Lastenheft Projektphase 1

Camp2Code

Florian Edenhofner und Robert Heise

Education4Industy GmbH

UNIVERSITY
4 INDUSTRY

Zuletzt aktualisiert: 2022-11-13

Inhaltsverzeichnis

1. Projektauftrag

Titel: Camp2Code Projektphase I

Thema: Raspberry Pi Car – ein kleines autonomes Auto

1.1. Zusammenfassung des Projektziels

Ein Raspberry Pi soll in ein Modellauto integriert werden, um dieses verschieden Fahrparcours bzw. in verschiednen Fahrmodi fahren zu lassen. Dazu steht ein Bausatz zur Verfügung, welcher ein lenk- und fahrbares Chassis sowie einen Ultraschallsensor und einen Helligkeitssensor umfasst. Die Implementierung der Software erfolgt mit Python. Es werden mehrere Basisklassen zu Verfügung gestellt, die den Zugriff auf die Motoren und Sensoren erlauben.

Ziel der ersten Woche ist es die Bauteile und Sensoren des Autos ansteuern bzw. auslesen zu können und erste einfache Fahrparcours zu bewerkstelligen. Das Ziel der zweiten Woche ist es dem Auto zu ermöglichen selbstständig einer Linie zu folgen. Zudem sollen die Fahrtdaten des Autos (Informationen über Bauteile und Sensoren während der Fahrt) aufgezeichnet und visualisiert werden können.

Die finale Software soll es den Anwender erlauben zwischen den verschiedenen Fahrparcours zu wählen und diese zu starten. Die Software soll in Form eines dokumentierten Quellcodes vorliegen. Eine kurze Dokumentation soll einem potentiellen Anwender die Bedienung ermöglichen bzw. erleichtern.

1.2. Anforderungen

1. Testen der Basisklassen. Die Basisklassen umfassen die Klassen *Back_Wheels*, *Front_Wheels*, *Ultrasonic* und *Infrared*. Ziel ist es diese vier Basisklassen anwenden zu können.
2. **Klasse - BaseCar:** Entwicklung und Testen einer Klasse BaseCar mittels der Basisklassen mit vorgegebenen Anforderungen. Die Klasse soll folgende Attribute und Methoden (inkl. Gettern und Settern) besitzen.

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

  Die Klasse BaseCar soll mittels 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.

3. **Klasse - SonicCar**: Eine Klasse SonicCar soll die Eigenschaften der Klasse BaseCar erben und zusätzlich den Zugriff auf den Ultraschall-Sensor ermöglichen. Die Fahrdaten sollen ausgezeichnet werden können. Sie umfassen die Geschwindigkeit, die Fahrtrichtung, den Lenkwinkel und die Daten des Ultraschall-Sensors. Dazu soll Folgendes entwickelt und getestet werden.

- **Fahrparcours 3 - Vorwärtsfahrt bis Hindernis**: Fahren bis ein Hindernis im Weg ist und dann stoppen. Während dieser Fahrt sollen die Fahrdaten (Geschwindigkeit, Lenkwinkel, Fahrrichtung, Sensordaten) aufgezeichnet werden.
- **Fahrparcours 4 - Erkundungstour**: Das Auto soll bei freier Fahrt die Fahrrichtung und eventuell auch die Geschwindigkeit variieren und im Falle eines Hindernisses die Fahrrichtung ändern und die Fahrt fortsetzen. Zur Änderung der Fahrrichtung soll ein maximaler Lenkwinkel eingeschlagen und rückwärts gefahren werden. Als Ergebniss soll das Auto den hindernsfreien Raum “erkunden”. Zusätzlich sollen die Fahrdaten aufgezeichent werden.

4. **Visualiserung der Fahrdaten mit Dash**: Die während einer Fahrt aufgezeichneten Sensordaten und Fahrdaten sollen mittels einer App visualisiert werden. Dazu soll eine Dash-App entwickelt werden.

- **Dash Stufe 1**: Eine erste Entwicklungsstufe der App soll eine passende Überschrift und KPIs (in Form von Karten) in der App angezeigt werden. Die KPIs sollen basierend auf den aufgezeichneten Fahrdaten erstellt werden: Maximale Geschwindigkeit, minimale Geschwindigkeit und Durchschnittsgeschwindigkeit, zurückgelegte Gesamtfahrstecke, Gesamtfahrzeit. Die zurückgelegte Fahrstrecke soll aus den aufgezeichneten Fahrdaten Geschwindigkeit und Fahrzeit ermittelt werden. Die Einheit der Geschwindigkeit kann vernachlässigt werden.
- **Dash Stufe 2**: Die zeitliche Entwicklung ausgewählter Fahrdaten soll grafisch dargestellt werden.
- **Dash Stufe 3**: Die App soll um ein interaktives Dropdown-Menü. Damit sollen die jeweiligen Fahrdaten für die Darstellung der zeitlichen Entwicklung gewählt werden können.

5. **Klasse - SensorCar**: Die Klasse SensorCar soll zusätzlich den Zugriff auf die Infrarot-Sensoren besitzen. Mittels dieser Sensoren soll das Auto in die Lage versetzt werden eine Linie auf dem Boden zu erkennen. Die Daten der Infrarotsenoren 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.

6. **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.

7. Die Software soll dahin gehen erweitert bzw. zusammengefasst, dass dem Nutzer erlaubt wird den jeweiligen Fahrparcour zu wählen und zu starten. Dies kann eine einfache Menüführung im Terminal oder auch durch eine Erweiterung der Funktionalität der Dash-App geschehen. Dem Nutzer soll eine kurze Anwendungsdokumentation zur Verfügung

gestellt werden.

8. Vollständige Dokumentation des Vorgehens und des erstellten Codes
9. Präsentation der Ergebnisse

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.

1. Raspberry Pi vorbereiten (OS installieren, Konfigurationen, Softwareinstallationen)
2. Notwendige Python-Files downloaden und sichten. Entsprechende Links für Repositorys werden gesondert zur Verfügung gestellt.
3. Funktionstests des Modellautos und gegebenenfalls Bugfixes
4. Einrichten einer individuellen Arbeitsumgebung auf den Raspberry Pi bzw. Remoteverbindung testen.

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 berücksichtigt werden. - Projektzieldefinition und -abgrenzung - Projektzeitplan - Aufgabenverteilung (siehe Arbeitsweise) - 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 Fehler 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