Anforderungen Projektphase II

Camp2Code

Florian Edenhofner und Robert Heise

Education4Industy GmbH

Zuletzt aktualisiert: 2024-09-18

Inhaltsverzeichnis

1 Überblick Projektziel

Das Ziel dieses Projektephase ist die Weiterentwicklung der Software der Projektphase 1. Die Software des Modellautos soll um weiter Fahrmodi erweitert werden. Dabei steht nun die Integration einer Kamera und die Steuerung des Auto mittels dieser Kamera im Vordergrund.

Das Ziel der ersten Woche besteht darin, die Kamera in die Software zu integrieren und mittels klassischer Methoden der Bildverarbeitung eine Steuerung zu entwickeln. Dabei darf durchaus ein mutiger und kreativer Ansatz gewählt werden.

In der zweiten Woche soll der Fokus auf der Verwendung eines Neuronalen Netzen zur Auswertung der Bilddaten der Kamera liegen

Die finale Software soll dem Anwender ermöglichen, zwischen verschiedenen Fahrmodi bzw. Steuerungverfahren zu wählen und diese zu starten. Sie wird in Form eines gut dokumentierten Quellcodes präsentiert. Eine kurze Anwenderdokumentation soll potenziellen Anwendern die Bedienung erleichtern bzw. ermöglichen.

2 Projektorganisation

2.1 Arbeitsweise

Das Projekt sollte als Teamarbeit in einer Gruppe von etwa fünf Personen durchgeführt werden. **Jedes Team soll gemeinsam eine Software bzw. einen Quellcode** entwickeln. Das Team ist für Planung, Entwurf, Entwicklung, Tests und Dokumentation verantwortlich.

In Anlehnung an eine agile Arbeitsweise soll vorrangig auf eine schlanke Projektplanung, die Definition von Teilaufgaben und die regelmäßige offene Kommunikation der Teammitglieder geachtet werden. Eine agile Methodologie (Scrum, Kanban) muss nicht strikt umgesetzt werden.

Teilaufgaben sollen als Arbeitspakete in kleinen Einheiten (User-Stories) beschrieben werden, die z.B. in einem Kanban- oder Scrum-Board organisiert werden können. So lassen sich Arbeitspaket innerhalb des Teams leicht dokumentieren und zuweisen.Es kann zusammen, ein Paaren oder einzeln an Teilaufgaben gearbeitet werden. Wichtig ist, dass die Arbeitsschritte aufeinander abgestimmt sind und agile Rituale wie z.B. ein “Daily Scrum” durchgeführt werden. (In einem Team können die Rollen des Product Owners und Scrum-Masters vergeben werden. Diese Teammitglieder sollten jedoch im Rahmen der Projektphase auch Teil des Entwicklerteam bleiben.)

2.2 Technische Voraussetzungen und Materialien

Folgende Schritte müssen zu Beginn des Projektes durchgeführt werden bzw. durchgeführt worden sein:

1. Raspberry Pi vorbereiten bzw. prüfen,
2. Notwendige Python-Files downloaden und sichten,
3. Funktionstests des Modellautos und gegebenenfalls Bugfixes,
4. Prüfen der Arbeitsumgebung aus Projektphase 1 (z.B. Remoteverbindung testen).
5. Testen der Software aus der Projektphase 1
6. Montage und Test der Kamera

2.3 Projektplanung

Die Anforderungen an die Software sind in diesem Dokument beschrieben. Folgende Aspekte sollten in der Planung und Teilplanung berücksichtigt werden.

- Zeitplan
- Aufgabenverteilung (siehe Arbeitsweise)
- Aspekte der technischen Zusammenarbeit (Datenaustausch, Versionierung)
- Modularisierung, Strukturierung und Dokumentation des Quellcodes
- Verwendung von Klassen und Funktionen

2.3.1 Quellcodeverwaltung und Versionierung

Die zu entwickelnde Software soll während der Entwicklung versioniert werden. Die Versionierungsverwaltung mittels einer passenden Ordnerstruktur (z.B. über Dateinamen und Archivordner) oder unter Verwendung von Git. In diesem Zusammenhang muss in der Planung geklärt werden wie Daten in den Teams ausgetauscht werden.

2.3.2 Dokumentation

Während der Projektarbeit soll begleitend eine Dokumentation angefertigt werden. Das soll zum einen dabei helfen den Projektfortschritt besser nachvollziehen zu können. Andererseits soll damit auch eine Grundlage für den Projektabschluss vorbereitet werden, damit Schnittstellen, Bedienungsweisen, erreichte Ziele, aber auch Abweichungen vom Projektziel oder Einschränkungen dokumentiert werden.

3 Anforderungsbeschreibung

Im folgenden werden die Anforderungen an die Software beschrieben.

3.1 Modularisierung

Die Software ist derart zu strukturieren, dass alle notwendigen Klassen und/oder Funktionen in einem oder mehreren Modulen zusammengefasst werden, welche in dem eigentlich auszuführenden Hauptprogramm verwendet werden.

3.2 Fahrmodi und Klassen

Dem Ansatz der objekt-orientierten Softwareentwicklung folgende sollen neue Klassen von der Klasse *BaseCar* der Projektphase 1 abgeleitet werden. Eine Klasse *CamCar* dient der Integration der Kamera. Von dieser sollen dann die Klassen *OpenCVCar* und *DeepCar* abgeleitet werden, welche jeweils einen eigenen Fahrmodus bereitstellen.

1. **Entwicklung einer Klasse CamCar** Diese Klasse soll auf BaseCar basieren und dessen Funktionalität erweitern. Die Klasse CamCar ermöglicht den Zugriff auf die Kamera bzw. auf ein aktuelles Bild der Kamera über eine geeignete Methode.
2. **Visualiserung der Fahrdaten und des Kamerabildes**. Das aktuelle Bild der Kamera sowie weitere interessante Fahrdaten sollen visualisiert werden. Dazu kann Dash verwendet werden.

   Anmerkung: Eine solche App kann sich als sehr nützlich für die Entwicklung der Klassen OpenCVCar und DeepCar erweisen, da einzelne Zwischenschritte der Bildverarbeitung zusätzlich im Bild visualisiert werden können.
3. **Entwickeln und Testen einer Klasse OpenCVCar**. Durch die Klasse ermöglicht eine Fahrspurverfolgung mittels “klassischer Methoden” mit OpenCV in Python ermöglicht werden. Dies entspricht einem eigenen Fahrmodus. Die Klasse OpenCVCar soll von der Klasse CamCar abgeleitet werden. Mittels dieser Klasse sind folgende Anforderungen umzusetzen:

- Verfolgen einer Spur mit je 2 Linien in ca. 15 cm Abstand (etwas mehr als die Breite des Autos).
- Starten und Stoppen dieses Fahrmodus durch entsprechende Methoden der Klasse
- Aufnehmen und Speichern von Bilddaten mit zugehörigem Lenkwinkel für das Training des Neuronalen Netzes der Klasse DeepCar (siehe 6.)

Die Gestaltung der Klasse Camcar, mittels Attributen und Methoden, obliegt Ihnen.

Anmerkung: Sie können Ihre eigene Klasse **BaseCar** aus Projektphase I oder die neu zur Verfügung gestellte Klasse nutzen.

4. **Entwickeln und Testen einer Klasse DeepCar**. Entwicklung einer Fahrspurverfolgung mittels eines Neuronalen Netzes unter Verwendung von Tensorflow/Keras. Dies entspricht ebenfalls einem eigenen Fahrmodus. Eine Klasse DeepCar soll ebenfalls von der Klasse CamCar abgeleitet werden. Es sollen folgende Anforderungen erfüllt werden:

   - Entwicklung und Trainieren eines Neuronalen Netzes, das ein Bild als Input erhält und den Lenkwinkel als Output zurückgibt
   - Integration des Netzes in die Software
   - Verfolgen einer Spur mit je 2 Linien in ca 15 cm Abstand ausschließlich unter Verwendung des trainierten Neuronalen Netzes
   - Starten und Stoppen dieses Fahrmodus durch entsprechende Methoden der Klasse

   Anmerkungen: Für das Trainieren des Neuronalen Netzes ist es notwendig Trainigsmaterial zu erstellen. Hier ist es hilfreich auf die Fahrspurverfolgung mit OpenCVCar zurückzugreifen. Alternativ können Sie, falls diese Fahrspurverfolgung nicht ausreichend gut funktioniert, auf eine Remotesteuerung zurückgreifen. Diese wird Ihnen zur Verfügung gestellt. “Image Augmentation” kann hilfreich sein, einen Trainingsdatensatz zu erweitern.

3.3 Nutzer Interface

Die Software soll dahin gehen erweitert bzw. zusammengefasst, dass dem Nutzer erlaubt wird, jeden der Fahrmodi auszuwählen und zu starten. Dies kann über eine einfache Menüführung im Terminal oder optional durch eine Erweiterung der Funktionalität der Dash-App geschehen.

Es soll eine kurze Anwendungsdokumentation für den Nutzer zur Verfügung gestellt werden.

3.4 Dokumentation

Die Dokumentation des erstellten Quellcodes soll als Minimalanforderung die Verwendung von *Doc-Strings* für Funktionen, Klassen und Methoden umfassen.

3.5 Präsentation der Software

Das fertige “Produkt” ist in einer abschließenden Präsentation vorzustellen. Die Präsentation sollte zum einen eine kurze an den potentiellen Anwender adressiert Demonstration beinhalten und zum anderen die Konzeption der Software (File, Klassen, Funktionen) erläutern.