abbildungsverzeichnis

Abbildung 1: Alternative Werkzeuge mit Logo und Einordung....15
Abbildung 2: FunktionsMASTeR mit Bedingung (Joppich, 2016, S. 13)....16
Abbildung 3: EigenschaftsMASTeR (Joppich, 2016, S. 27)....17
Abbildung 4: UmgebungsMASTeR (Joppich, 2016, S. 30)....17
Abbildung 5: Verwendete OSS....19
Abbildung 6: Kontextsicht der OSS DAP....20
Abbildung 7: Cluster Verwaltungssicht....21
Abbildung 8: Cluster Erweiterungs- und Anpassungsdiagramm....21
Abbildung 9: OSS DAP Instanz Bausteinsicht....22
Abbildung 10: Rechenlastauslagerung mit Apache Airflow....22
Abbildung 11: Systemarchitektur des Carrera Demonstrators gedreht....24
Abbildung 12: Systemarchitektur des Carrera Demonstrators....25
Abbildung 13: Datenfluss Systemebene....25
Abbildung 14: Datenfluss Cluster....25
Abbildung 15: Carrera Demonstrator Ablauf der Benutzersicht....30
Abbildung 16: Streckenplan....31
Abbildung 17: Hardware Aufbau....32
Abbildung 18: Hardware Aufbau von unten....32
Abbildung 19: Carrera CU mit Bahnprotokollanschluss....33
Abbildung 20: Carrera CU mit ESP32 und Rundenzähleranschluss....34
Abbildung 21: FTDI....35
Abbildung 22: Carrera Auto mit verbauter Sensorik....36
Abbildung 23: Plattenspieler mit ESP32 und LSM9ds1 Sensor....36
Abbildung 24: Beschleunigungsdiagramme des LSM9ds1 Sensors....37
Abbildung 25: Geöffnetes Carrera Auto mit verbauter Sensorik....37
Abbildung 26: Lackierte Carrera Autos....38
Abbildung 27: Videoaufzeichnung mit Positionserkennung....39
Abbildung 28: Ablauf der Videoaufzeichnung und Analyse....40
Abbildung 29: Carrera Auto Position und Velocity....41
Abbildung 30: Carrera Auto Geschwindigkeit Diagramm....42
Abbildung 31: Gyro Diagramm eines Rennens....43
Abbildung 32: Carrera Auto Positions Diagramm mit Abflugpunkten....44
Abbildung 33: Splitzeiten und schnellste Runde....45
Abbildung 34: Runden- und Splitzeitanalyse....46
Abbildung 35: Elektroauto Demonstrator Datenaufzeichnung und Analyse....47
Abbildung 36: OVMS Startseite....48
Abbildung 37: OVMS Web-Editor....48
Abbildung 38: Digramme des Elektroauto Demonstrators....50

Akronyme

OVMS Open Vehicle Monitoring System

RPI Raspberry Pi

k8s Kubernetes

AWS Amazon Web Services

DevOps Development and Operations

AKS Azure Kubernetes Service

EKS Elastic Kubernetes Service

GKS Google Kubernetes-Engine

GCP Google Cloud Platform

SaaS Software as a Service

RBAC Role-based access control

CLI command line interface

RKE2 Rancher Kubernetes Engine 2

OBD On-Board-Diagnose

OSS Open-source software

OSH Open-source hardware

CSS Closed source software

CI/CD Continuous Integration and Continuous Delivery

DAP Datenanalyseplattform

JS JavaScript

Kops Kubernetes Operations

CPU central processing unit

RAM Random-access memory

HDD hard disk drive

EV electric vehicle

VM Virtual machine

IaC Infrastructure as code

Amazon RDS Amazon Relational Database Service

MR Merge Request

DAC Digital-Analog-Wandler

CU Control Unit

SoH State of Health

SoC State of Charge

OEM Original Equipment Manufacturers

IDE integrated development environment

VS Visual Studio

GPG GNU Privacy Guard

1 Einleitung

1.1 Statistik und Datenanalyse

Während die Statistik die Grundlage bildet, hat die Datenanalyse einen breiteren Anwendungsbereich. Mit statistischen Techniken können Daten analysiert werden, um die zugrunde liegenden Muster und Beziehungen zu verstehen. Die Analytik umfasst ein breiteres Spektrum an Methoden zur Gewinnung verwertbarer Erkenntnisse, zur Erstellung von Vorhersagen und zur Unterstützung von Entscheidungen.

1.2 Der Weg zu Kubernetes

Bevor die darauf folgenden Technologien entwickelt wurden war es nur möglich auf einem Computer Programme direkt auf dem System zu installieren, somit lief das Programm auf der physischen Hardware, wie CPU, RAM und HDD. Mit der Entwicklung der Virtuellen Maschine kann die Hardware virtualisiert werden, wobei diese physische Ressourcen wie CPU und RAM zwischen mehreren virtuellen Maschinen geteilt werden. Die Programme teilen sich dann dieses Dateisystem auf der virtualisierten HDD. Als Verbesserung VM kam dann der Docker Container als eine "leichtgewichtige VM"-Lösung, bei der jedes Programm sein eigenes isoliertes Dateisystem hat. Container teilen sich jedoch den Kernel des Host-Betriebssystems.

Basierend auf diesen Technologien kam dann die Cloud. In der Cloud stehen Millionen von Servern zur Verfügung, die je nach Bedarf gemietet werden können, um beispielsweise VM oder Docker Container zu betreiben. Die Bezahlung erfolgt nach Nutzung. In der Cloud gibt viele Konfigurationsmöglichkeiten sowie zusätzliche Dienste, wie Speicher, serverlose Funktionen und spezialisierte Services. Beispiele für Cloud Anbieter sind AWS, Microsoft Azure und GCP. Innerhalb dieser Cloud Plattformen oder in einem eigenen Rechenzentrum ermöglicht Kubernetes das Management von tausenden Docker-Containern, die über viele verschiedene virtuelle oder physische Maschinen verteilt sind. Somit ist auch eine gute Ausfallsicherheit leicht erreichbar. Es ermöglicht auch die Zuordnung zu Cloud Provider spezifischen Ressourcen, um eine optimale Nutzung der Infrastruktur zu gewährleisten.

1.3 Was ist Kubernetes?

Kubernetes ist eine leistungsstarke Plattform für die Container-Orchestrierung, die in verschiedenen Umgebungen eingesetzt werden kann. Hier sind einige Schlüsselmerkmale und Einsatzszenarien von Kubernetes:

Container-Orchestrierung: Kubernetes ermöglicht die Automatisierung der Bereitstellung, Skalierung und Verwaltung von Containern über einen Cluster von Computern hinweg. Es organisiert und koordiniert Containeranwendungen, um eine reibungslose Ausführung zu gewährleisten.

Skalierbarkeit und Selbstheilung: Kubernetes ist skalierbar und selbst heilend. Es kann automatisch Ressourcen hinzufügen oder entfernen, um die Auslastung zu optimieren, und es kann auf Ausfälle reagieren, indem es defekte Pods ersetzt und die Anwendungsverfügbarkeit aufrecht erhält.

Deklarative Konfiguration: Kubernetes verwendet deklarative Konfiguration, was bedeutet, dass Benutzer den gewünschten Zustand der Anwendung angeben, und Kubernetes sorgt automatisch dafür, dass die Anwendung diesen Zustand erreicht und beibehält. Dies erleichtert die Automatisierung und Wartung von Anwendungen.

Cloud-Unabhängigkeit: Kubernetes ist cloudunabhängig und kann in verschiedenen Cloud-Umgebungen eingesetzt werden. Es wird von verschiedenen Cloud-Providern unterstützt, darunter der AKS, der EKS von AWS und die GKS von GCP. Es gibt auch andere Plattformen und Dienste, die Kubernetes unterstützen, wie OpenShift, Heptio, Rancher und DigitalOcean.

Verwaltet und selbst gehostet: Kubernetes kann von Cloud-Providern verwaltet oder selbst gehostet und verwaltet werden. Verwaltete Kubernetes-Dienste, wie die oben genannten, bieten vorkonfigurierte und verwaltete Kubernetes-Cluster als Dienstleistung an. Sie können aber auch selbst gehostet und verwaltet werden, mit Werkzeugen wie kops, kubeadm oder kubespray.

Lokale Entwicklung und Tests: Kubernetes bietet verschiedene Optionen für die lokale Entwicklung und Tests von Anwendungen. Werkzeuge wie Docker-Desktop ermöglichen die Ausführung eines Kubernetes-Clusters auf einem Desktop-Computer. Leichtgewichtige Kubernetes-Distributionen sind k3s und k0s welche für ressourcenbeschränkte und Edge-Computing-Szenarien optimiert sind. Es gibt auch spezialisierte Werkzeuge wie minikube, kind, k3d und microk8s, die es Entwicklern ermöglichen, Kubernetes-Cluster schnell und einfach lokal zu erstellen und zu verwalten.

Insgesamt bietet Kubernetes eine flexible und leistungsstarke Plattform für die Container Orchestrierung und Container Verwaltung, die in verschiedenen Umgebungen eingesetzt werden kann und eine Vielzahl von Anwendungsfällen unterstützt, von der Entwicklung und Bereitstellung von Anwendungen bis hin zur Skalierung und Verwaltung von Produktionsworkloads.

1.4 Kubernetes Namespaces

In Kubernetes können Ressourcen entweder global im gesamten Cluster verfügbar sein oder einem bestimmten Namensraum (Namespace) zugeordnet werden. Ein Namespace ist ein isolierter Bereich innerhalb eines Kubernetes-Clusters, der dazu dient, Ressourcen zu gruppieren und zu isolieren. Namen müssen innerhalb ihres jeweiligen Bereichs eindeutig sein, das heißt, es kann keine zwei Ressourcen mit demselben Namen im selben Namespace geben.

RBAC und ResourceQuotas sind Mechanismen, die in Kubernetes eingesetzt werden können, um den Zugriff auf Ressourcen zu steuern und die Nutzung von Ressourcen zu begrenzen, insbesondere in Multi-Tenant-Clustern, in denen mehrere Benutzer oder Teams denselben Kubernetes-Cluster gemeinsam nutzen. RBAC ermöglicht es Administratoren, fein granulierte Zugriffsrechte für Benutzer basierend auf definierten Rollen und Berechtigungen festzulegen. ResourceQuotas ermöglichen es Administratoren, die Nutzung von Ressourcen wie CPU, Speicher und Speicherplatz pro Namespace zu begrenzen, um sicherzustellen, dass Ressourcen fair und effizient verteilt werden und um eine Überbeanspruchung des Clusters zu verhindern.

Durch die Kombination von Namensräumen, RBAC und ResourceQuotas können Kubernetes-Administratoren komplexe Multi-Tenant-Cluster effektiv verwalten und steuern, indem sie die Zugriffsrechte, Ressourcennutzung und Isolation zwischen verschiedenen Benutzern oder Teams verwalten.

1.5 Vorteile von Kubernetes

Einfache Skalierung und Verwaltung von Services: Kubernetes bietet leistungsstarke Funktionen für das Skalieren und Verwalten von Anwendungen und Services. Es ermöglicht die automatische Skalierung basierend auf der Nachfrage nach Ressourcen und bietet Mechanismen zur Lastverteilung und zur Bereitstellung von Anwendungen auf mehreren Knoten.

Viele offizielle Helm-Charts und Docker-Images für Services: Kubernetes verfügt über eine umfangreiche Sammlung von offiziellen Helm-Charts und Docker-Images für verschiedene Anwendungsfälle und Services. Diese vorgefertigten Ressourcen erleichtern die Entwicklung und Bereitstellung von Anwendungen, da sie als Ausgangspunkt oder Referenz dienen können.

TLS-Zertifikatmanagement und Subdomain-Routing: Kubernetes bietet integrierte Funktionen für das TLS-Zertifikatmanagement und das Subdomain-Routing, die die Sicherheit und Verfügbarkeit von Anwendungen verbessern. Es ermöglicht die automatische Erstellung, Verwaltung und Aktualisierung von TLS-Zertifikaten und bietet Mechanismen für das Routing von Anfragen an verschiedene Dienste basierend auf Subdomains.

Einfaches Deployment in unterschiedlichen Umgebungen: Kubernetes erleichtert das Deployment von Anwendungen in unterschiedlichen Umgebungen, einschließlich lokaler Entwicklungsumgebungen, Testumgebungen und Produktionsumgebungen. Kubernetes verfügt über Mechanismen für das Konfigurationsmanagement und die Umgebungsvariablen, welche es ermöglichen, Anwendungen in verschiedenen Umgebungen zu betreiben ohne den Code ändern zu müssen.

Cluster-Management mit kops oder mithilfe eines verwalteten Kubernetes Service z.B. EKS: Kubernetes kann entweder selbst gehostet und verwaltet werden, beispielsweise mit Hilfe von kops, einem Open-Source-Werkzeug für die Bereitstellung und Verwaltung von Kubernetes-Clustern, oder es kann von einem Cloud-Provider verwaltet werden, wie zum Beispiel AWS EKS. Diese Flexibilität ermöglicht es Unternehmen, Kubernetes nach ihren individuellen Anforderungen und Präferenzen einzusetzen.

1.6 Zielsetzung

In dieser Arbeit soll deshalb basierend auf Open Source eine Daten Analyse Plattform und dessen Infrastruktur weitergebracht, genutzt und angepasst werden, um große Datenmengen von verschieden Datenquellen zu analysieren.

Des Weitern dient die Arbeit der Entwicklung von Prototypen und Demonstratoren, die Daten erfassen, visualisieren und analysieren. Ein Demonstrator soll entworfen werden, um Daten von einem Elektrofahrzeug über den OBD 2-Anschluss zu sammeln und um verschiedene Daten wie Batteriestatus, Ladezustand, Temperatur und weitere relevante Parameter zu erfassen und zu analysieren. Der andere Demonstrator wird entwickelt, um Daten von einer Carrera-Rennbahn zu extrahieren, zu sammeln und anschließend die Runden und das Rennen zu analysieren.

Dies soll unter der Verwendung von Sensoren an der Rennstrecke und an den Fahrzeugen bewerkstelligt werden, um Daten wie Geschwindigkeit, Beschleunigung, Rundenzeiten und andere rennrelevante Informationen zu erfassen. Die gesammelten Daten sollen dann genutzt

werden, um Rennen zu analysieren, Einblicke in das Fahrverhalten und die Leistung der Fahrer zu gewinnen. Dies kann beispielsweise die Identifizierung von Fahrfehlern auf der Strecke sein, die Analyse von Rundenzeiten oder die Vergleichsanalyse zwischen verschiedenen Fahrern umfassen.

2 Hintergrund

In vielen Unternehmen sammeln sich Unmengen von Datenmaterial an, welche in ungeeigneter oder unperformanter Weise gespeichert, verarbeitet und Analysiert werden.

Entweder gehen die Daten "verloren" beziehungsweise bleiben ungenutzt, weil sie auf den ersten Blick nicht wertvoll erscheinen, oder es erfordert einen enormen Aufwand an Zeit und Geld, sie in die richtige Form zu bringen. Das führt zu Betriebsblindheit, ungenutztem Wissen, verschwendetem Potenzial, teuren Prozessen und Ressourcenverschwendung in Unternehmen.

Die momentane Werkzeuglandschaft besteht bisher zwischen extrem teuren, maßgeschneiderten Hochleistungslösungen oder unpassenden Standardanwendungen. Meistens können Standardanwendungen nicht die gesamte Funktionalität liefern und individuelle High-End-Lösungen sind teuer.

Häufige Probleme oder Zustände die zu Problemen führen sind zum Beispiel manueller Export, Speicherort und Ablagesystem welche Fehleranfällig sind. Daraus entstehen oft inkonsistente Datensätze. Des Weiteren leidet die Auswertbarkeit, oftmals ist deshalb eine manuelle Datenaufbereitung, beispielsweise die Kontextualisierung nötig. Ein anderes Problem ist der Zeitverlust durch Datentransfer (Kopieren und Speichern) und manuelle Aktivitäten, wodurch sich der gesamte Verarbeitungszyklus erheblich gegenüber eines Verarbeitugszykluses mit Echtzeitdatenerfassung und anschließender Verarbeitung unterscheidet. Des öfteren ist auch die Performance bei der lokalen Berechnung von der Leistung des verwendeten Rechners abhängig und nur eine Person kann diese Ressource nutzen.

Zu den verschiedenen OSS- und Datenanalysewerkzeugen gibt es bereits Erfahrungen in deren Verwendung und des ineinandergreifens dieser miteinander. In der Vergangenheit wurden schon ähnliche Systeme mit einem wesentlich geringeren Umfang, sowie Funktionsumfang wie das geplante System, hochgezogen und verwendet.

Bezüglich der Demonstratoren ist der Hintergrund für den Elektroauto Demonstrator einerseits dass die Batterie eines Elektrofahrzeugs etwa 35% des Gesamtwerts ausmacht. Daher ist der Restwert eines gebrauchten oder geleasten Elektrofahrzeugs maßgeblich vom Gesundheitszustand (SoH) der Antriebsbatterie abhängig. Natürlich verwenden Original Equipment Manufacturers (OEMs) Batteriemanagementsysteme und eigene Telemetrie zur Überwachung des SoH jedoch sind diese nicht frei zugänglich gewesen, eine Entscheidung des Europäischen Gerichtshofs (EuGH) in der Rechtssache C-296/22. Dieses Urteil entschied, das OEMs den freien Zugang zur On-Board-Diagnose (OBD) und den Fahrzeugendaten ermöglichen müssen, wodurch ein Batteriepass und Nutzerdaten frei zugänglich werden sollen.

Der Carrera Demonstrator soll dazu dienen die moderne Datenanalyse in der Cloud den verschiedensten Personen anhand eines bekannten Spielzeugs näher zu bringen und aufzuzeigen was alles mit einer OSS DAP möglich ist. Um das Bewusstsein zu schaffen, dass die Verwendung einer OSS DAP für jegliche Daten oder Projekte eine Lohnenswerte Option darstellt.

3 Alternative Werkzeuge

- C3.ai (https://www.c3.ai/)
- Data iku (https://www.dataiku.com/product/dataiku-for-analytics/)
- Alteryx (https://www.alteryx.com/solutions/role/alteryx-for-data-analysts)
- Qlik (https://www.qlik.com/)
- Power BI (https://www.microsoft.com/en-us/power-platform/products/power-bi)
- Excel (https://www.microsoft.com/de-de/microsoft-365/excel?market=de)
- Tableau (https://www.tableau.com/)
- Knime (https://www.knime.com/)
- Matlab (https://www.mathworks.com/products/matlab.html)
- DIAdem (https://www.ni.com/en/shop/data-acquisition-and-control/application-software-for-data-acquisition-and-control-category/what-is-diadem.html)
- AVL (https://www.avl.com/en/testing-solutions/all-testing-products-and-software/connected-development-software-tools/avl-data)
- Anaconda (https://www.anaconda.com/)
- Jupyterlab (https://jupyter.org/)
- Binder (https://mybinder.org/)
- Bitnami (https://bitnami.com/)
- AWS Analytics (https://aws.amazon.com/big-data/datalakes-and-analytics/?nc2=h_ql_prod_an_a)

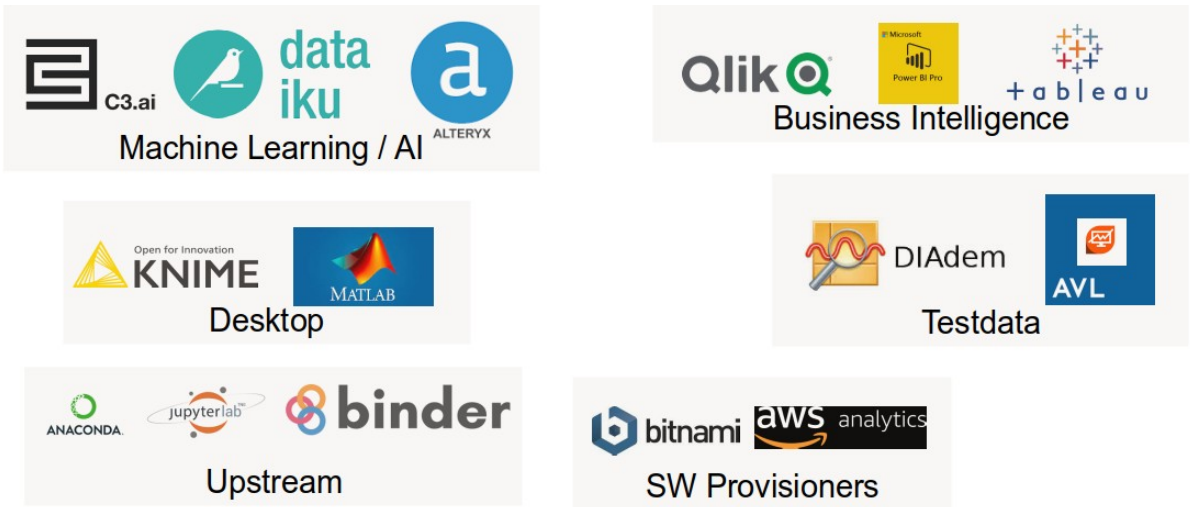

Abbildung 1: Alternative Werkzeuge mit Logo und Einordung

4 Anforderungen

In diesem Kapitel werden Anforderungen beschrieben, welche an die OSS DAP, das Infrastruktur Cluster und die Demonstratoren gestellt werden.

Zunächst wird das Schema zur Beschreibung erläutert, das im weiteren Verlauf verwendet wird. Anschließend werden die einzelnen Anforderungen aufgeführt, wobei zuerst die funktionalen Anforderungen (FA) genannt werden, bevor auf die nicht-funktionalen Anforderungen (NFA) eingegangen wird.

4.1 Funktionale Anforderungen OSS DAP

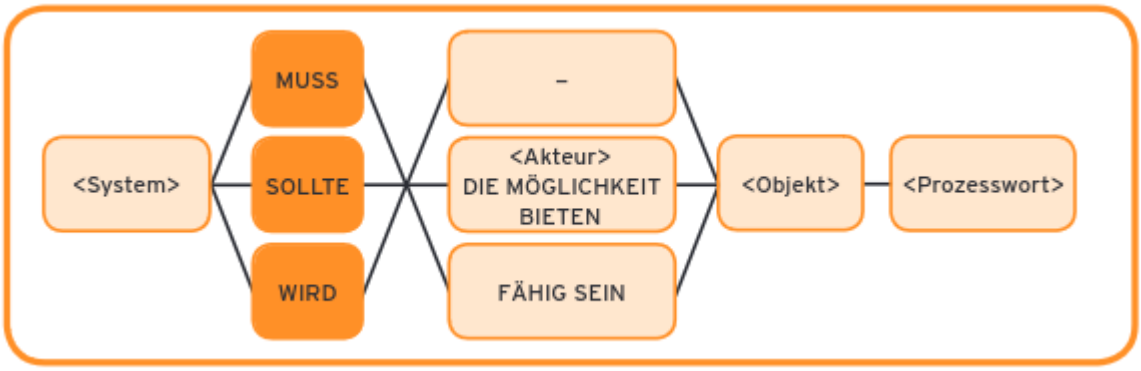

Abbildung 2: FunktionsMASTeR mit Bedingung (Joppich, 2016, S. 13)

FA-1: Der Zugang zur OSS DAP muss einfach sein und erfordert lediglich einen Webbrowser, um darauf zuzugreifen.

FA-2: Die OSS DAP soll verwaltete OSS Werkzeuge verwenden, wodurch sie stets flexibel und auf dem neuesten Stand der Technik bleibt.

FA-3: Es soll mithilfe von Werkzeugen die Überwachung und Verwaltung der verfügbaren und genutzten OSS-Pakete sichergestellt werden.

FA-4: Benutzer sollen die Möglichkeit haben, gemeinsam an denselben Datensätzen und Datenanalyse Notebooks zu arbeiten, wodurch eine reibungslose Zusammenarbeit ermöglicht wird.

FA-5: Die OSS DAP soll auf Cloud-Ressourcen zurückgreifen, wodurch ihre Leistung nicht durch die Kapazität der lokalen Hardware begrenzt ist.

FA-6: Die OSS DAP muss in der Lage sein, eine Vielzahl von Datentypen zu akzeptieren und zu analysieren. Hierzu gehören unter anderem Tabellenkalkulationen, Zeitreihen, Bilder sowie Audio- und Videodateien.

FA-7: Es soll ein gemeinsamer Speicher eingerichtet werden, der allen Benutzern den Zugriff auf Notebooks und Daten ermöglicht.

FA-8: Die OSS DAP muss die effiziente Bereitstellung und Verwaltung gemeinsam genutzter Rechenressourcen für Docker-Container gewährleisten, um eine optimale Auslastung und Leistungsfähigkeit sicherzustellen.

FA-9: Die Authentifizierung sollte mittels eines bestehenden Domänenaccounts möglich sein.

4.2 Nicht-funktionale Anforderungen OSS DAP

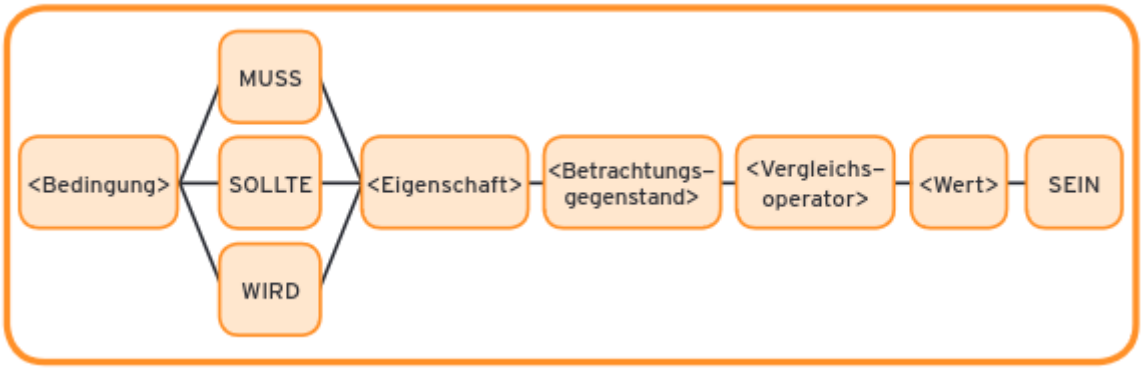

Abbildung 3: EigenschaftsMASTeR (Joppich, 2016, S. 27)

Die NFA sind auch nach Vorlagen von Joppich (2016) formuliert.

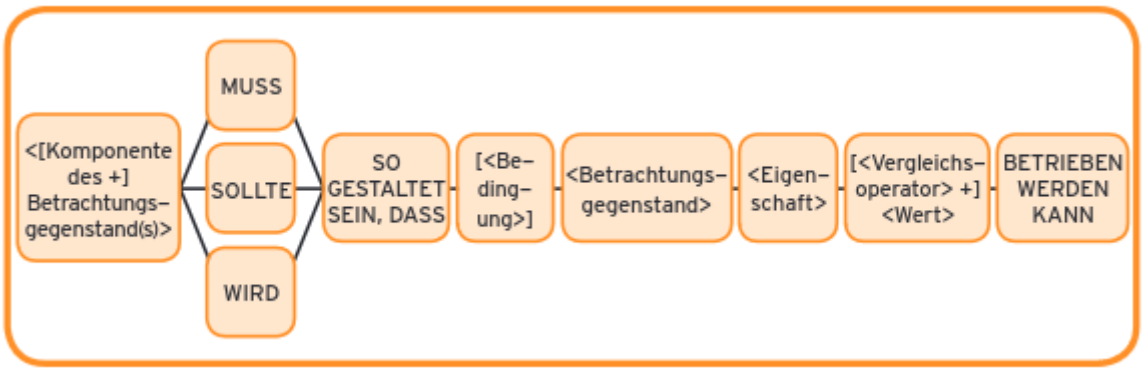

Abbildung 4: UmgebungsMASTeR (Joppich, 2016, S. 30)

NFA-1: Die OSS DAP muss so gestaltet sein, dass sie innerhalb von AWS und lokalen Clusterlösungen zur Verfügung gestellten Umgebung aufgespielt und betrieben werden kann.

NFA-2: Die OSS DAP sollte in der Lage sein, mit dem Wachstum der Daten und Benutzer zu skalieren, um eine effiziente Verarbeitung großer Datenmengen zu ermöglichen.

NFA-3: Die OSS DAP sollte nahtlos mit anderen Systemen und Werkzeugen integrierbar sein, um Daten aus verschiedenen Quellen zu erfassen, zu verarbeiten und zu analysieren.

NFA-4: Die OSS DAP sollte robuste Sicherheitsmechanismen bieten, um die Vertraulichkeit, Integrität und Verfügbarkeit der Daten zu gewährleisten und Datenschutzrichtlinien und -vorschriften einzuhalten.

4.3 Funktionale Anforderungen der Demonstratoren

FA-1: Die Notebooks sollen eine Kombination aus Dokumentation und Python-Code enthalten.

FA-2: Der Datenfluss soll automatisiert erfolgen, beginnend von der Datenquelle bis zum Cluster.

FA-3: Der Carrera Demonstrator soll transportabel und mit annehmbaren Aufwand ab- und auf-baubar sein.

FA-4: Der jeweilige Demonstrator muss fähig sein Daten der zugrunde liegenden Demonstratorbasis zu aggregieren.

FA-5: Der jeweilige Demonstrator muss fähig sein die aggregierten Daten zu verarbeiten und zu visualisieren.

4.4 Nicht-funktionale Anforderungen der Demonstratoren

NFA-1: Der Carrera Demonstrator muss so gestaltet sein, dass er innerhalb einer lokalen Clusterlösungen zur Verfügung gestellten Umgebung aufgespielt und betrieben werden kann.

NFA-2: Der Carrera Demonstrator muss so gestaltet sein, dass er ohne großen Werkzeugeinsatz transportiert werden kann.

5 Architektur

5.1 OSS basierte DAP

Die OSS DAP basiert auf verwalteten OSS-Werkzeugen, bei denen relevante Best-in-Class-Werkzeuge und hochmoderne Bibliothekspakete, die von Data Scientists und Machine Learning Ingenieuren häufig verwendet werden, von Anfang an verfügbar gemacht werden.

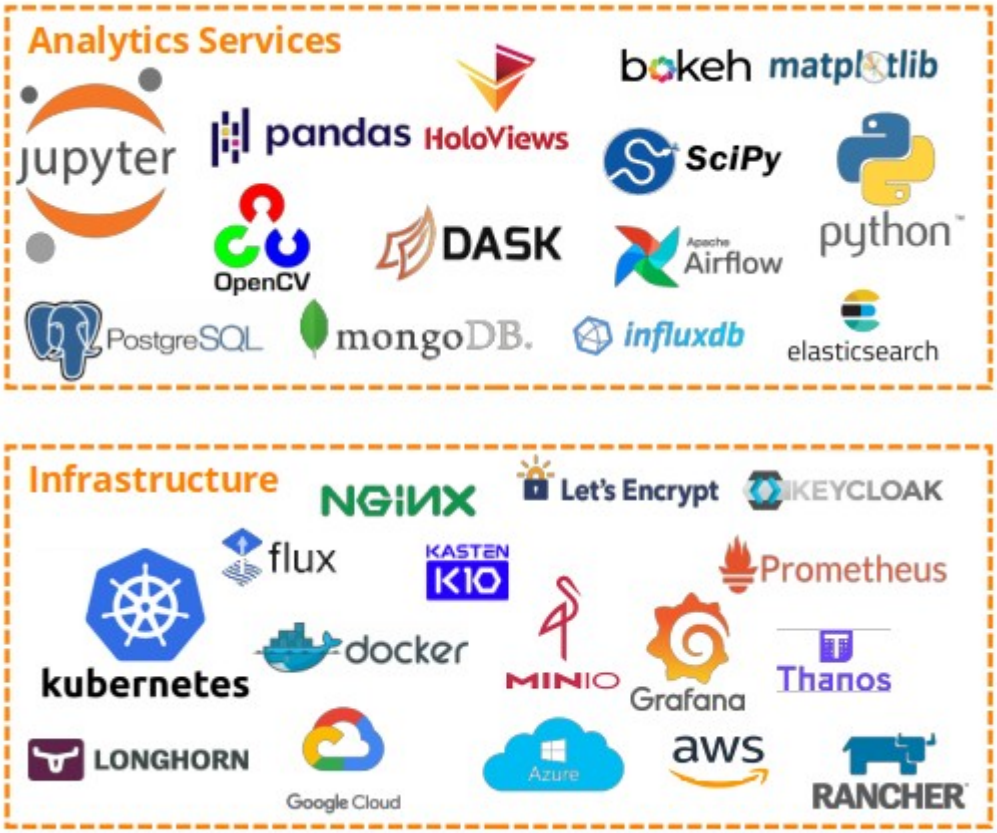

Abbildung 5: Verwendete OSS

Die OSS DAP wurde speziell als flexible Analyselösung entwickelt, die mit einer Vielzahl von Datenquellen integriert werden kann.Wir haben erfolgreich verschiedene Instanzen mit benutzerdefinierten, handelsüblichen Datenkonnektoren implementiert, welche Datenquellen von IoT-Edge-Geräten bis hin zu unternehmensinternen oder cloudbasierten Datenspeichern aufnehmen können.

Die OSS DAP wurde bewusst als vielseitige Plattform konzipiert, die ein breites Spektrum an Datentypen verarbeiten kann, das über typische Geschäftsdaten hinausgeht. Es wurden Standardkonfigurationen und benutzerdefinierte Dateneingabemodule entwickelt und verwendet um Sensormessungen von eingebetteten Systemen sowie unstrukturierte digitale Daten in der OSS DAP zu verarbeiten. Somit kann die OSS DAP für die Verarbeitung jeglicher Daten verwendet werden.

Die einzelnen Services, Werkzeuge und deren Verwendung werden in Kapitel 6 näher beleuchtet und erklärt.

5.1.1 Grundlegende Architekturentscheidungen

- Entwickelt unter Berücksichtigung von Datenschutz und Vertrauen durch die Bereitstellung der OSS DAP auf jeder Kundeninfrastruktur, ob Cloud-basiert oder lokal, wodurch das Risiko des Datenzugriffs durch Dritte ausgeschlossen wird.
- Browser-basierter Zugriff zur Unterstützung der Zusammenarbeit unter Wahrung der Sicherheit
- Maßgeschneiderter Ansatz für sich entwickelnde Anforderungen an die Datenanalyse
- Skalierbarkeit durch die Implementierung zusätzlicher Analyse- und Visualisierungskomponenten, um steigende Anforderungen an Informationen und verbesserte Ergebnisse zu erfüllen
- Organisatorische Transparenz durch verbesserte Datenverfügbarkeit
- Flexibilität, um sowohl strukturierte Daten als auch Multimedia-Dateien aus internen und externen Quellen zu analysieren
- Umfassende Einrichtungs Automatisierung um innerhalb weniger Stunden browserbasierten Zugriff auf die DAP zu ermöglichen.
- Umfangreiche Analyse- und Visualisierungswerkzeugen , die in einem einzigen Framework zusammenarbeiten
- webbasiert und Cloud-nativ
- Reproduzierbare Datenverarbeitung und Visualisierung
- Verwendung von bewährten OSS Werkzeugen
- Absicherung durch HTTPS und TLS
- Infrastruktur als Code durch GitOps

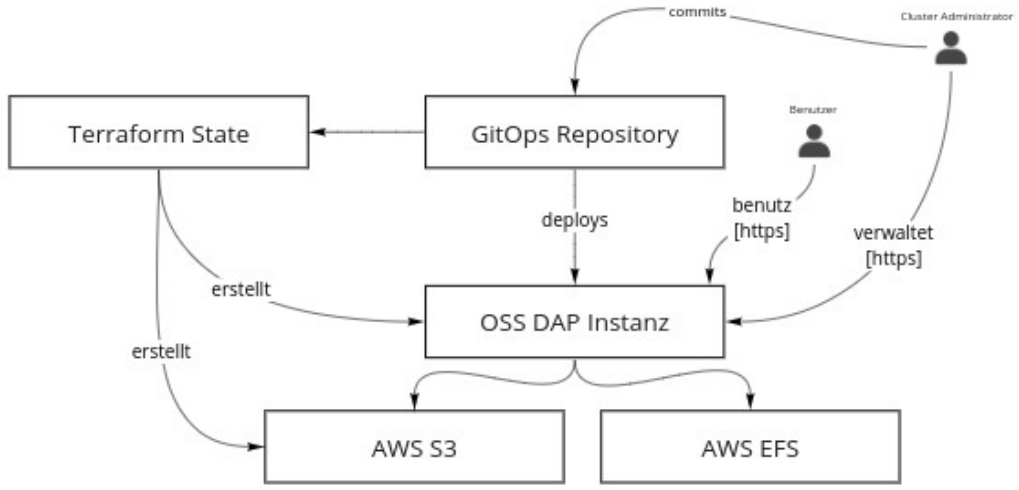

Abbildung 6: Kontextsicht der OSS DAP

5.1.2 Überwachung und Verwaltung des Cluster

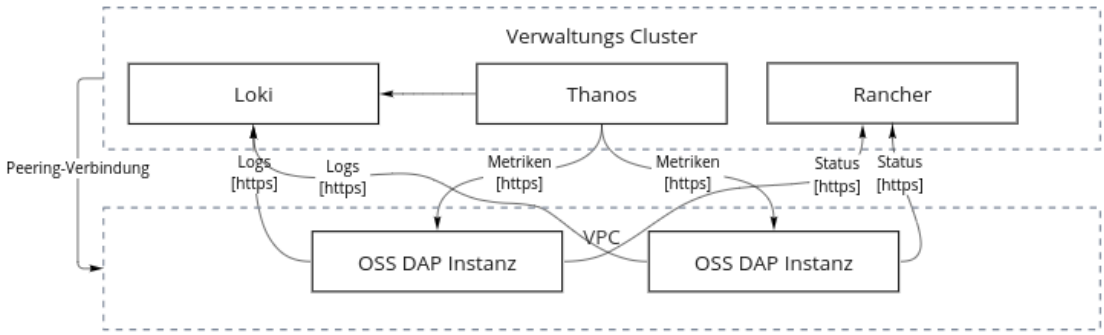

Abbildung 7: Cluster Verwaltungssicht

5.1.3 Cluster Erweiterungs- und Anpassungsdiagramm

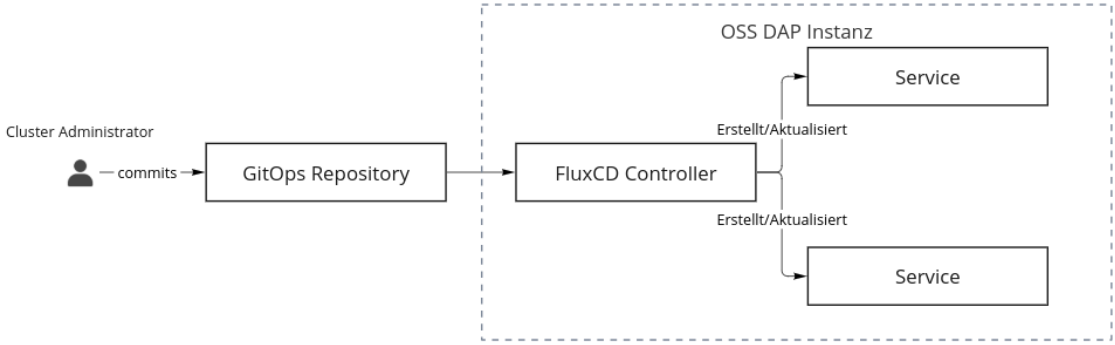

Abbildung 8: Cluster Erweiterungs- und Anpassungsdiagramm

5.1.4 Bausteinsicht

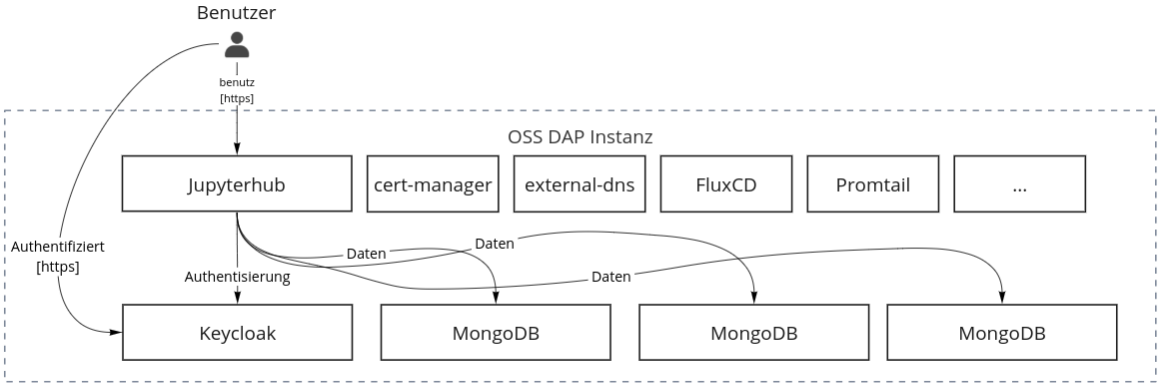

Abbildung 9: OSS DAP Instanz Bausteinsicht

5.1.5 JupyterLab Rechenlastauslagerung

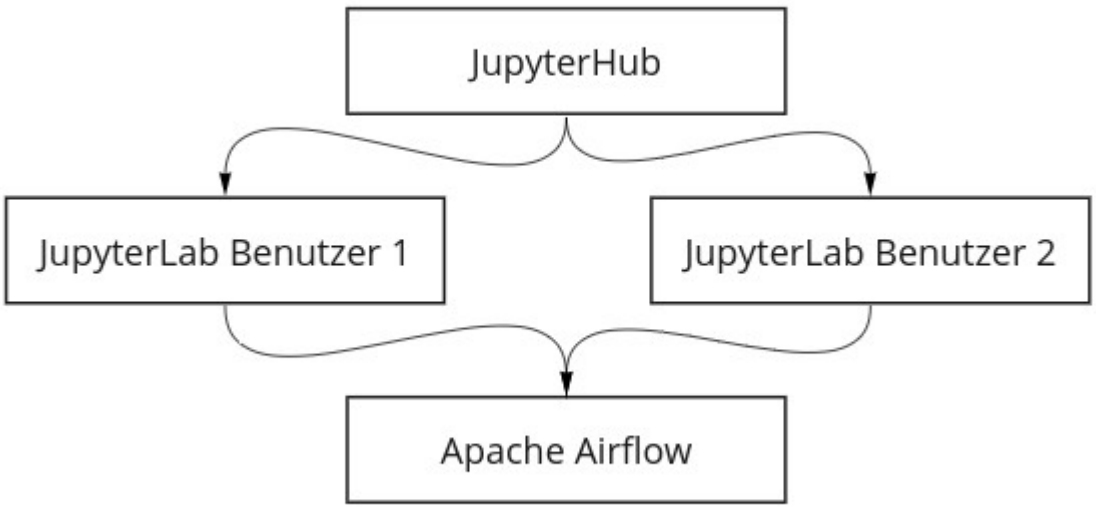

Abbildung 10: Rechenlastauslagerung mit Apache Airflow

5.2 Carrera Demonstrator

5.2.1 Anpassungen ans Cluster

Das Cluster wird mit Harvester bereitgestellt und verwaltet.

5.2.2 Systemarchitektur

Interessenten können Slotcar-Piloten werden und an einem Wettbewerb um die schnellste Zeit teilnehmen. Zwei Piloten fahren gleichzeitig ein Rennen über ca. 10 Runden (ca. 1 Minute).

Während jedes Rennens werden Telemetriedaten aufgezeichnet und anschließend in Cluster hochgeladen.

Nach dem Rennen werden die Telemetriedaten ausgewertet, um Verbesserungsmöglichkeiten zu erkennen. Der Pilot kann basierent auf den schlüssen die er aus der Analyse gezogen hat, ein zweites Rennen bestreiten, um seine Rennzeit zu verbessern.

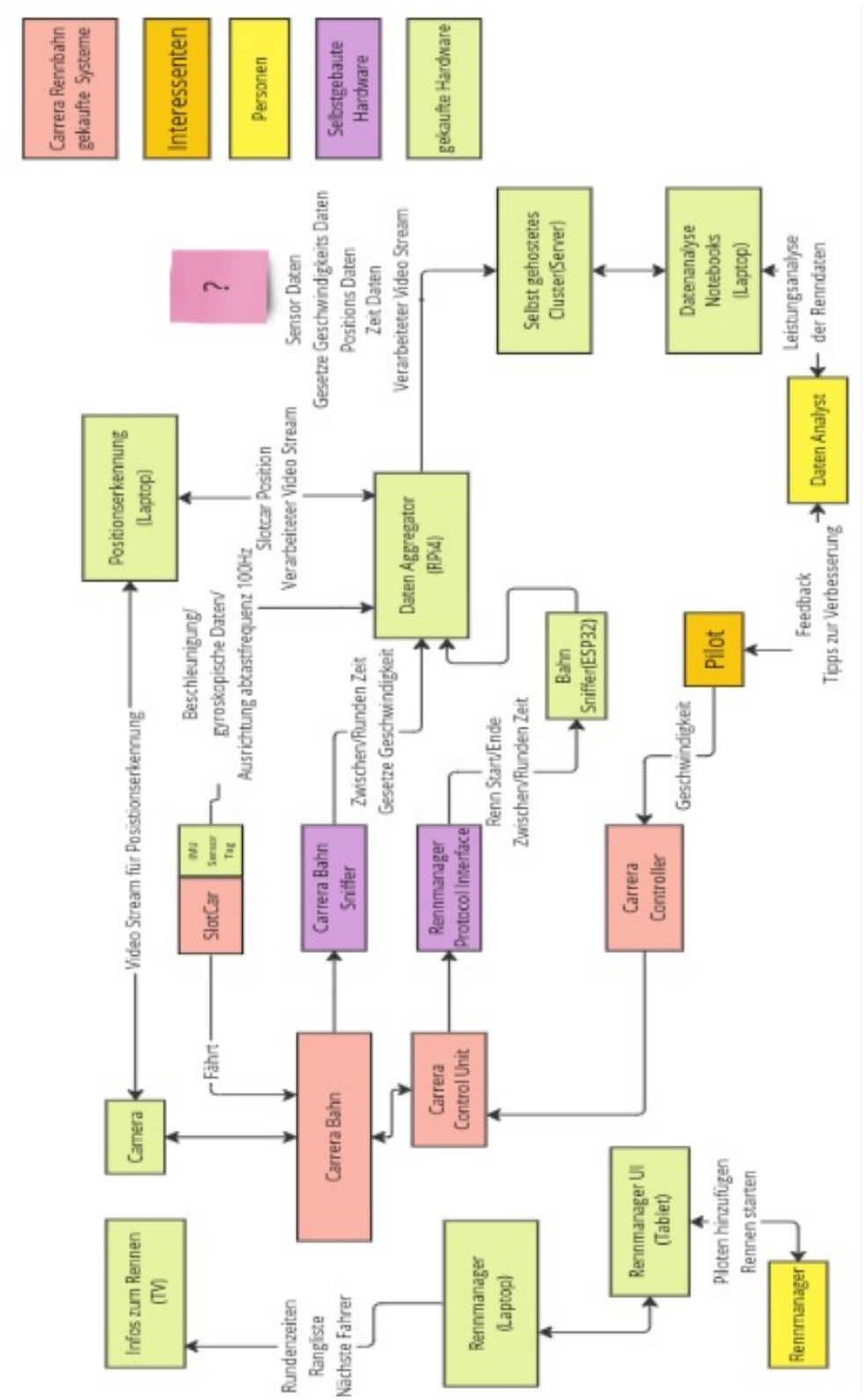

Abbildung 11: Systemarchitektur des Carrera Demonstrators gedreht

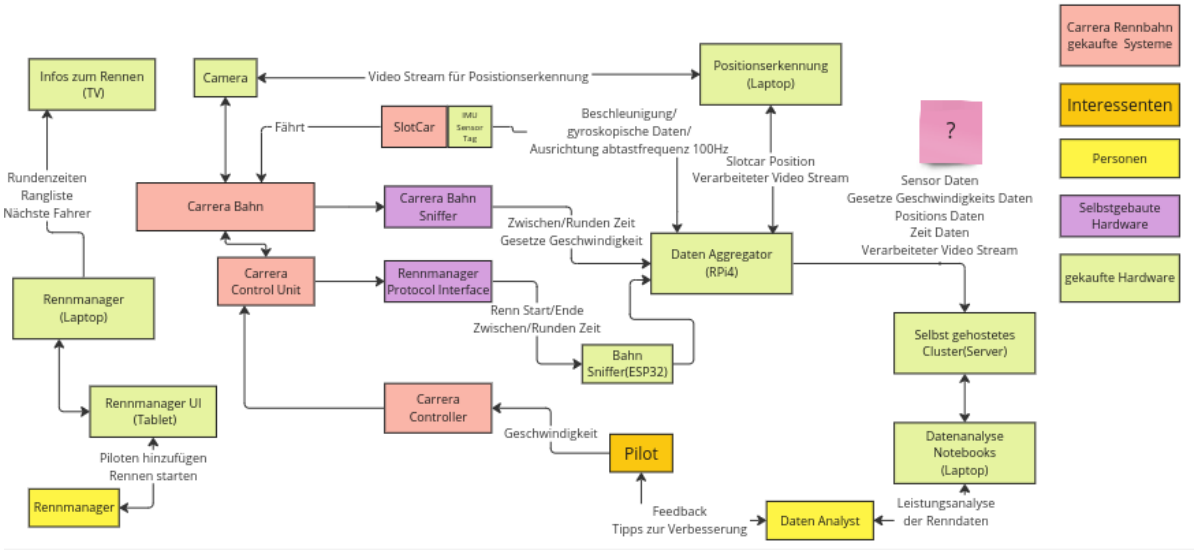

Abbildung 12: Systemarchitektur des Carrera Demonstrators

5.2.3 Datenfluss

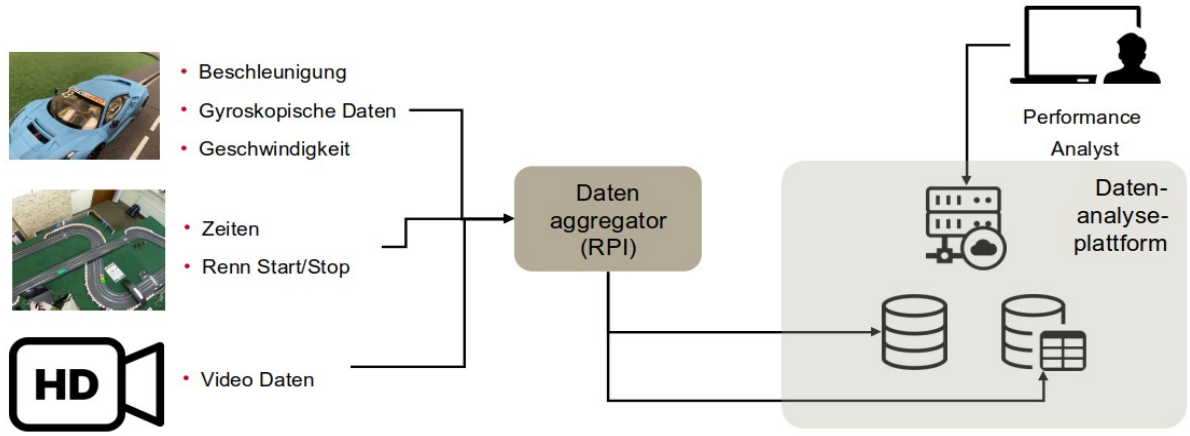

Abbildung 13: Datenfluss Systemebene

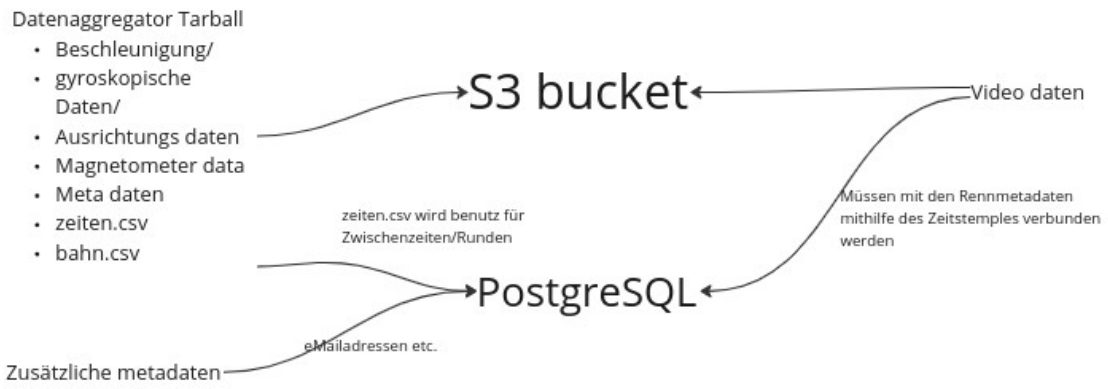

Abbildung 14: Datenfluss Cluster

6 Design, Implementierung, Datenauswertung und Verwendung

6.1 OSS DAP

Die OSS DAP besteht aus einer Reihe von Diensten, die so konfiguriert sind, dass sie ineinander-greifend zusammenarbeiten. Im folgenden sind Services und Werkzeuge mit ihren Funktionalitäten und ihrem Einsatz im OSS DAP beschrieben.

6.1.1 Kubernetes

- **Kubectl**: CLI zur Interaktion mit Cluster-Ressourcen welches Berechtigungen gemäß RBAC hat. Ressourcen können direkt oder über API-Dateien verwaltet werden. API-Dateien verwenden die YAML-Syntax. Die Konfiguration kann mehrere Cluster und Benutzer enthalten.Die meisten Befehle verwenden das folgende Format: kubectl <Aktion> <Ressourcentyp> <Name> [Optionen]

  Wir benötigen kubectl für die Kommunikation mit dem Cluster und für OpenLens.
- **Pod**: kleinste Einheit im Cluster. Kann aus einem oder mehreren Containern mit gemeinsamem localhost und optional gemeinsamen Speicher bestehen.
- **Vertical Pod Autoscaler**: zur dynamischen Anpassung der Ressourcennutzung der Pods zu Optimierung der Ressourcenauslastung.
- **ConfigMap**: Key-Value-Speicherung, welche Umgebungsvariable verwendet oder als Ordner/Datei gemountet werden kann. Der Inhalt der eingebundenen Dateien wird automatisch aktualisiert.
- **Secret**: Key-Value-Speicher Ähnlich wie ConfigMap, dieser kann separate Zugriffsrechte haben. Kann als Umgebungsvariable verwendet werden oder als Ordner/Datei gemountet werden. Der Inhalt der gemounteten Dateien wird automatisch aktualisiert.
- **Service**: Festlegung eines Namen für (mehrere) Pods. Lastausgleich zwischen Pods (ClusterIP). Nur von innerhalb des Clusters sichtbar. Pods außerhalb des Clusters sichtbar machen mit LoadBalancer und NodePort Proxy zu externen Ressourcen (ExternalName).
- **Ingress**: ermöglicht den Zugriff auf Dienste im Cluster von außen HTTP/HTTPS, Routing von spezifischen Pfaden und unterstützt TLS-Terminierung am Ingress-Controller.
- **Persistentes Speichern**: dynamische oder statische Provisionierung. Persistenter Speicher auf externem Volumes. Die Implementierung des Volumes ist anbieterspezifisch.

6.1.2 AWS-Cloud

Bei der Verwaltung der Daten in der AWS Cloud kommt der Amazon RDS zum Einsatz welcher es ermöglicht relationale Datenbanken wie zum Beispiel Cassandra, MongoDB, MySQL und PostgreSQL. in der Cloud zu erstellen und zu verwalten, ohne sich um die zugrunde liegende Infrastruktur kümmern zu müssen. Der Service bietet Funktionen wie

automatische Skalierung, automatische Backups, Patch-Management und Überwachung, um die Verwaltung und Wartung von Datenbanken zu erleichtern. Mit dem AWS Cli ist es möglich, über die Kommandozeile mit der AWS-Cloud zu kommunizieren. Es ermöglicht auch die Authentifizierung mit Access und Secret Key für andere Dienste, die die AWS Cloud nutzen.

6.1.3 Cluster-Verwaltung

Die Cluster werden mit Rancher bereitgestellt und verwaltet.

6.1.4 GitOps

- Cluster-Konfiguration wird im Git-Repository gespeichert
- Änderungsworkflow mit PR
- PRs werden geprüft und in der Staging-Umgebung bereitgestellt
- Zusammengeführte PRs lösen Controller im Produktionscluster aus
- Der GitOps-Controller prüft die neue Revision und wendet die Konfiguration an.

6.1.5 Benutzerverwaltung

Die Benutzerverwaltung bietet ein zentrales Authentifizierungs- und Autorisierungsmanagement für Plattformbenutzer und eine Single-Sign-On-Lösung für alle Dienste. Wir verwenden Keycloak, um Benutzer und OIDC-Clients für die Authentifizierung und Autorisierung zu verwalten. Für die automatische Konfiguration verwenden wir zusätzlich keycloak-config-cli, mit dem wir alle Keycloak-Einstellungen über eine Konfigurationsdatei konfigurieren können.

6.1.6 Private Cloud Lösung

Für den Betrieb unserer Cluster auf lokaler Hardware verwenden wir Harvester zur Verwaltung der virtuellen Maschinen wie z.B beim Carrera Demonstrator.

6.1.7 Weitere verwendete OSS Lösungen und Werkzeuge

- **YAML:** die für den Menschen lesbare Daten-Serialisierungssprache welche unter anderem für die Kubernetes-Manifestdateien eingesetzt wird.
- **Helm:** Paketmanager für Kubernetes, der das Templating von Ressourcen, die Versionierung von Charts und eine Vielzahl von offiziellen Charts zur einfachen und direkten Verwendung bietet. Mit Helm können Benutzer Deployments mehrerer Instanzen mit unterschiedlichen Konfigurationen verwalten und über eine CLI mit Charts, Releases und Repositories interagieren. Die Plattform bietet Flexibilität bei der Konfiguration, da Werte über Kommandozeilen-Flags oder aus YAML-Dateien gesetzt werden können. Darüber hinaus bietet Helm Plugins für die Behandlung von Geheimnissen, Diagramm-Uploads und andere Funktionen. Die Integration mit kubectl ermöglicht eine nahtlose Verwendung von Kubernetes-Ressourcen. Die Befehle von Helm verwenden folgendes Format: 'helm <Aktion> <Release> <Repository> [Optionen]'.
- **Helm-Chart:** Ein Helm-Chart ist ein nützliches Instrument zur Verwaltung von Kubernetes-Anwendungen, da es viele vorgefertigte Kubernetes-API-Dateien enthält

und oft Standardwerte mitbringt. Diese Charts können andere Helm-Charts einschließen und sind normalerweise an eine bestimmte Version und ein Docker-Image gebunden. Der große Vorteil ist, dass somit die Konfiguration von der Implementierung abstrahiert, sodass sich Nutzer nicht um Implementierungsdetails kümmern müssen. Die Werte innerhalb der Chart sollten nach Komponenten strukturiert sein, was die Verwaltung und Anpassung erleichtert.

- **Rancher:** wird verwendet, um das Cluster bereitzustellen, zu verwalten und den Status des Clusters, oder Protokolle einzusehen, sowie den Status von Pods zu überprüfen.
- `elastic`**:** ELK Stack in Kubernetes (Elasticsearch, Logstash, Kibana) und Beats Client. Dient dazu die Logs von Geräten von überall einsehen zu können, ohne direkten Zugriff auf das Gerät zu benötigen.
- **Keycloak-Operator:** automatisierte Verwaltung und Skalierung von Keycloak-Instanzen im Cluster. Dadurch ist die Authentifizierung und Autorisierung bei den Anwendungen und Diensten mit Single-Sign-On möglich.
- **JupyterHub:** parallele - und gemeinsame Nutzung von Jupyter Notebooks
- **Git Server:** Versionsverwaltung und CI/CD pipelines
- **K10:** als Backup Lösung für die Sicherung und Wiederherstellung der Daten des Clusters
- **Terraform:** für die Infrastrukturautomatisierung und Verwaltung. Die Terraform-Konfigurationsdateien werden im Git repository versionsverwaltet.
- **S3, MinIO und NFS:** Speicherlösung je nachdem ob in der privaten Cloud-Umgebung wie beim Carrera Demonstrator mit MinIO oder beim E-Auto Demonstrator in AWS mit S3.
- **Longhorn:** Kubernetes Block Storage Lösung zur Sicherstellung eines hochverfügbaren Speicher für persistenten Storage der zum Beispiel auf drei verschiedenen Nodes gespeichert wird und somit einfach repliziert werden kann. Ebenfalls für die Erstellung von Snapshots innerhalb des Clusters. Longhorn kann mithilfe von S3, Minio oder NFS für Backups außerhalb des Clusters verwendet werden. Longhorn ist von Rancher, aber man benötigt Rancher nicht, um Longhorn zu verwenden.
- **Cert-manager:** automatisiertes TLS-Zertifikatmanagement
- **Prometheus und Grafana:** Überwachung der Cluster.
- **Harbor**: als Speicher für Docker Images und Helm Charts.
- **Flux-CD**: Flux hilft uns, unsere Helm-Releases in Kombination mit unserem GitLab-Repository besser zu verwalten. Verwenden Sie diese Installationsanleitung, um es für Ihr Betriebssystem einzurichten.
- **OpenLens**: OpenLens ist ein Dekstop-Kubernetes-Verwaltungstool für Cluster im Stil einer IDE, ähnlich wie VS Code. Es ermöglicht das Hinzufügen von KubeConfigs und das Betrachten der Ressourcen des Clusters, während es auch bearbeitet werden kann.
- **Sops:** Sops ist ein Ver-/Entschlüsselungswerkzeug von Mozilla, um Dateien (YAML, JSON, etc.) mit verschiedenen Schlüsselverwaltungsdiensten und -standards wie AWS-KMS, Azure Key-Vault oder GPG zu verschlüsseln. In unserem Fall wird es zum Ver- und Entschlüsseln der K8s-Secrets verwendet.

- **GitLab:** DevOps-Plattform zur Versionsverwaltung, CI/CD und Zusammenarbeit.
- **GitLab CI:** direkt in GitLab integrierte CI/CD-Lösung, wodurch sich die Einrichtung und Nutzung sehr einfach gestaltet.
- **Apache airflow:** mithilfe der workflow execution engine Apache airflow werden Verarbeitungsaufgaben aus Jupyter-Notebooks ausgelagert.
- **Renovate und Renovate Bot:** hochgradig konfigurierbares Tool zur Aktualisierung von Abhängigkeiten, das wir verwenden, um unsere Abhängigkeiten mit der Upstream-Entwicklung auf dem neuesten Stand zu halten und unsere eigenen Repositories zu synchronisieren. Renovate Bot wird mit der Gitlab CI geplant und ausgeführt Renovate erstellt einen MR mit einer geeigneten Konfiguration, wenn er beim Scannen zum ersten Mal ein neues Repository sieht. Mit einer Konfiguration im Repository kümmert sich Renovate dann auch um die Aktualisierung von Abhängigkeiten beziehungsweise stellt einen MR, dass eine Aktualisierung vorhanden ist.
- **Cassandra:** Open-Source-Datenbank, die für die Verwaltung großer Mengen von strukturierten und unstrukturierten Daten entwickelt wurde. Sie ist bekannt für ihre hohe Skalierbarkeit und Ausfallsicherheit und wird häufig in verteilten Systemen eingesetzt.
- **Kanister:** Open-Source-Werkzeug, das die Bereitstellung und Verwaltung von Anwendungen in Containern vereinfacht. Es bietet Funktionen wie automatische Skalierung, Überwachung und Sicherheit für Containeranwendungen.
- **MongoDB:** dokumentenorientierte NoSQL-Datenbank, die für die Speicherung und Verwaltung von unstrukturierten Daten entwickelt wurde. Sie ermöglicht eine flexible Datenmodellierung und Skalierbarkeit und wird häufig in Anwendungen eingesetzt, die eine schnelle Verarbeitung großer Datenmengen erfordern.
- **MySQL:** relationale Datenbank, die für die Speicherung und Verwaltung strukturierter Daten entwickelt wurde. Sie ist eine der am häufigsten verwendeten Open-Source-Datenbanken und bietet Funktionen wie Transaktionsunterstützung, ACID-Konformität und Skalierbarkeit.
- **PostgreSQL:** relationale Datenbank, die für die Speicherung und Verwaltung strukturierter Daten entwickelt wurde. Es ist bekannt für seine erweiterten Funktionen wie Unterstützung für komplexe Abfragen, Geodatenverarbeitung und JSON-Datentypen. PostgreSQL ist ebenfalls eine Open-Source-Datenbank und bietet eine hohe Zuverlässigkeit und Skalierbarkeit.
- **Poetry:** Poetry ist das Werkzeug, das wir zur Verwaltung von Paketen und Abhängigkeiten für unser Python-Projekt verwenden. Es kümmert sich auch um die Auflösung von Abhängigkeiten durch SAT-Solving und installiert daher nur "kompatible" Versionen. Poetry erstellt und verwendet eine Datei poetry.lock, um Versionen von Abhängigkeiten festzuhalten. Diese Datei enthält alle Abhängigkeiten zusammen mit einer festen Version und anderen Metainformationen, wie Prüfsummen für jedes Paket.

6.2 Carrera Demonstrator

6.2.1 Verwendete Hardware und Software

Als Basis für den Demonstrator wurde eine Carrera Digital 132 angeschafft. Zum Datenaggregieren von der Carrera-Rennbahn wurden mehrere Esp 32 feather v2, Arduinos, Optokoppler, Step down converter, RPIs, LSM9ds1 Sensoren, Li-Akkus, Weitwinkel 4k webcam und FTDI LC231X angeschafft. Für den transportablen Aufbau wurden Pressspanplatten, Holzleisten, Dübel, Schrauben, Tischbeine und Spannschaniere gekauft. Softwareseitig läuft auf den ESP32 ein Zephyr. Auf einem RPI mit Yocto Betriebsystem werden hauptsächlich Python und Python Bibliotheken genutzt. OpenLap und OpenCV werden beide mithilfe eines Laptops genutzt.

6.2.2 Demonstrationsablauf

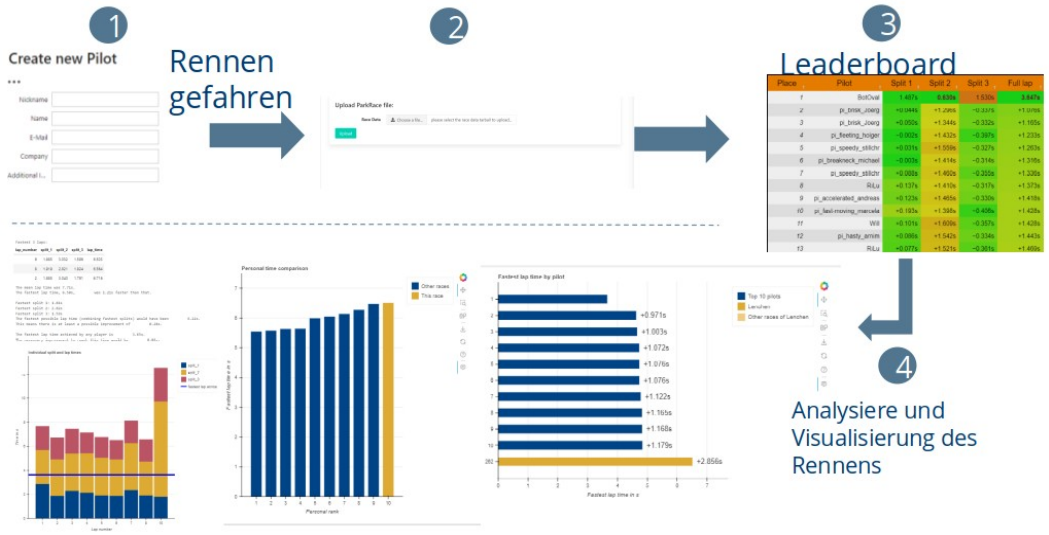

Abbildung 15: Carrera Demonstrator Ablauf der Benutzersicht

Es wird ein Fahrer in der Datenbank angelegt, welcher anschließend das Rennen fährt. Dabei werden die verschiedensten Daten des Rennens aufgezeichnet und abgespeichert. Dann werden die Daten ins Cluster hochgeladen und anschließend analysiert und visualisiert.

6.2.3 Netzwerk und Kommunikation des Demonstrators

Für die Verbindung von Cluster, Raspberry Pi und den ESP32 wird ein LTE Router eingesetzt. Zum leichteren Wiederfinden der Komponenten werden statische IP Adressen verwendet. Die Kommunikation findet entweder Seriell oder über Websockets statt. Die ESPs und der RPI z.B kommunizieren mithilfe von Websockets und des Protobuf Protokolls.

6.2.4 Aufbau der Bahn

Sobald die Tische montiert sind, kann die Carrera-Rennbahn gemäß Abbildung 16 aufgebaut werden. Beim Einbau der Zwischenzeitmesser und der Carrera CU in die Bahn ist die Fahrtrichtung zu beachten. Außerdem scheinen sich die Zwischenzeitmesser die von ihnen gelieferte Zwischenzeit zu merken. Daher haben wir jedes Zwischenzeitstück mit Anmerkungen versehen, damit wir die Stücke für Zwischenzeit 1 und Zwischenzeit 2

identifizieren können. Für die Bahn wurde die Strecke so ausgelegt, dass beide Spuren gleich lang sind, somit können die Fahrergebnisse miteinander verglichen werden.

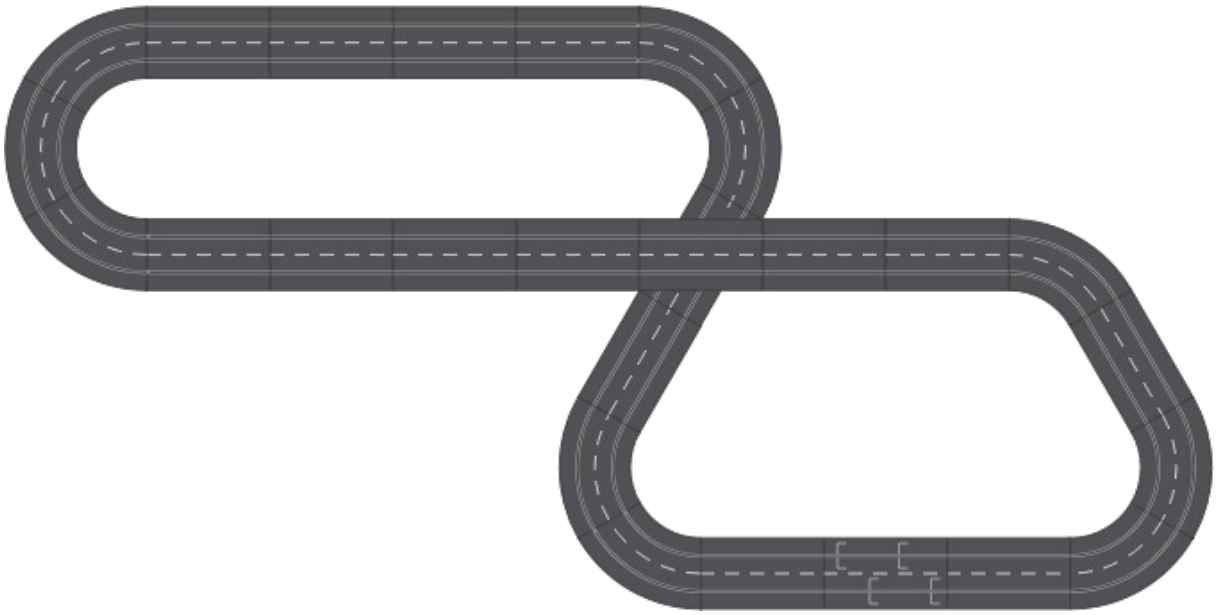

Abbildung 16: Streckenplan

6.2.5 Hardware Aufbau

Abbildung 17: Hardware Aufbau

Abbildung 18: Hardware Aufbau von unten

6.2.6 Auslesen der Streckeninformationen

Das Streckenprotokoll wird verwendet, um Befehle von den Controllern an die Autos zu übermitteln, wie zum Beispiel die aktuelle Drossel Einstellung. Wir haben die Carrera CU so modifiziert, dass sie den Zugriff auf das Streckensignal über einen Anschluss ermöglicht, der neben dem Anschluss für den Rundenzähler hinzugefügt wurde, der sich links von dem Anschluss für die serielle PC-Einheit befindet.

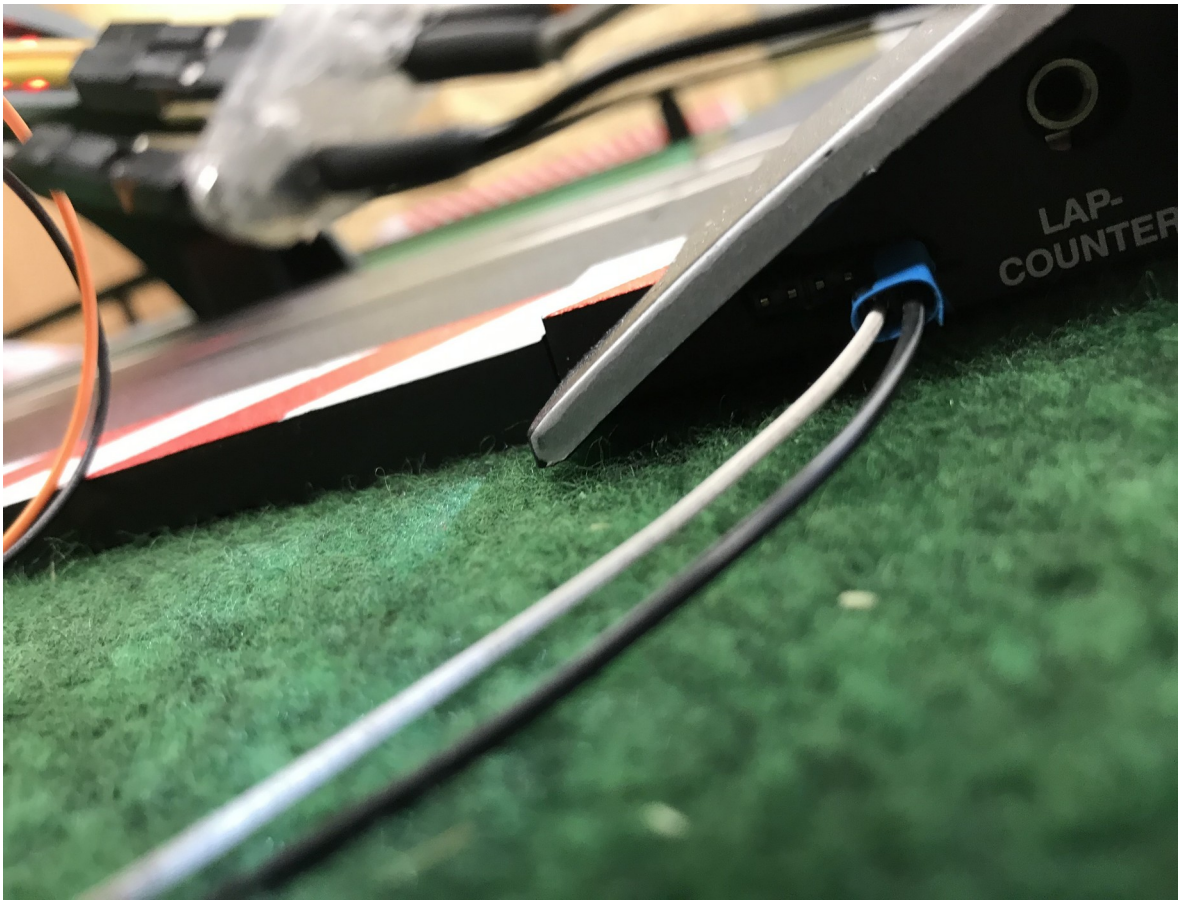

Abbildung 19: Carrera CU mit Bahnprotokollanschluss

Der ESP32, der das Signal dekodiert, muss mit dem entsprechen Zephyr Code geflasht sein und mit einem selbst angefertigten Kabel, bestehend aus zwei Drähten, an den Bahnprotokollanschluss angeschlossen werden. Achten Sie beim Einstecken darauf, dass die blaue Farbe des Kabels mit der blauen Farbe der CU übereinstimmt.

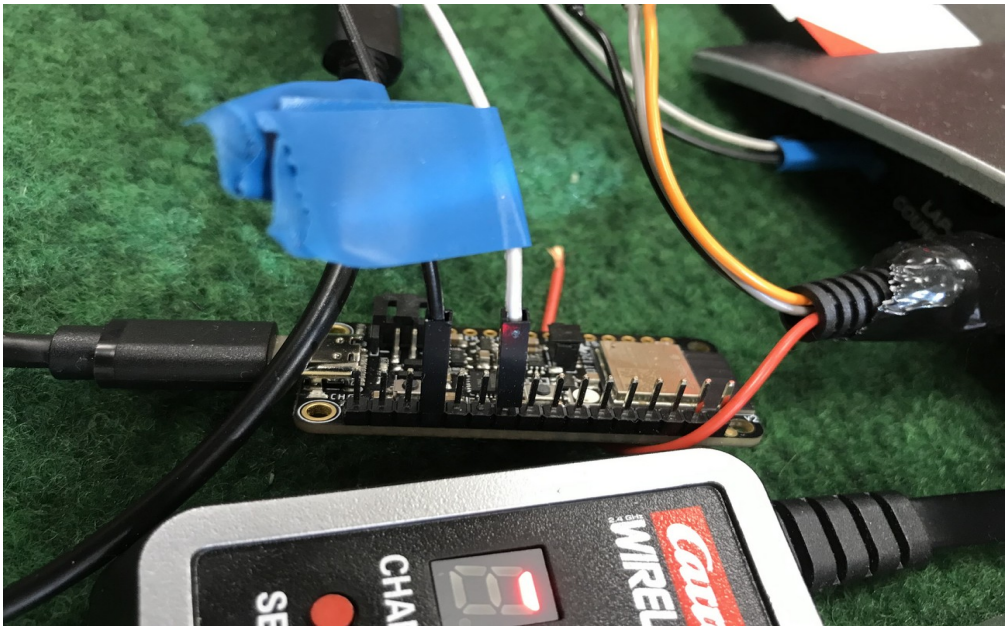
Abbildung 20: Carrera CU mit ESP32 und Rundenzähleranschluss

An der Runden-Zähler Buchse wurde ein LC231X TTL zu USB Konverter angeschlossen. Der Anschluss ist im Bild zu sehen neben dem ESP32. Der TTL Konverter ist dann per USB an den RPI angeschlossen.

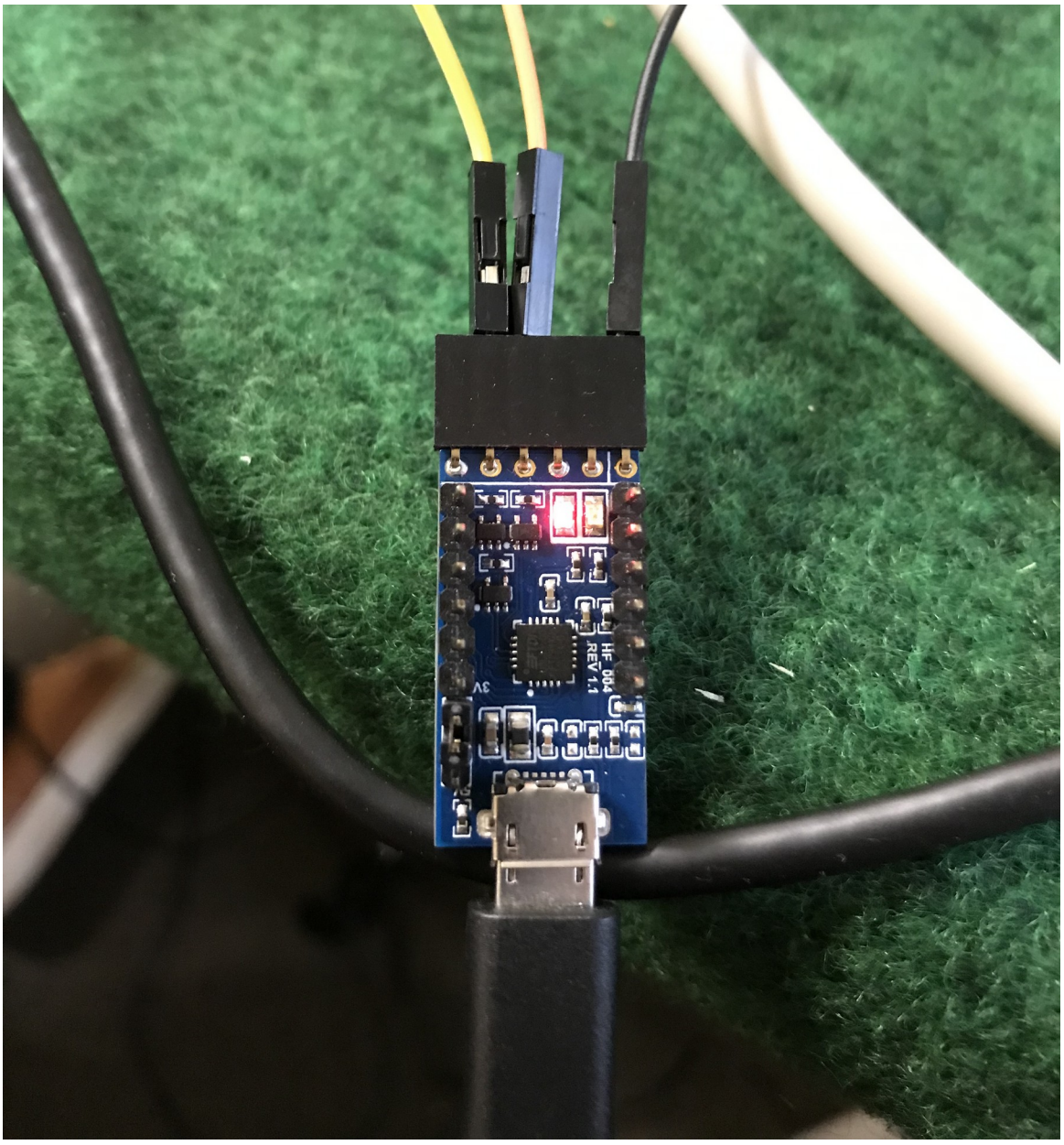

Abbildung 21: FTDI

6.2.7 Modifikationen an den Carrera Autos

Bei den Autos wurden die Heckmagnete entfernt damit sie driften und des Weiteren leichter aus der Spur raus fliegen können. Die Autos enthalten ein ESP und einen Sensor, der Beschleunigung, Magnetkraft und Drehung messen kann. Das ESP wird von einer Batterie gespeist. Er kann durch Anschluss eines Netzteils an die USB-C-Buchse des ESP aufgeladen werden. Der ESP muss mit dem entsprechen Zephyr Code geflasht sein. Idealerweise sollte der ESP nach dem RPI eingeschaltet werden, damit er sich mit dem laufenden Websocket-Server verbinden und die gemessenen Sensordaten senden kann.

Abbildung 22: Carrera Auto mit verbauter Sensorik

6.2.8 Sensorgenauigkeit des LSM9ds1

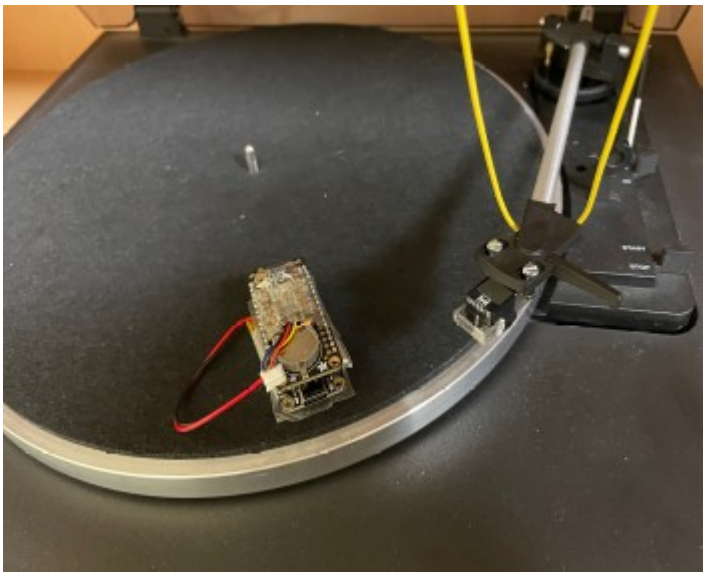
Abbildung 23: Plattenspieler mit ESP32 und LSM9ds1 Sensor

Um die Qualität der Daten, welche mithilfe des LSM9ds1 Sensors erfasst werden, zu evaluieren wurde der ESP32 mit Akku und LSM9ds1 Sensor auf einem Plattenspieler befestigt. Es wurden Daten mit zwei Drehgeschwindigkeiten, 33 RPM und 45 RPM, und mit verschiedenen Radien aufgenommen. Der Vorteil des Plattenspielers gegenüber der späteren Einsatzumgebung im Carrera Auto auf der Carrera-Rennbahn ist, dass beim Plattenspieler eine wesentlich störungsfreiere Aufzeichnungsumgebung gegeben ist bei der Aufzeichnung. Der LSM9ds1 erfährt eine konstante und berechenbare Beschleunigung über einer längere Zeit. Mit diesem Prototypen wurde eine Abtastung der Sensorwerte von 120Hz erreicht. In Abbildung 24 sieht man die Beschleunigungswerte einer Versuchsreihe für 33 RPM und 45 RPM.

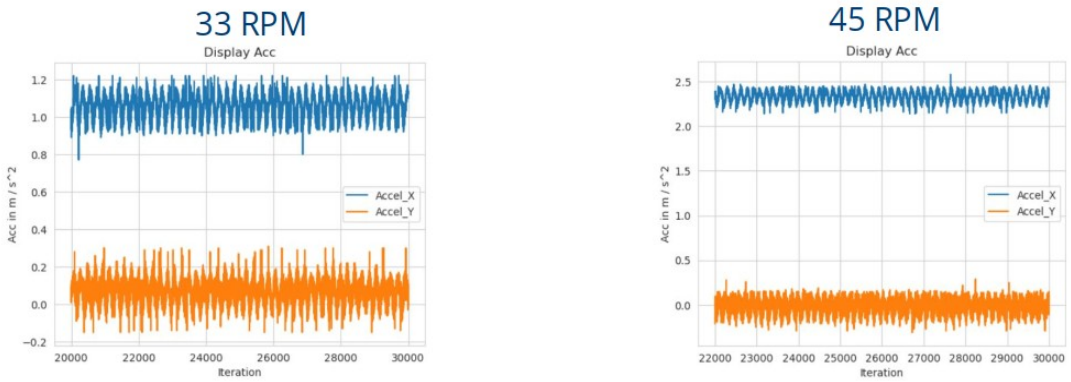

Abbildung 24: Beschleunigungsdiagramme des LSM9ds1 Sensors

Bestimmung der Genauigkeit anhand der bekannten "echten" Beschleunigung. Wie man sehen kann befindet sich ein erhebliches Grundrauschen auf den Beschleunigungsdaten. Was sie somit für die initial geplante Drifterkennung nicht nutzbar macht.

6.2.9 Sensorbefestigung

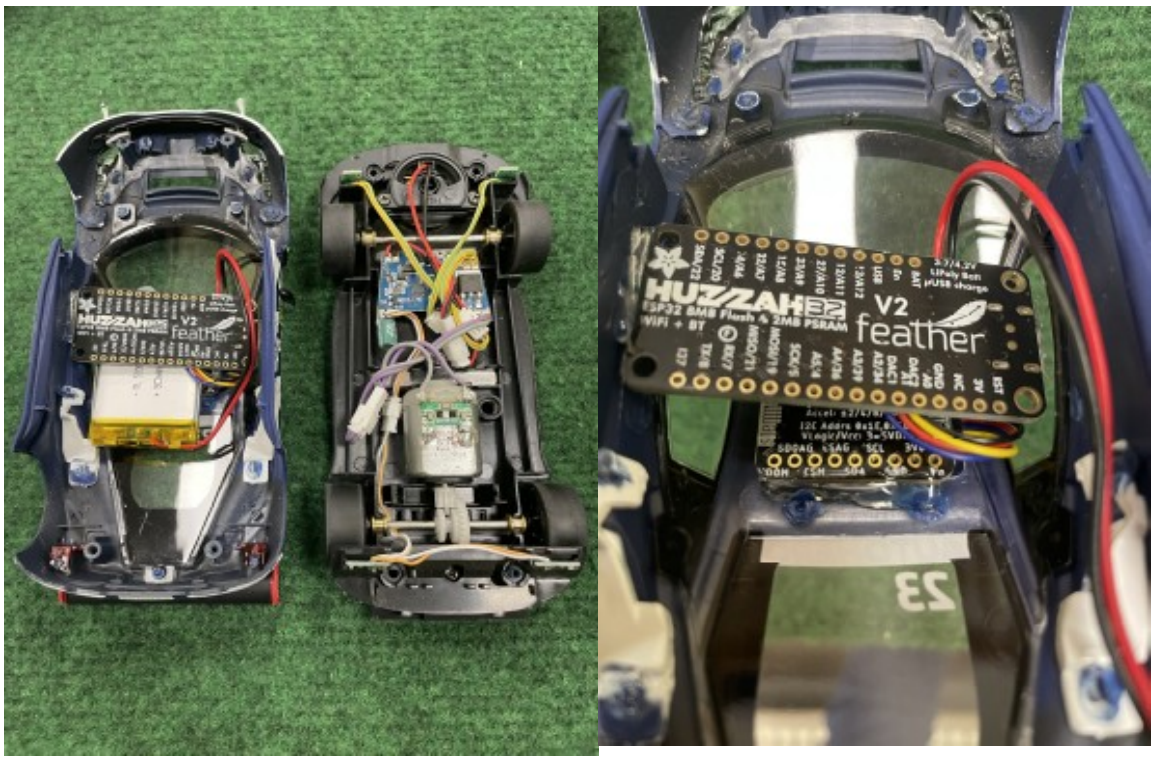

Abbildung 25: Geöffnetes Carrera Auto mit verbauter Sensorik

6.2.10 Videodaten für die Fahrzeugverfolgung

Abbildung 26: Lackierte Carrera Autos

Die Rennstrecke wird während der Rennen mit opencv aufgezeichnet, und die Positionen der Fahrzeuge werden anhand der aufgezeichneten Videos berechnet. Diese Aufgabe wird von einem separaten Windows-Computer übernommen, da die USB-Kamerakonfiguration dort einfacher funktioniert. Es wird eine 4k Angetube 914F Webcam verwendet da mit deren Weitwinkel die ganze Strecke erfasst werden konnte. Die Meldungen über den Start und das Ende eines Rennens empfängt der Laptop ebenfalls über Websockets. Der Dienst packt die resultierenden Daten in ein tar-Archiv, welches man dann hoch lädt. Die Positionen der Fahrzeuge werden in einem Nachtbearbeitungsschritt aus der Aufnahme ermittelt, da dies einige Rechenressourcen erfordert welche die Aufnahme der Strecke behindern könnten. Um die Autos erfolgreich erkennen zu können wurden diese in blau und orange angesprüht, da die Originale mit zu viel Werbung gestaltet waren, was das Tracken nicht möglich machte.

Abbildung 27: Videoaufzeichnung mit Positionserkennung

6.2.11 Renn-Manager

Das Open Source Projekt Open Lap wird als Race Manager UI verwendet, welches auf dem RPI läuft. Open Lap ist über Websockets (Serielles Backend) mit der Carrera CU verbunden. Bei dieser können die Fahrer und Runden-zahl eingestellt und das Rennen gestartet werden.

6.2.12 Datenaufzeichnung

Die Aufzeichnung der Renndaten beginnt automatisch mit dem Start des Renncountdowns und endet entweder:

- sobald alle Fahrer die Zielrundenzahl erreicht haben, wenn "Alle Runden beenden" aktiviert ist
- sobald der erste Fahrer die Zielrundenzahl erreicht hat, wenn "Alle Runden beenden" deaktiviert ist
- wenn ein Fahrer einen Fehlstart versucht hat
- wenn man manuell auf die Startampel in OpenLap klickt, damit alle Ampeln wieder aufleuchten

Es werden folgende Daten aufgezeichnet beziehungsweise Dateien angelegt und beschrieben:

- eine JSON-Datei mit den Metadaten des Rennens, z. B. den Namen der Fahrer und der Anzahl der zu fahrenden Runden({start_time}_metadata.json)
- eine CSV-Datei mit den Runden- und Zwischenzeiten({start_time}_times.csv)
- eine CSV-Datei mit den Streckeninformationen, z. B. Geschwindigkeitsdaten des Reglers({start_time}_track.csv)
- CSV-Dateien mit Beschleunigungs-, Kreisel- und Magnetsensordaten({start_time}_esp32_{esp_device_id}_lsm9ds1-acc-gyro@6b.csv, {start_time}_esp32_{esp_device_id}_lsm9ds1-magn@1e.csv)

Sobald das Rennen beendet ist, werden die Renndaten aus dem Unterordner in ein tar-Archiv gepackt. Dieses Archiv wird auf der gleichen Ebene wie die Unterordner des Rennens gespeichert. Sein Name besteht aus dem Namen des Unterordners und dem Suffix ".tgz". Die Daten müssen dann anschließend über einen Webservice ins Cluster geladen werden.

6.2.13 Datenanalyse und Visualisierung

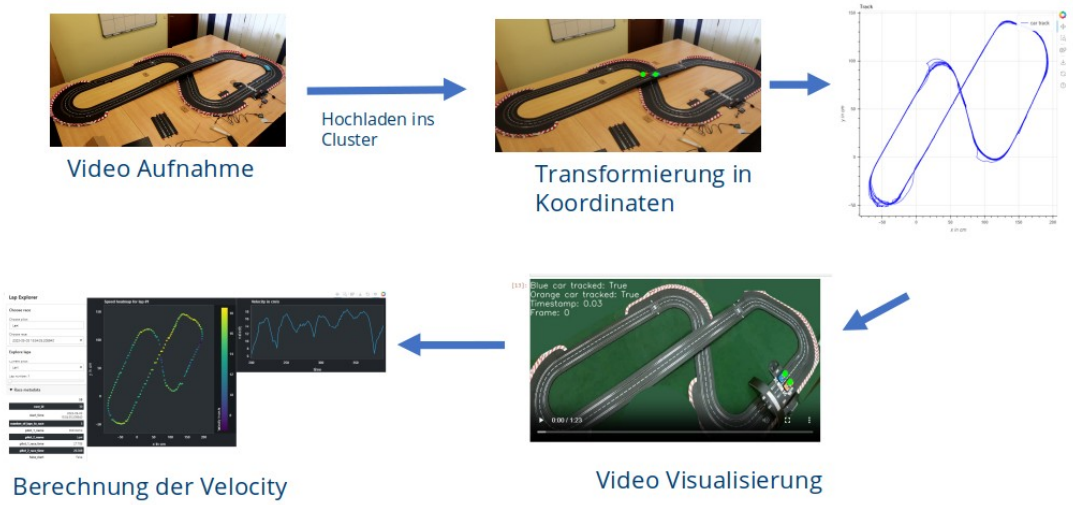

Abbildung 28: Ablauf der Videoaufzeichnung und Analyse

1. Es wurde ein Notebook geschrieben welches die Fahrzeugpositionen aus dem Video berechnet, um reale Koordinaten aus dem aufgenommenen Video zu erhalten. Das Notebook konvertiert die gespeicherten Fahrzeugpositionsdaten aus "tracked_data.csv" in 2D-Koordinaten und speichert die Daten in "transformed_data.csv". Was nach weiteren Aufbereitungschritten wie folgt aussieht:

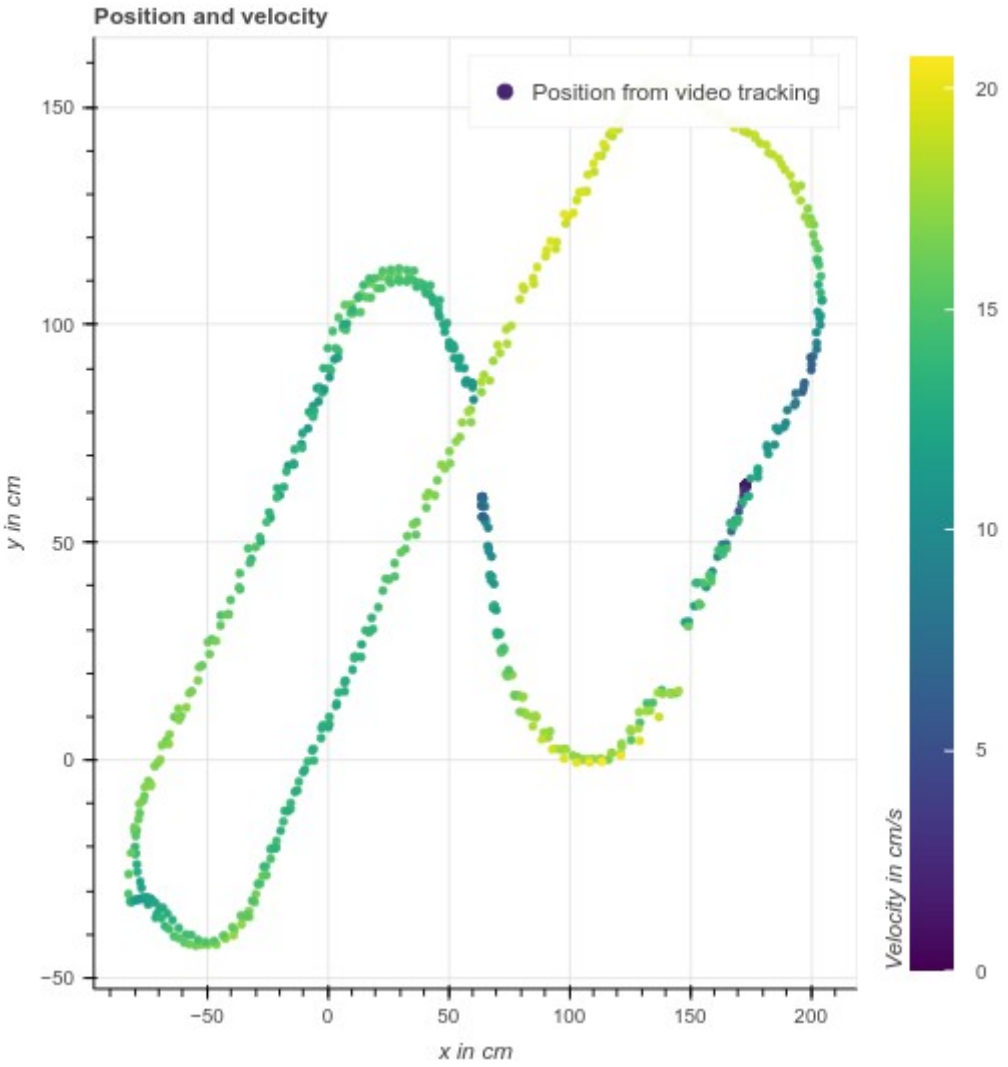

Abbildung 29: Carrera Auto Position und Velocity

2. Es wurde ein Notebook geschrieben um zu sehen, wo auf der Strecke das Auto welche Geschwindigkeit hatte. Es verwendet die gespeicherten Daten "transformed_data.csv", "timestamps_only.csv" und "...times.csv", um die Geschwindigkeit mit einem Kalman-Filter aus den Positionen zu berechnen. Des Weiteren wird die Allan variance verwendet um ein Stabileres Signal zur Berechnung der Geschwindigkeit zu haben.

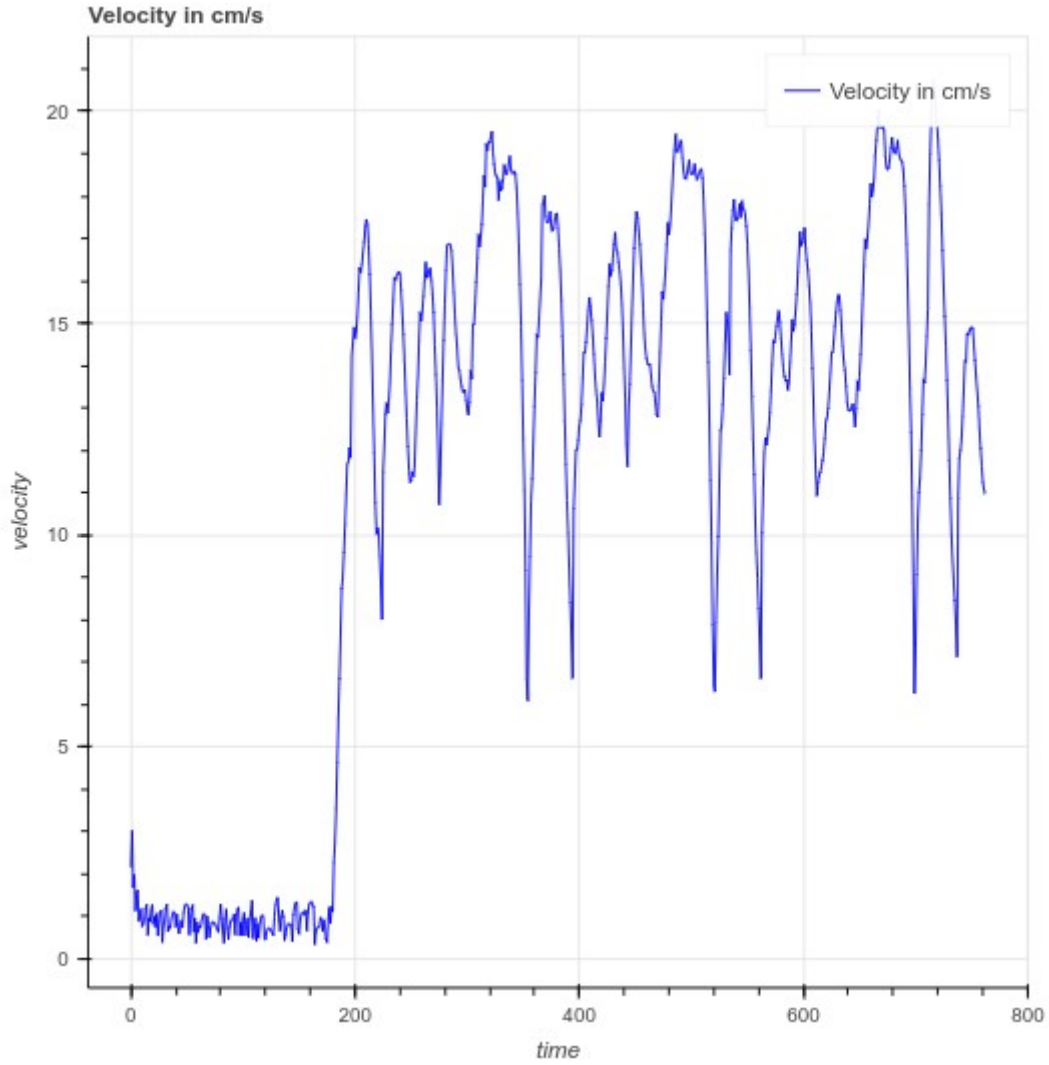

Abbildung 30: Carrera Auto Geschwindigkeit Diagramm

3. Es wurde ein Notebook geschrieben welches die Zusammenstöße oder Abflüge auf der Strecke darstellt. Es verwendet die Daten des LSM9ds1 Gyrometers, welches die Rotation misst, um Zusammenstöße des Autos zu erkennen. Zusammenstöße erzeugen eine starke Rotationsspitze in den Gyrometerdaten. Das Notebook verwendet die Daten des orangefarbenen Autos eines bestimmten Rennens. Im folgenden sind die Gyrodaten zu sehen:

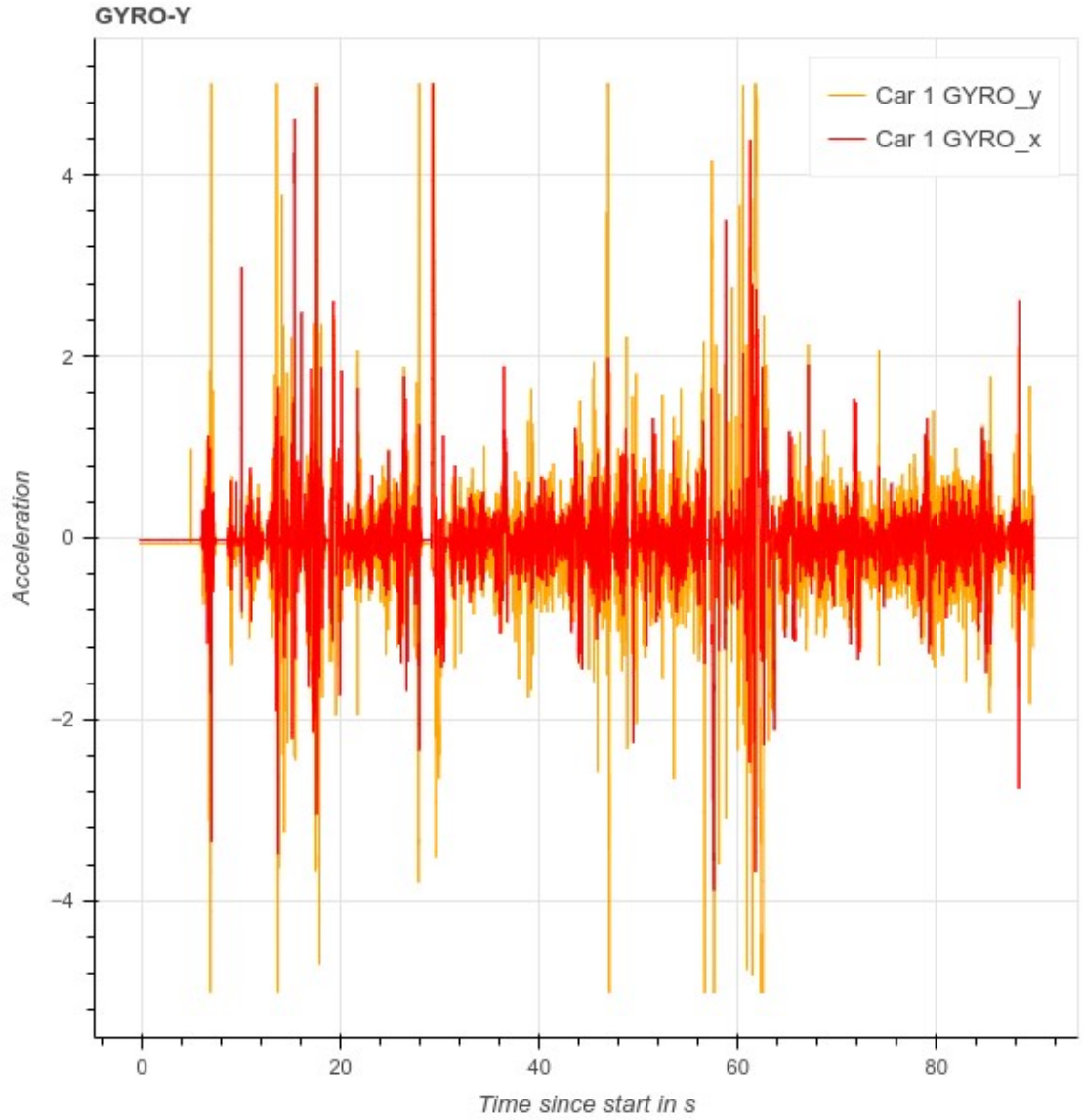

Abbildung 31: Gyro Diagramm eines Rennens

Im Folgenden ein Diagramm der Zusammenstöße und Abflüge:

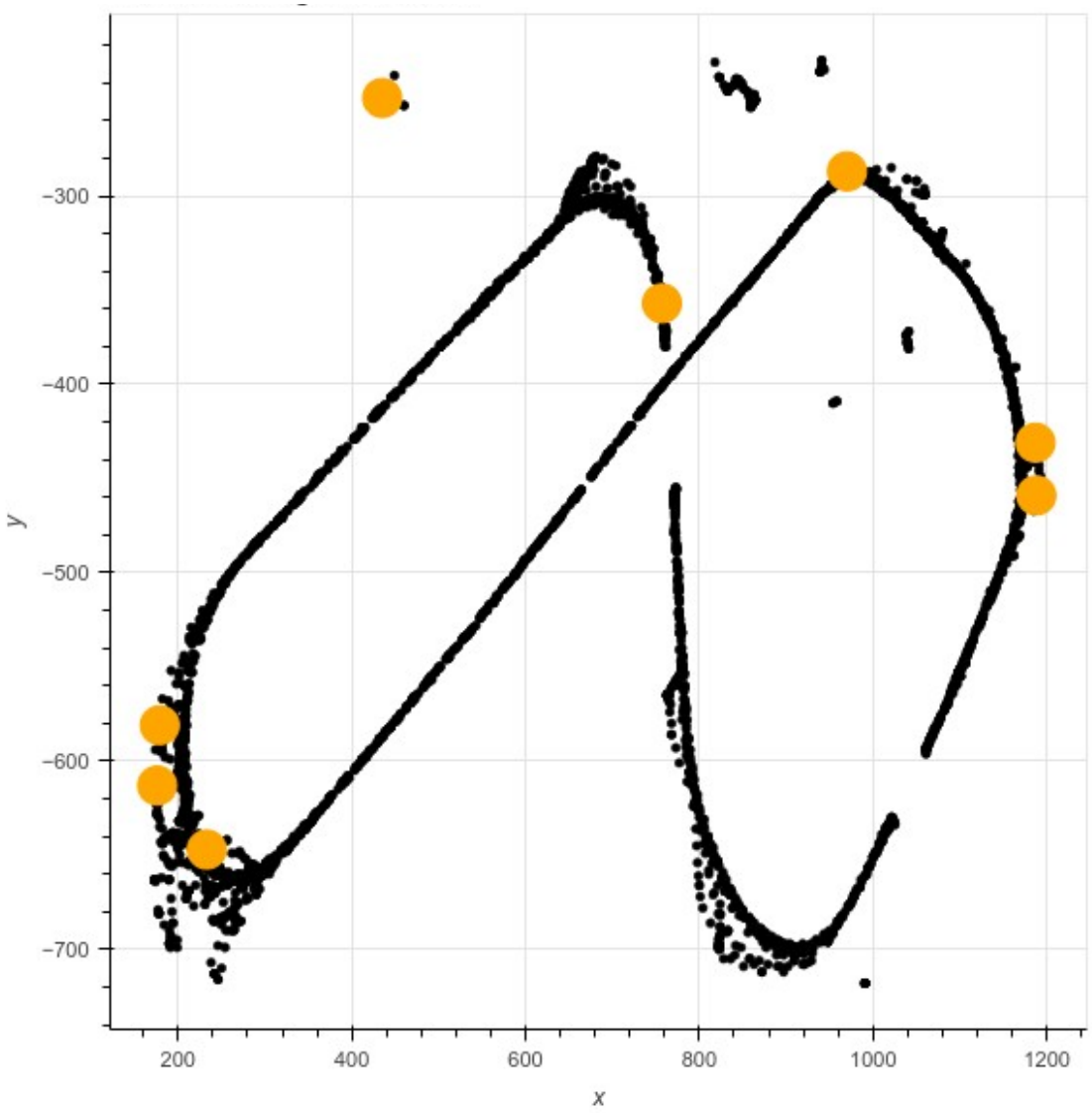

Abbildung 32: Carrera Auto Positions Diagramm mit Abflugpunkten

4. Das Notebook bokeh_holoviews_comparison wird verwendet um die Daten eines ausgewählten Rennens in Bezug auf die Rennzeit zu anzuzeigen: schnellste Runde, schnellster Splitzeit, durchschnittliche Zeit pro Runde, notwendige/mögliche Verbesserung zu visualisieren. Es vergleicht das Rennen mit anderen Rennen im Allgemeinen und auch mit anderen Rennen des gewählten Piloten.

	lap_number	split_1	split_2	split_3	lap_time
0	1	2.016	2.761	1.344	6.121
1	3	1.513	2.460	1.258	5.231
2	4	1.471	1.972	1.207	4.650
3	5	2.952	3.050	1.259	7.261
4	7	1.796	2.169	2.751	6.716
5	8	1.607	3.062	1.200	5.869
6	9	1.565	2.084	1.774	5.423
7	10	3.672	2.088	1.431	7.191

Abbildung 33: Splitzeiten und schnellste Runde

Fastest 3 laps:

lap_number	split_1	split_2	split_3	lap_time
4	1.471	1.972	1.207	4.650
3	1.513	2.460	1.258	5.231
9	1.565	2.084	1.774	5.423

```
The mean lap time was 6.06s.
The fastest lap time, 4.65s,         was 1.41s faster than that.

Fastest split 1: 1.47s
Fastest split 2: 1.97s
Fastest split 3: 1.20s
The fastest possible lap time (combining fastest splits) would have been         4.64s.
This means there is at least a possible improvement of         0.01s.

The fastest lap time achieved by any player is         3.32s.
The necessary improvement to reach this time would be         1.33s.
```

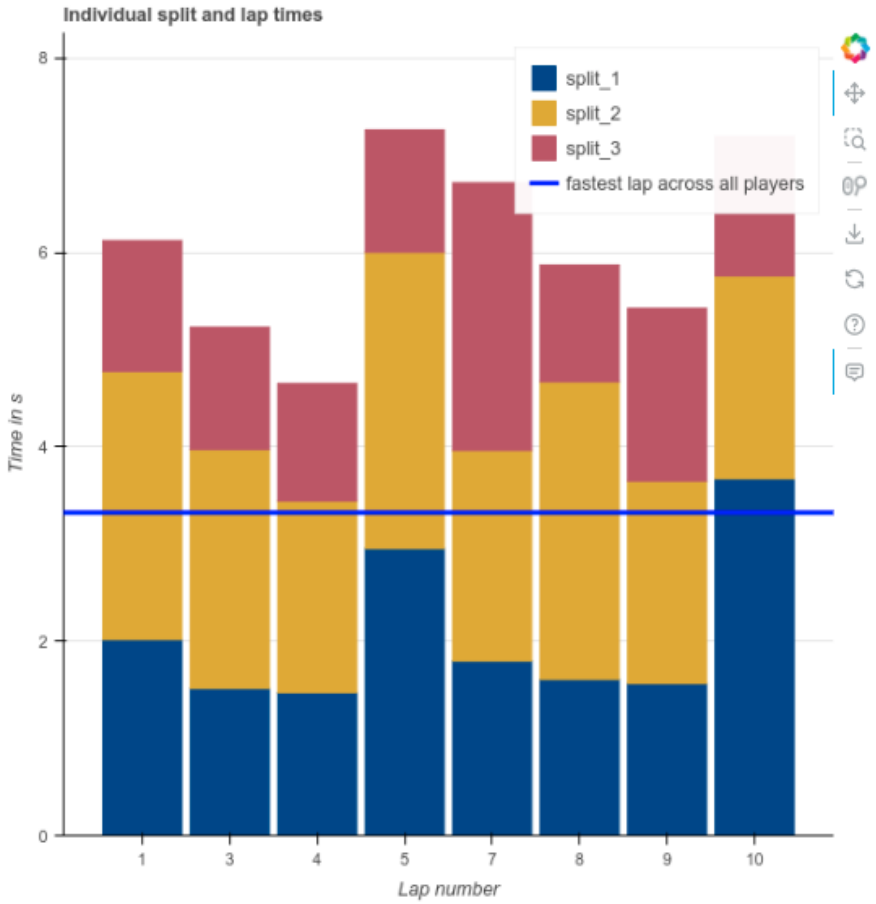

Abbildung 34: Runden- und Splitzeitanalyse

6.2.14 Automatische Carrera Autosteuerung

CarController ist ein Server, der auf einem RPI läuft, der über eine kabelgebunde Schnittstelle mit der Carrera CU verbunden ist.

Um den kabelgebundenen Carrera Kontroller zu verwenden wurde der Kontroller geöffnet und analysiert das Ergebnis war das er einfach einen Spannungswert zwischen 0V und 2,3V liefert. Somit war es möglich das Carrera Auto mithilfe eines RPI, welcher mit einem Adafruit MCP4725 DAC über I2C verbundenen ist und einem Spannungsteiler der die Spannung auf maximal 2,3V begrenzt, zu steuern. Der Spannungsteiler wird dann in der Zukunft gegen einen Step Down ausgetauscht. Um den DAC zu steuern, wird das entsprechende Python-Paket von Adafruit verwendet. Dieses Paket benötigt zusätzlich zu seinen Python-Abhängigkeiten einige Systemweite Konfigurationen und Pakete.

6.3 Elektroauto OBD Demonstrator

Zunächst wurde ein OBD Auslesegerät Vgate iCar Pro angeschafft um erste Daten zu aggregieren und zu visualisieren. Anschließend wurde ein OSS OBD Auslesegerät OVMS V3 angeschafft da die Abtastrate des OBD Auslesegerät Vgate iCar Pro mit einer Sekunde nicht ausreichen war.

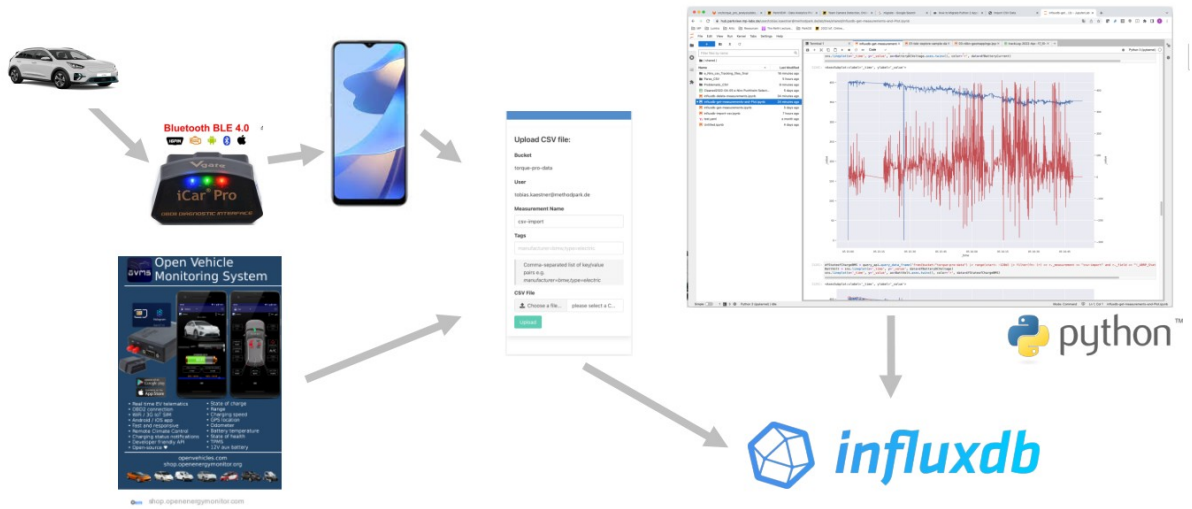

Abbildung 35: Elektroauto Demonstrator Datenaufzeichnung und Analyse

6.3.1 Torque Pro Analyse

Mit der App Torque Pro wurden Daten mit einem elektrischen KIA Niro gesammelt.

6.3.2 OVMS Datenaggregation

Im OVMS kann über die Weboberfläche des OVMS der jeweilige Autotyp konfiguriert werden.

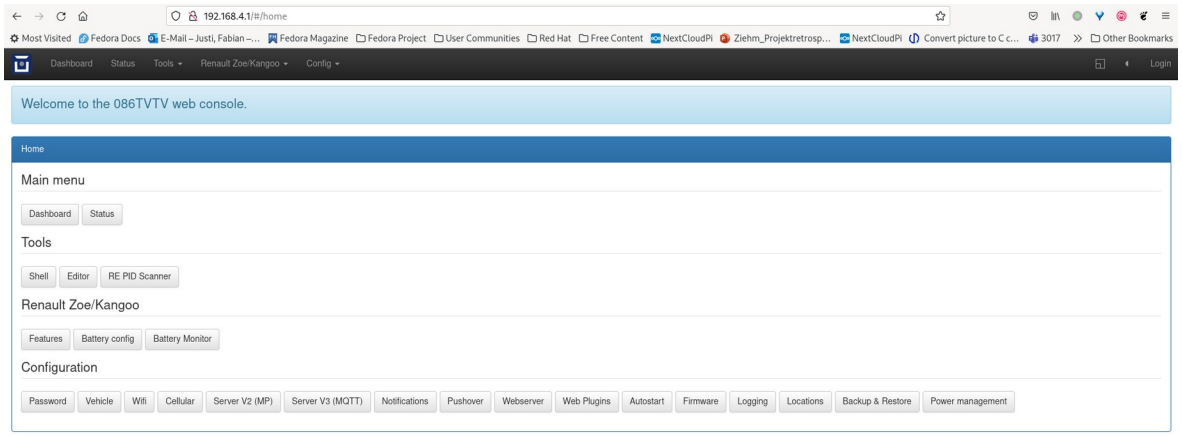

Abbildung 36: OVMS Startseite

Zum Datenaggregieren mithilfe des OVMS V3 wurde ein JS Programm geschrieben welches alle 500ms bestimmte Werte über den OBD port ausließt und in eine log Datei auf der SD Karte des OVMS schreibt da die übertragung über das LTE Netz zu verzögerungen bei der Aufzeichung führte. Um die Daten in der Log Datei mit einem korrekten Zeitstemple zu versehen wurde ein Programm geschrieben, welches beim Hochfahren des OVMS V3 LTE aktiviert und sich über NTP das aktuelle Datum und die Aktuelle Uhrzeit holt. Nachdem die Aktuelle Uhrzeit im System gesetzt wurde, wird das Datenaufzeichnungsprogramm ausgeführt.

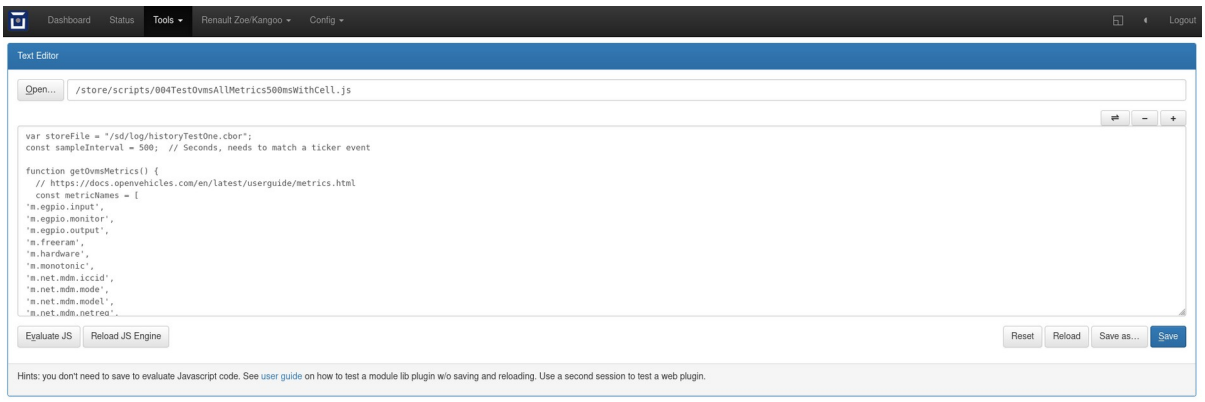

Abbildung 37: OVMS Web-Editor

6.3.3 Beschreibung der Demonstrators

Der Demonstrator enthält derzeit zwei interaktive Werkzeuge und fünf Diagramme. Darüber hinaus kann mit den Diagrammen durch Klicken, Ziehen und Zoomen interagiert werden.

Ein Diagramm stellt den Weg des Autos auf einer Weltkarte dar. Die Darstellung wird durch eine Heatmap eingefärbt, die entweder auf Geschwindigkeit in km/h oder Batterieleistung in kW eingestellt werden kann. Die Option "Batterieleistung" zeigt an, wie viel Energie die Batterie dem Auto zur Verfügung gestellt hat, entweder durch Beschleunigung oder durch Rückgewinnung von Energie beim Bremsen. Der Schieberegler für die Reichweite verschiebt zwei Punkte auf der Strecke, um den Weg zu veranschaulichen, den das Auto in diesem Zeitintervall zurückgelegt hat.

Das zweite Diagramm zeigt die Zellenspannungen im Zeitverlauf. Die Zellenspannung entspricht direkt der in der Zelle gespeicherten Energie. Die Grafik zeigt die Spannungen der stärksten Batteriezelle, der schwächsten Batteriezelle und den Durchschnitt aller Zellen. Große Abweichungen zwischen verschiedenen Zellen sind ein Zeichen für eine alternde Batterie und sind von großem Interesse, wenn die Batterie eines Elektrofahrzeugs weiterverkauft wird. Um 15:15 Uhr ist eine Spitze in der Zellenspannung zu sehen, was bedeutet, dass die Batterie zu diesem Zeitpunkt Energie rekuperiert hat.

Die dritte Grafik veranschaulicht die Höhe und die Geschwindigkeit über die Zeit hinweg. Beides sind wichtige Informationen über die Fahrt und können zur Überprüfung der Daten auf ihre Richtigkeit verwendet werden. Man beachte, dass das Auto in den letzten 5 Minuten Stillstand, wie man an der Geschwindigkeit von Null erkennen kann.

Das vierte Diagramm zeigt zwei verschiedene Messungen des Ladezustands des Akkus. Wir können uns die Situationen genauer ansehen, in denen der Akku schnell entladen war, zum Beispiel Von 16:20 Uhr bis 16:30 Uhr. Wir können entweder das Hover-Tool verwenden, um die entsprechenden Punkte auf dem ersten Diagramm zu finden, das den Pfad anzeigt, oder wir stellen den Bereichsschieberegler auf dieses Zeitintervall ein. Anhand der Geschwindigkeits-Heatmap können wir sehen, dass das Auto in diesem Intervall mit Höchstgeschwindigkeit fuhr.

Das letzte Diagramm stellt ein Histogramm dar, das angibt, wie viel Energie die Batterie an jedem Datenpunkt geliefert hat. Das Histogramm berücksichtigt nur die Punkte in dem durch den Schieberegler definierten Intervall. Wenn Sie den Schieberegler auf den vollen Bereich einstellen, beobachten Sie eine Spitze bei Null, die entfernt werden kann, indem Sie die letzten 5 Minuten mit dem Schieberegler ausschließen, als das Auto still stand.

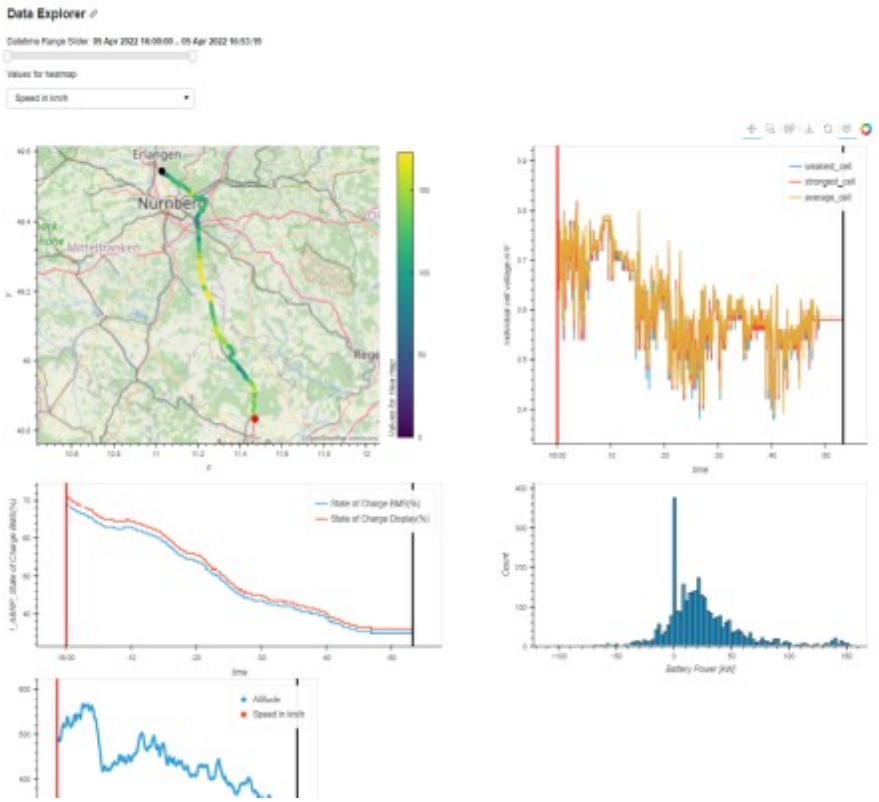

Abbildung 38: Digramme des Elektroauto Demonstrators

7 Evaluation

Im vorliegenden Kapitel wird die Umsetzung der Anforderungen, die im Kapitel 4 festgelegt wurden, in Bezug auf die Implementierungen dieser Arbeit bewertet. Zuerst wird die Erfüllung der funktionalen Anforderungen überprüft, bevor die nicht-funktionalen Anforderungen behandelt werden.

7.1 Funktionale Anforderungen

FA-1: Der Zugang zur OSS DAP muss einfach sein und erfordert lediglich einen Webbrowser, um darauf zuzugreifen.

Die Anforderung FA-1 wurde erfüllt.

FA-2: Die OSS DAP soll verwaltete OSS Werkzeuge verwenden, wodurch sie stets flexibel und auf dem neuesten Stand der Technik bleibt.

Die Anforderung FA-2wurde erfüllt.

FA-3: Es soll mithilfe von Werkzeugen die Überwachung und Verwaltung der verfügbaren und genutzten OSS-Pakete sichergestellt werden.

Die Anforderung FA-3 wurde erfüllt.

FA-4: Benutzer sollen die Möglichkeit haben, gemeinsam an denselben Datensätzen und Datenanalyse Notebooks zu arbeiten, wodurch eine reibungslose Zusammenarbeit ermöglicht wird.

Die Anforderung FA-4 wurde erfüllt.

FA-5: Die OSS DAP soll auf Cloud-Ressourcen zurückgreifen, wodurch ihre Leistung nicht durch die Kapazität der lokalen Hardware begrenzt ist.

Die Anforderung FA-5 wurde erfüllt.

FA-6: Die OSS DAP muss in der Lage sein, eine Vielzahl von Datentypen zu akzeptieren und zu analysieren. Hierzu gehören unter anderem Tabellenkalkulationen, Zeitreihen, Bilder sowie Audio- und Videodateien.

Die Anforderung FA-6 wurde erfüllt.

FA-7: Es soll ein gemeinsamer Speicher eingerichtet werden, der allen Benutzern den Zugriff auf Notebooks und Daten ermöglicht.

Die Anforderung FA-7 wurde erfüllt.

FA-8: Die OSS DAP muss die effiziente Bereitstellung und Verwaltung gemeinsam genutzter Rechenressourcen für Docker-Container gewährleisten, um eine optimale Auslastung und Leistungsfähigkeit sicherzustellen.

Die Anforderung FA-8 wurde erfüllt.

FA-9: Die Authentifizierung sollte mittels eines bestehenden Domänenaccounts möglich sein.

Die Anforderung FA-9 wurde erfüllt.

7.2 Nicht-funktionale Anforderungen

NFA-1: Die OSS DAP muss so gestaltet sein, dass sie innerhalb von AWS und lokalen Clusterlösungen zur Verfügung gestellten Umgebung aufgespielt und betrieben werden kann.

Die Anforderung NFA-1 wurde erfüllt.

NFA-2: Die OSS DAP sollte in der Lage sein, mit dem Wachstum der Daten und Benutzer zu skalieren, um eine effiziente Verarbeitung großer Datenmengen zu ermöglichen.

Die Anforderung NFA-2 wurde erfüllt.

NFA-3: Die OSS DAP sollte nahtlos mit anderen Systemen und Werkzeugen integrierbar sein, um Daten aus verschiedenen Quellen zu erfassen, zu verarbeiten und zu analysieren.

Die Anforderung NFA-3 wurde erfüllt.

NFA-4: Die OSS DAP sollte robuste Sicherheitsmechanismen bieten, um die Vertraulichkeit, Integrität und Verfügbarkeit der Daten zu gewährleisten und Datenschutzrichtlinien und -vorschriften einzuhalten.

Die Anforderung NFA-4 wurde erfüllt.

7.3 Funktionale Anforderungen der Demonstratoren

FA-1: Die Notebooks sollen eine Kombination aus Dokumentation und Python-Code enthalten.

Die Anforderung FA-1 wurde erfüllt.

FA-2: Der Datenfluss soll automatisiert erfolgen, beginnend von der Datenquelle bis zum Cluster.

Die Anforderung FA-2 wurde teilweise erfüllt.

FA-3: Der Carrera Demonstrator soll transportabel und mit annehmbaren Aufwand ab- und auf-baubar sein.

Die Anforderung FA-3 wurde erfüllt.

FA-4: Der jeweilige Demonstrator muss fähig sein Daten der zugrunde liegenden Demonstratorbasis zu aggregieren.

Die Anforderung FA-4 wurde erfüllt.

FA-5: Der jeweilige Demonstrator muss fähig sein die aggregierten Daten zu verarbeiten und zu visualisieren.

Die Anforderung FA-5 wurde erfüllt.

7.4 Nicht Funktionale Anforderungen der Demonstratoren

NFA-1: Der Carrera Demonstrator muss so gestaltet sein, dass er innerhalb einer lokalen Clusterlösungen zur Verfügung gestellten Umgebung aufgespielt und betrieben werden kann.

Die Anforderung NFA-1 wurde erfüllt.

NFA-2: Der Carrera Demonstrator muss so gestaltet sein, dass er ohne großen Werkzeugeinsatz transportiert werden kann.

Die Anforderung NFA-2 wurde erfüllt.

7.5 OSS DAP im Vergleich

7.5.1 OSS DAP vs. Power BI, Qlik oder Tableau

Werkzeuge wie Power BI sind auf Business Intelligence und interaktives Reporting mit Datenvisualisierung zugeschnitten. Außerdem macht Power BI keine Datenbereinigung. Die OSS DAP konzentriert sich auf Flexibilität und einfachen Zugang. Es bietet ein unbegrenztes Toolkit auf dem neuesten Stand der Technik, das auf OSS-Werkzeugen basiert. Daher kann die OSS DAP all diese Lösungen und noch mehr liefern.

7.5.2 OSS DAP vs. Anaconda Cloud oder Google Colab

Anaconda Cloud und Google Colab bieten beide eine Cloud-basierte Umgebung zur Ausführung von Python-Code für Data Science mit Funktionen zur Zusammenarbeit. Die OSS DAP basiert ebenfalls auf einer umfassenden Plattform für die interaktive Entwicklung von Analyse- und Machine-Learning-Lösungen. Darüber hinaus ist die OSS DAP so konzipiert, dass es schnell auf Ihrer eigenen Cloud- oder On-Premise-Infrastruktur bereitgestellt werden kann, so dass Kosten und Informationsbesitz völlig transparent sind.

7.5.3 OSS DAP vs. Data iku

Die OSS DAP stellt ähnliche Funktionalitäten bereit kann aber da sie komplett auf Open Source basiert, ist sie transparenter und leichter erweiterbar und kann angepasst werden.

7.5.4 OSS DAP vs. Diadem

Das viel in der Automobilindustrie eingesetzte CSS Werkzeug Diadem ist spezialisiert auf Echtzeit -analyse und -visualisierung von technischen Anwendungen mit physikalischen Daten. Im Gegensatz dazu bietet eine moderne DAP basierend auf OSS eine anpassbare Konfiguration und Funktionalität, außerdem können die verschiedensten Datenquellen analysiert werden.

7.5.5 OSS DAP vs. Excel

Excel ist ein hervorragendes Datenanalysewerkzeug mit einer sehr großen Nutzerzahl welches zur Analyse und Datenverwaltung von nicht zu großen Datenmengen genutzt wird beziehungsweise genutzt werden sollte. Mit Excel kann man leicht simple Analysen und Visualisierungen machen, jedoch kommt es zu Performance- und Übersichtlichkeitsproblemen bei zu großen Datenmengen und bei komplexen Analysen. Dem gegenüber kann eine moderne DAP gut mit großen Datenmengen von den unterschiedlichsten Formaten und Erscheinungsformen, wie z.B. Video Daten umgehen. Darauf können dann komplexe Analysen gemacht und die Ergebnisse ansprechend dargestellt werden, wie man anhand des Carrera Demonstrator gesehen hat.

8 Fazit

Das Ziel dieser Arbeit war es eine OSS DAP weiterzuentwickeln und weiterzubringen.

Das Ergenis dieser Arbeit sind zwei Demonstratoren, die bei Einzelpersonen und bei Messen sehr gut angekommen sind und die Vorteile und Möglichkeiten einer OSS DAP gut vermitteln und aufzeigen konnten. Dabei wurde die automatisierte Datenaufnahme und Analyse der neuen aktualisierten Daten veranschaulicht. Vor allem brachte der Carrera-Rennbahn Demonstrator einen hervorragenden Mehrwert durch den Spaßfaktor bei den Interessenten und das Aufzeigen, dass die verschiedensten Datenquellen, insbesondere Zeitreihen, Beschleunigungs- und Gyrosensordaten sowie Bild- und Videodateien, relativ leicht und performant zur Analyse genutzt werden konnten.

Appendix A Beispiel Notebook

Crash detection

In this notebook, we detect car crashes from the data collected by the gyrometer. We make the reasonable assumptions that crashes cause extremely fast turns in some dimension, so they should be visible as spikes in the gyrometer data.

Imports

[1]:
```
import ast

import numpy as np
import pandas as pd
from bokeh.models import ColumnDataSource, HoverTool
from bokeh.plotting import figure, output_notebook, show
from IPython.display import Video

output_notebook()
```

BokehJS 3.4.1 successfully loaded.

Configure

[2]:
```
acc_gyro_path = (
    "test_data/2023-05-10T10_47_17.604365_esp32_e89f6d23fee4_lsm9ds1-acc-gyro@6b.csv"
)
video_path = "test_data/masked_video.mp4"
positions_path = "test_data/tracked_positions.csv"
video_frames_per_second = 30
# timestamp offset has been eyeballed, check the first frame of the video you are analyzing
timestamp_offset_video = 0.06
```

Load data

[3]:
```
acc_gyro_df = pd.read_csv(acc_gyro_path)
positions_df = pd.read_csv(positions_path)
```

[4]:
```
acc_gyro_df["timeSinceStart[s]"] = acc_gyro_df["timeSinceStart[µs]"] / 10**6
```

[5]:
```
acc_gyro_df
```

[5]:

	timeSinceStart[µs]	clientTimestamp[µs]	ACCEL_X[m/s^2]	ACCEL_Y[m/s^2]	ACCEL_Z[m/s^2]	GYRO_X[1/s]	GYRO_Y[1/s]	GYRO_Z[1/s]	timeSinceStart[s]
0	-269349	6309315925	-0.091581	0.390865	9.711178	-0.027641	-0.063682	0.003970	-0.269349
1	-259439	6309325835	-0.153233	0.357944	9.740508	-0.027794	-0.062919	0.003970	-0.259439
2	-249063	6309336211	-0.122108	0.369915	9.772831	-0.023518	-0.061086	0.003665	-0.249063
3	-239146	6309346128	-0.145452	0.381886	9.787196	-0.027488	-0.061391	0.004886	-0.239146
4	-229162	6309356112	-0.122706	0.386675	9.735719	-0.027336	-0.060933	0.003970	-0.229162
...	...	...	...	...	...	...	...	...	...
8735	89701646	6399286920	1.465296	-0.726662	2.620533	-0.255036	0.297644	-0.008093	89.701646
8736	89711646	6399296920	15.608277	-2.636096	14.049604	-0.570548	-0.189826	0.011759	89.711646
8737	89721646	6399306920	-15.840521	4.776578	14.273469	-0.190284	-0.003970	0.127059	89.721646
8738	89731654	6399316928	-6.778191	4.230084	11.301575	0.020005	-0.142637	-0.037415	89.731654
8739	89741650	6399326924	-4.926220	7.341445	19.154797	0.470366	-1.206764	-0.012522	89.741650

8740 rows × 9 columns

Prepare data from position tracking

The data is saved in tracked_positions.csv as a tuple written to a column. If we parse this, we get a string "(x, y)" instead of a tuple.

[6]:
```
tupled = positions_df["position_orange_car"].map(ast.literal_eval).apply(pd.Series)
positions_df["position_x_orange_car"] = tupled[0]
# - because of computer graphics convention
positions_df["position_y_orange_car"] = -tupled[1]

tupled = positions_df["position_blue_car"].map(ast.literal_eval).apply(pd.Series)
positions_df["position_x_blue_car"] = tupled[0]
# - because of computer graphics convention
positions_df["position_y_blue_car"] = -tupled[1]
```

Plot acceleration data in z-coordinate

[15]:
```
source1 = ColumnDataSource(data=acc_gyro_df)

p1 = figure(
    title="GYRO-Y",
    x_axis_label="Time since start in s",
    y_axis_label="Acceleration",
)

p1.line(
    x="timeSinceStart[s]",
    y="GYRO_Y[1/s]",
    legend_label="Car 1 GYRO_y",
    color="orange",
    source=source1,
)

p1.line(
    x="timeSinceStart[s]",
    y="GYRO_X[1/s]",
    legend_label="Car 1 GYRO_x",
    color="blue",
    source=source1,
)

show(p1)red
```

Crash detection

[9]:
```
def detect_crash(gyro_value: float):
    # threshold is the result of eye balling
    crash_threshold = 3.0

    return abs(gyro_value) > crash_threshold


def find_crashes_from_gyro_data(df):
    # using GYRO_Y is the result of eyeballing
    crash_list = df["GYRO_Y[1/s]"].apply(detect_crash, 1)
    return crash_list
```

[*]:
```
acc_gyro_df["car_has_crashed"] = find_crashes_from_gyro_data(acc_gyro_df)
```

[11]:
```
def get_crash_events(df):
    # This function gets a list of individual crash events
    # We count a detected crash as a new crash
    # when the last detected crash happened more than crash_delimit_size timesteps ago
    # One timestep is about 10 ms
    # It returns a list of indices where the crash events started
    crash_delimit_size = 100
    crash_list = df.index[df["car_has_crashed"]]
    index_differences = [j - i for i, j in zip(crash_list[:-1], crash_list[1:])]

    crash_event_list = []
    # The first crash timestamp is always a valid new crash
    if not crash_list.empty:
        crash_event_list.append(crash_list[0])

    # The ith index difference tells us whether the i+1th timestep is a new crash
    for i in range(len(index_differences)):
        if index_differences[i] > crash_delimit_size:
            crash_event_list.append(crash_list[i + 1])

    return crash_event_list
```

Literaturverzeichnis

Breitner, J. (2019). *YAML-Referenz.* Das tttool-Buch. https://tttool.readthedocs.io/de/latest/yaml-referenz.html.

Harbor Authors. (2024). *Harbor 2.2 Documentation*. Goharbor. https://goharbor.io/docs/2.2.0/.

Helm Authors (2024). *Using Helm*. Helm. https://helm.sh/docs/intro/using_helm/.

Heß, S. (2019). *Carrera Digital 124/132 / CU Daten-Protokoll*. Slotbaer. http://slotbaer.de/carrera-digital-124-132/9-cu-daten-protokoll.html.

Heß, S. (2019). *Carrera Digital 124/132 / CU Rundenzähler-Protokoll.* Slotbaer. http://slotbaer.de/carrera-digital-124-132/10-cu-rundenzaehler-protokoll.html.

HoloViz developers. (2005). *User Guide. Holoviews*. Holoviews. https://holoviews.org/user_guide.

Joppich, R. et al. (2016). *MASTeR. Schablonen für alle Fälle*. Nürnberg:Sophist.

Jupyter Team. (2015). *Project Jupyter documentation*. Jupyter. https://docs.jupyter.org/en/latest/.

Kemmer, T. (2017, 2021, 2023). *Carrera Digital Control Unit.* Github. https://github.com/tkem/CarreraDigitalControlUnit.

Kemmer, T. (2015-2023). *Carreralib.* Github. https://github.com/tkem/carreralib.

Kemmer, T. (2016-2024). *Open Lap*. Github. https://github.com/tkem/openlap.

Longhorn Authors. (2019-2024). *The Longhorn Documentation*. Longhorn. https://longhorn.io/docs/1.6.0/.

Mirantis. (2024). *Lens - The way the world runs Kubernetes*. Github. https://github.com/lensapp/lens.

NumPy Developers. (2008-2022). *NumPy documentation*. Numpy. https://numpy.org/doc/stable/.

Open Vehicles Developers. (2019-2024). *Open Vehicles Monitoring System*. Open Vehicles. https://docs.openvehicles.com/en/latest/.

Pandas. (2024). *Pandas documentation*. Pydata. https://pandas.pydata.org/docs/.

Project Jupyter Contributors. (2024). *JupyterHub*. JupyterHub. https://jupyterhub.readthedocs.io/en/stable/.

Prometheus Authors. (2014-2024). *What is Prometheus?*. Prometheus. https://prometheus.io/docs/introd12 ptuction/overview/.

The Elasticsearch Authors. (2024). *Elasticsearch Guide*. Elastic. https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html.

The Flux authors. (2024) *Flux Documentation*. Fluxcd. https://fluxcd.io/flux/.

The Kasten Authors. (2017-2024). *K10 Overview.* Kasten. https://docs.kasten.io/latest/index.html.

The Kubernetes Authors. (2024). Kubernetes. https://kubernetes.io/docs/home/.

The Kubernetes Authors. (2024). *Command line tool (kubectl)*. Kubernetes. https://kubernetes.io/docs/reference/kubectl.

The Kubernetes Authors. (2024). *ConfigMaps.* Kubernetes. https://kubernetes.io/docs/concepts/configuration/configmap.

The Kubernetes Authors. (2024). *Ingress.* Kubernetes. https://kubernetes.io/docs/concepts/services-networking/ingress/.

The Kubernetes Authors. (2024). Persistent *Volumes.* Kubernetes. https://kubernetes.io/docs/concepts/storage/persistent-volumes/.

The Kubernetes Authors. (2024). *Pods.* Kubernetes. https://kubernetes.io/docs/concepts/workloads/pods.

The Kubernetes Authors. (2024). *Secrets*. Kubernetes. https://kubernetes.io/docs/concepts/configuration/secret/.

The Kubernetes Authors. (2024). *Service.* Kubernetes. https://kubernetes.io/docs/concepts/services-networking/service/.

Thielen, M. (2013). *SlotHub*. Github. https://github.com/MarkThielen/SlotHub.

The cert-manager Authors. (2024). cert-manager. https://cert-manager.io/docs/.

Utrilla, A. & Vehent, J. (2015) *Simple and flexible tool for managing secrets*. Github. https://github.com/getsops/sops.

Vertical Pod Autoscaler. . (2021). Github. https://github.com/kubernetes/autoscaler/tree/master/vertical-pod-autoscaler.