Lastenheft Projektphase 1

Camp2Code

Florian Edenhofner und Robert Heise

Education4Industy GmbH

Zuletzt aktualisiert: 2024-01-13

Inhaltsverzeichnis

Inhaltsverzeichnis **i**

1. Projektauftrag **1**
1.1. Beschreibung des Projektziels 1
1.2. Anforderungen 2

2. Projektaufbau **5**
2.1. Technische Voraussetzungen und Materialien 5
2.2. Arbeitsweise 5
2.3. Projektplanung 6
2.4. Versionierung 6
2.5. Dokumentation 6
2.6. Vorstellung der Ergebnisse 6

A. Anhang **7**
A.1. Links 7
A.2. Weiterführende Dokumente 7

1. Projektauftrag

Titel: Camp2Code Projektphase I

Thema: Raspberry Pi Car – ein kleines autonomes Auto

1.1. Beschreibung des Projektziels

Das Ziel dieses Projektes besteht darin, eine Software zu entwickeln, die es dem Nutzer ermöglicht, ein Modellauto in verschiedenen Fahrmodi bzw. Fahrparcours fahren zu lassen. Zu diesem Zweck steht ein Modellauto zur Verfügung, das aus einem Chassis mit Lenkung und Antrieb besteht und zusätzlich mit einem Ultraschallsensor und einem Infrarotsensor ausgestattet ist. Als zentrale Steuereinheit für alle Komponenten wird ein Raspberry Pi mit dem Betriebssystem Raspberry-Pi-OS verwendet. Die Umsetzung der Software erfolgt mithilfe der Programmiersprache Python. Es werden verschiedene grundlegende Python-Klassen bereitgestellt, die den Zugriff auf die einzelnen Komponenten ermöglichen.

Das Ziel der ersten Woche besteht darin, die Bauteile und Sensoren des Autos anzusteuern bzw. auszulesen und erste einfache Fahrstrecken zu bewältigen. In der zweiten Woche soll das Auto in der Lage versetzt werden, eigenständig einer auf den Boden befindlichen Linie zu folgen. Darüber hinaus sollen die Fahrdaten des Autos (Informationen über den Status der Bauteile und Sensoren) während der Fahrt aufgezeichnet und visualisiert werden können.

Die endgültige Software soll es dem Anwender ermöglichen, zwischen verschiedenen Fahrstrecken 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.

1.2. Anforderungen

1. **Testen der Basisklassen:** Die Basisklassen werden im File *basisklassen.py* zur Verfügung gestellt. Sie umfassen die Klassen *BackWheels*, *FrontWheels*, *Ultrasonic* und *Infrared*. Klären Sie die Bedeutung der einzelnen Argumente der Konstruktoren und den Zwecke der einzelnen Klassenmethoden. (Das File *basisklassen.py* beinhaltet weitere Klassen, welche von den Basisklassen verwendet werden. Mit diesen brauchen Sie sich nicht auseinandersetzen oder sie verstehen.)
2. **Modularisierung:** Die Software ist derart zu strukturieren, dass alle nötigen Klassen und/oder Funktionen in einem oder mehreren Modulen zusammengefasst werden, welche in dem eigentlich auszuführenden Hauptprogramm verwendet werden.
3. **Klasse - BaseCar:** Entwicklung und Testen einer Klasse *BaseCar* mittels der Basisklassen mit vorgegebenen Anforderungen. Die Klasse soll folgende Properties und Methoden besitzen.

    - *steering_angle*: Setzen und Zugriff auf den Lenkwinkel (Property mit Setter)
    - *speed*: Setzen und Zugriff auf die Geschwindigkeit (Property mit Setter)
    - *direction*: Zugriff auf die Fahrrichtung (1: vorwärts, 0: Stillstand, -1 Rückwärts) (Property ohne Setter)
    - *drive(speed:int, direction:int)*: Methode zum Setzen von Geschwindigkeit und Fahrrichtung
    - *stop*: Methode zum Anhalten des Autos. Sie setzt die Geschwindigkeit auf Null und den Lenkwinkel auf Geradeaus.

    Prüfen Sie, ob die Properties *steering_angle*, *speed* und *direction* immer die korrekten Werte liefern. Dies muss unabhängig von der Verwendungsgeschichte einer korrekten Instanz gewährleistet sein. Die Klasse *BaseCar* soll mittels der folgenden Aufgaben getestet werden.

- **Fahrparcours 1 - Vorwärts und Rückwärts**: Das Auto fährt mit langsamer Geschwindigkeit 3 Sekunden geradeaus, stoppt für 1 Sekunde und fährt 3 Sekunden rückwärts.
- **Fahrparcours 2 - Kreisfahrt mit maximalem Lenkwinkel**: Das Auto fährt 1 Sekunde geradeaus, dann für 8 Sekunden mit maximalen Lenkwinkel im Uhrzeigersinn und stoppt. Dann soll das Auto diesen Fahrplan in umgekehrter Weise abfahren und an den Ausgangspunkt zurückkehren. Die Vorgehensweise soll für eine Fahrt im entgegengesetzten Uhrzeigersinn wiederholt werden.

4. **Klasse - SonicCar**: Eine Klasse *SonicCar* soll die Eigenschaften der Klasse BaseCar erben und zusätzlich den Zugriff auf den Ultraschallsensor ermöglichen. Gestalten Sie die Software derart, dass mittels einer Instanz von *SonicCar* folgende Fahrmodi gestartet werden können.

Während dieser Fahrten sollen die Fahrdaten so aufgezeichnet werden, dass diese nach der Fahrt von der Instanz abgefragt werden und beispielsweise gespeichert oder anderweitig verarbeitet werden können. Die Fahrdaten umfassen die Geschwindigkeit, die Fahrtrichtung, den Lenkwinkel und die Daten des Ultraschallsensors. Die Daten sollen die verschiedenen Steueranweisungen und die ihnen zugrunde liegenden Sensordaten einer Fahrt reflektieren.

- **Fahrparcours 3 - Vorwärtsfahrt bis Hindernis**: Fahren bis ein Hindernis im Weg ist und dann stoppen.
- **Fahrparcours 4 - Erkundungstour**: Das Auto soll bei freier Fahrt die Fahrrichtung, und optional auch die Geschwindigkeit, variieren. Im Falle eines Hindernisses soll das Auto die Fahrrichtung ändern und die Fahrt dann fortsetzen. Zur Änderung der Fahrrichtung ist dabei ein maximaler Lenkwinkel einzuschlagen und rückwärts zu fahren. Als Ergebnis soll das Auto den hindernsfreien Raum "erkunden". Die genaue Gestaltung obliegt Ihnen.

5. **Visualiserung der Fahrdaten mit Dash**: Die aufgezeichneten Sensordaten und Fahrdaten sollen mittels einer App visualisiert werden. Dazu ist eine App mit *Ploty Dash* zu entwickeln.

- **Dash Stufe 1**: Eine erste Entwicklungsstufe der App soll eine passende Überschrift und KPIs (z.B. in Form von Karten) in der App angezeigt werden. Folgende KPIs sollen basierend auf den aufgezeichneten Fahrdaten ermittelt und angezeigt werden:
  - die maximale Geschwindigkeit,
  - die minimale Geschwindigkeit
  - die Durchschnittsgeschwindigkeit, sowie
  - die zurückgelegte Gesamtfahrstecke und
  - Gesamtfahrzeit.

  Dabei soll der gesetzt Werte der Geschwindigkeit wie eine echte Geschwindigkeit (mit Einheit) behandelt werden. Entsprechend ergibt sich die Strecke aus dem Produkt von Geschwindigkeit und Fahrzeit. Die Einheit der Geschwindigkeit kann vernachlässigt bzw. muss nicht ermittelt werden. Berücksichtigen Sie, dass die konkrete Berechung der KPIs aus der Art der Aufzeichnung der Fahrdaten ergibt.
- **Dash Stufe 2**: Die zeitliche Entwicklung ausgewählter Fahrdaten soll grafisch dargestellt werden.
- **Dash Stufe 3**: Die App soll um ein interaktives Dropdown-Menü erweitert werden. Damit sollen die jeweiligen Fahrdaten für die Darstellung der zeitlichen Entwicklung gewählt werden können.

6. **Klasse - SensorCar**: Die Klasse *SensorCar* soll zusätzlich den Zugriff auf die Infrarotsensoren erlauben. Mittels dieser Sensoren soll das Auto in die Lage versetzt werden eine

Linie auf dem Boden zu erkennen. Die Daten der Infrarotsensoren sollen ebenfalls aufgezeichet werden. Anmerkung: Die Sensitivität der Infrarotsensoren kann durch das blaue Potentiometer eingestellt werden. Dies kann zur erheblichen Verbesserung der Ergebnisse führen.

- **Fahrparcours 5 - Linienverfolgung** : Folgen einer etwas 1,5 bis 2 cm breiten Linie auf dem Boden. Das Auto soll stoppen, sobald das Auto das Ende der Linie erreicht hat. Als Test soll eine Linie genutzt werden, die sowohl eine Rechts- als auch eine Linkskurve macht. Die Kurvenradien sollen deutlich größer sein als der maximale Radius, den das Auto ohne ausgleichende Fahrmanöver fahren kann.
- **Fahrparcours 6 - Erweiterte Linienverfolgung**: Folgen eine Linie, die sowohl eine Rechts- als auch eine Linkskurve macht mit Kurvenradien kleiner als der maximale Lenkwinkel.

7. **Fahrparcours 7 - Erweiterte Linienverfolgung mit Hindernisserkennung (Optional)**: Kombination von Linienverfolgung per Infrarot-Sensor und Hinderniserkennung per Ultraschall-Sensor. Das Auto soll einer Linie folgen bis ein Hindernis erkannt wird und dann anhalten.

8. **Nutzer Interface**: Die Software soll dahin gehen erweitert bzw. zusammengefasst, dass dem Nutzer erlaubt wird den jeweiligen Fahrparcours zu wählen und zu starten. Dies kann über eine einfache Menüführung im Terminal oder optional durch eine Erweiterung der Funktionalität der App geschehen. Es soll eine kurze Anwendungsdokumentation für den Nutzer zur Verfügung gestellt werden.

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

10. **Präsentation**: Das fertige “Produkt” ist in einer abschließenden Präsentation vorzustellen. Die Präsentation sollte zum einen eine kurze an den potentiellen Anwender adressiert Demonstration beeinhalten und zum anderen die gewählte konzeptionelle Umsetzung erläutern.

2. Projektaufbau

Die Teams haben die Möglichkeit die Softwarearchitektur flexibel selbst festzulegen. Es sollen dafür auf Basisklassen für die Motorsteuerung und Sensorabfragen genutzt werden.

2.1. Technische Voraussetzungen und Materialien

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

1. Raspberry Pi vorbereiten (OS installieren, Konfigurationen, Softwareinstallationen),
2. Notwendige Python-Files downloaden und sichten ,
3. Funktionstests des Modellautos und gegebenenfalls Bugfixes,
4. Einrichten einer individuellen Arbeitsumgebung (z.B. Remoteverbindung testen).

Alle dafür notwendigen Material werden separat zur Verfügung gestellt.

2.2. Arbeitsweise

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

Im Rahmen der agile Arbeitsweise soll vorrangig auf eine schlanke Projektplanung, die Definition von Teilaufgaben und die regelmäßige offene Kommunikation der Teammitglieder geachtet werden. Eine bestimmte agile Methodologie (Scrum, Kanban) muss nicht 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. Das Team wählt die einzelnen Arbeitspakete selbst. Es kann zusammen, zu Paaren oder einzeln an Teilaufgaben gearbeitet werden. Wichtig ist, dass die Arbeitsschritte aufeinander abgestimmt sind und agile Rituale wie das “Daily Scrum” durchgeführt werden. In jedem 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.

2.3. Projektplanung

Folgende Aspekt sollten in der Planung und Teilplanung berücksichtigt werden.

- Zieldefinition und -abgrenzung
- Zeitplan
- Aufgabenverteilung (siehe Arbeitsweise)
- Strukturierung und Dokumentation des Quellcodes
- Aspekte der technischen Zusammenarbeit (Datenaustausch, Versionierung)

2.4. 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.

2.5. Dokumentation

Während der Projektarbeit soll begleitend eine Dokumentation angefertigt werden. Das soll zum einen dabei helfen den Projektforschritt besser nachvollziehen zu können. Andererseits soll damit auch eine Grundlage für den Projektabschluss (siehe 14. und 15.) vorbereitet werden, damit Schnittstellen, Bedienungsweisen, erreichte Ziele, aber auch Abweichungen vom Projektziel oder Einschränkungen dokumentiert werden.

2.6. Vorstellung der Ergebnisse

Abschließend sollen die Ergebnis jedes Teams kurz präsentiert werden. Dabei soll die Software als Produkt vorgestellt und ihre Anwendung demonstriert werden. Zusätzlich soll auf die grundlegende Struktur der erstellten Software eingegangen werden.

A. Anhang

A.1. Links

A.2. Weiterführende Dokumente