Pflichtenheft: Parksystem

Dominik Lysiak Daniel Assfalg

7. April 2017

Inhaltsverzeichnis

1 Zielbestimmung

1.1 Musskriterien

Es soll eine möglichst genaue Anzeige über die Verfügbarkeit (frei/besetzt) von mit Sensoren bestückten Parkflächen geliefert werden. Auf einer einfachen Weboberfläche soll dann der Status der Parkplätze für den Benutzer angezeigt werden.

1.2 Sollkriterien

Das System sollte so entworfen werden, dass bei einfacheren Störungen - wie eine kurzzeitige Annäherung einer Person - abgefangen werden können, um mögliche Falschinformationen filtern oder sogar berichtigen zu können.

1.3 Kannkriterien

Eine Analyse der gespeicherten Verfügbarkeiten der Parkflächen, um Prognosen liefern zu können oder um nützliche "Big DataÏnformationen wie zum Beispiel PeakZeiten für die Benutzung der Parkplätze oder um eine Heatmap auf der Weboberfläche generieren zu können. Eine Navigation zum nächsten freien Parkplatz wäre für eine App denkbar. Außerdem wäre eine unabhängige Stromversorgung durch Solarenergie für die einzelnen Module ein Ziel für die Umweltsunabhängigkeit des Systems.

1.4 Abgrenzung

Im Gegensatz zum Parkhausparkplatzmanagement soll sich dieses Projekt auf die Erfassung einzelner Parkflächen unabhängig Ihrer Lage befassen. Dabei gilt es das Projekt unabhängig von der Umgebung und Lage des Parkplatzes zu implementieren zu können. In jeder Umgebung und Situation. Außerdem soll nicht wie im Parkhaus nur die Freie Anzahl der Parkplätze angezeigt werden, sondern zudem die genaue Position z.B. durch GPS-Koordinaten.

2 Produkteinsatz

2.1 Anwendungsbereiche

Das Projekt soll Autofahren die Parkplatzsuche erleichtern und dabei assistieren. Es soll ebenfalls Hochschulen oder Unternehmen, sowie Kommunen oder Städten bei der Planung von Veranstaltungen oder Errichtung neuer Parkflächen helfen und damit die Pünktlichkeit und Effizienz steigern.

2.2 Zielgruppen

Die Zielgruppen sind dabei Autofahrer wie Pendler, die jeden Tag das Problem mit der Parkplatzsuche erleben. Besucher, die kein Wissen über die Parksituation haben. Einwohner, die in Städten oft Schwierigkeiten haben einen Parkplatz zu finden. Aber auch Städte, größere Unternehmen und Hochschulen sind Zielgruppen, da die gewonnen Information bei der Planung von Veranstaltungen helfen können oder bei der Planung von neuen Parkplatz Anlagen entscheidende Einflüsse liefern. Ebenfalls kann die Stadt zum Beispiel die Parkdauer auf zeitlich begrenzten Parkplätzen überprüfen

2.3 Betriebsbedingungen

Das System wird in Module eingeteilt und sendet dauerhaft die gewonnen Informationen der Sensoren an einen Hauptserver. Ein Modul besteht dabei aus einer kleinen Recheneinheit und aus einem oder mehreren Sensoren. Für die ersten Phasen des Projekts wird dabei einfachheitshalber auf ein RaspberryPi sowie auf Ultraschallsensoren zurück gegriffen. Der Server wird ebenfalls auf einem RaspberryPi betrieben.

3 Produktumgebung

3.1 Software

Für die Module wird bei der Verwendung des RaspberryPi das mitgelieferte Rasbian OS benutzt und der Server wird auf einer Linux Distribution betrieben. Auf beiden Betriebssystemen wird Python installiert und verwendet. Zusätzlich wird auf dem Server die Apache2 Software mit der MariaDB Erweiterung installiert. Für die Benutzeroberfläche des Servers zur Ausgabe der Parkplatzsituation wird das Django-Framework für Python installiert.

3.2 Hardware

Jedes Modul besteht aus einer Recheneinheit sowie aus einem oder mehren Ultraschallsensoren. Dazu wird ein Server benötigt welcher die Auswertung der Ergebnisse, sowie die Ausgabe auf einer Weboberfläche übernimmt.

3.3 Produktschnittstellen

Die Übertragung der Module erfolgt in JSON TCP Paketen. Das Einlesen der Daten auf dem Server ebenso. Gesendet wird dabei die Modulidentifikationsnummer, Sensor-ID, Aufnahmedatum, und ein Token für das authentifizieren der Dateneingabe.

4 Produktfunktionen bzw. Projektumsetzung

Die Funktionen unterteilen sich in zwei Bereiche, Server und Modul. Die Modul-Einheit muss die Sensor-Daten auswerten, berechnen, validieren können. Zusätzlich muss das Ergebnis dann an den Server verschickt werden. Der Server empfängt die Daten und nach einer Validierung und Authentifizierung speichert er diese in der Datenbank ab. Die Ausgabe des Webinterface erfolgt separat und wird durch ein Framework (Django) generiert.

5 Produktdaten

Um zukünftig das System effizienter gestalten zu können, müssen die Daten der Parkflächen für eine spätere Verarbeitung gespeichert werden. Dabei wird im Projekt auf eine SQL Datenbank zurück gegriffen. Diese Daten ergeben sich aus dem Parkplatz-Status (frei / belegt) und der Zeitangabe, sowie den Ortsdaten (Koordinaten).

6 Produktleistungen

Das Produkt soll mindestens 3 Parkplätze auf einmal verarbeiten können. Das Produkt soll in 90 Prozent aller Parkplatzauswertungen richtig liegen. Die Parkplatzsituation soll sich jede 10 Sekunden aktualisieren.

7 Qualitätsanforderungen

Bezeichnung	sehr gut	gut	normal	nicht relevant
Robustheit	x	-	-	-
Zuverlässigkeit	x	-	-	-
Korrektheit	-	x	-	-
Benutzerfreundlichkeit	-	-	x	-
Effizienz	-	x	-	-
Übertragbarkeit	-	-	-	x

8 Benutzungsoberfläche

Der Client ist ein beliebiger Webbrowser, wie z.B. Internet Explorer, Firefox, Opera oder Safari und die Bedienung der Benutzungsoberfläche erfolgt hauptsächlich durch die Maus, Eingaben erfolgen per Tastatur.

9 Spezielle Anforderungen an die Entwicklungsumgebung

10 Gliederung in Teilprodukte

Der Hauptbestandteil dieses Produkt ist die Zusammenführung der unten aufgelisteten Teilprodukte.

- Modulentwicklung - Auslesen und versenden der Daten
- Servermanagement - Verarbeitung und Ausgabe der Daten
- Webseitenentwicklung - Benutzeroberfläche