# DAIS Dokumentation

Dashboard and Infotainment System

Gruppe 7

Noll, Marco, 870529
Frenzel, Julian, 870716
Schug, Jens Cedric, 870886
Zimmermann, Philipp, 870872

## Übersicht

Im Verlauf dieses Projektes wurden zwei Anwendungen entwickelt, die in einem Simulator für Autos verwendet werden sollen. Um einen guten Überblick über die zu implementierenden Funktionen zu haben wurde zu Beginn ein Konzeptpapier erstellt, in dem die wichtigsten Funktionen und mögliche Zusatzfunktionen festgehalten wurden. In der darauffolgenden Zeit wurde dann ein Design auf Basis des „MAVE“-Designs entwickelt. Die Kommunikation zwischen dem Simulator und der Oberfläche erfolgt dabei über MQTT.

## Dashboard

Die Anwendungen verwenden das Qt-Framework und wurden in dem dazugehörigen QtCreator erstellt. Der grundlegende Aufbau basiert auf vergangen Projekten und orientiert sich an Beispielen für QML und MQTT.

### Software-Design

Die C++ - Dateien bilden hierbei ein Interface, über das die QML-Oberfläche mit dem MQTT-Broker kommunizieren kann. Die „main.cpp“ übernimmt dabei zum Großteil nur die Aufgabe, das QML-Programm und den benötigten MQTT-Client zu initialisieren. Die weitere Verarbeitung der Daten, die von dem MQTT-Broker empfangen werden, werden dann innerhalb der QMLs weiterverarbeitet.

Die Oberfläche ist aus mehreren Elementen zusammengesetzt, deren Kern die Datei „dashboard.qml“ darstellt. Hier wird der MQTT-Client instanziiert und mit dem MQTT-Broker verbunden. Anschließend werden an alle Benötigten Variablen mit dem MQTT-Client „subscribed“. Empfangene Daten werden dann in dem „ValueSource“-Element zwischen gespeichert und von dort aus weiterverwendet. Die einzelnen Elemente verwenden dann zum Beispiel zur Anzeige der Temperatur die in der „ValueSource“ entsprechende Variable.

### Aufbau

Ausgehend von der „dashboard.qml“-Datei werden die einzelnen Anzeigen mit verschachtelten Objekten eingebunden. Unterteilt wurde diese Datei in mehrere Layout-Bereiche (leftPart, centerPart, rightPart), dem Hintergrundframe und einem versteckten Login-Fenster. Da sich das Dashboard grundlegend auf einem horizontalen rechteckigen Bildschirm befindet, wurde ein Hintergrund eingebunden, der der Anwendung eine ansprechende Form verleiht. Das Login-Fenster kann nur mit einer Tastenkombination (Ctrl+L) hervorgehoben werden, da die MQTT-Verbindung lediglich beim Starten des Simulators aufgebaut wird. Die Layout-Bereiche dagegen sind durchgehend sichtbar. Sie enthalten weitere Design-Objekte, die in separaten QML-Dateien ausgelagert sind. Unter der „leftSignals.qml“ und „rightSignals.qml“ Datei befinden sich alle benötigten Signalleuchten, wie den Blinkern, den Motorkontrollleuchten und den Fahrzeugleuchten. Darunter sind „TachoMeter.qml“ und „SpeedMeter.qml“ angeordnet. Sie zeigen die aktuelle Geschwindigkeit, Drehzahl und Gangart an. Abgeschlossen werden die Anzeigen mit den Temperaturen des Öl- und dem Kühlwasserkreislaufes, die in den Dateien „TempOil.qml“ und „TempWasser.qml“ zusammengestellt wurden. Der

„centerPart“-Bereich besteht aus einer Uhrzeit und der Tankanzeige aus „FuelSystem.qml“. Zusätzlich befindet sich ein interaktives Fenster im Zentrum des Dashboards. Das darin enthaltene Objekt „CarStatus.qml“ kann mit den Tastenkürzel „Ctrl+Left“ und „Ctrl+Right“ mit der Datei „Media.qml“ ausgetauscht werden. Dadurch wird es dem Nutzer ermöglicht zwischen den aktuellen Fahrzeuginformationen oder der Medienanzeige zu wechseln und die für ihn nützlichere Anzeige zu wählen.
Die dabei verwendeten Inhalte wie Icons und Bilder wurden unter „https://www.flaticon.com/packs/car-dashboard-signals“ und „https://imgbin.com/download-png/E43Rdn1b“ erworben und nachträglich bearbeitet.

# Infotainment

Die Infotainmentanwendung besteht aus einer großen Applikation, welche aus fünf unterschiedlichen, kleineren Modulgruppen zusammengesetzt ist. Diese Modulgruppen beinhalten: Einen Musikspieler, eine Simulation eines Telefons, die Anzeige von autospezifischen Informationen, eine Kartenanwendung und Navigationsmöglichkeit und das Impressum.

## Software-Design

Die Datei „main.qml“ beinhaltet alle Daten, die für die Interaktion und das Design der Anwendung wichtig sind. Die Dateien Pageform1.ui.qml bis Pageform5.ui.qml beinhalten nur wenig Code, da sie als Instanz in der „main.qml“ verwendet werden und dort alle wichtigen Elemente (Buttons, Images usw.) hinzugefügt werden. Es ist nicht möglich eine Interaktion von Buttons in „ui.qml“ Dateien zu implementieren, deshalb musste die gesamte Implementierung in die „main.qml“ ausgelagert werden. „ui.qml“-Dateien dienen primär der Darstellung des UI-Inhalts und -Design. In „AddressModel.qml“ und „MP3Model.qml“ stehen Informationen, die für den Musikspieler und bei der Nutzung des Kartenmaterials relevant sind. Die restlichen Dateien implementieren die MQTT-Schnittstelle und initialisieren das Anwendungsfenster. Die empfangenen Daten vom MQTT-Broker werden durch die Klasse ValueSource repräsentiert. In dem Ordner „background“ befinden sich alle verwendeten Icons und Bilder. Alle Musikstücke des Musikspielers sind im Ordnerverzeichnis „music“.

## Aufbau

### Verbindung zum MQTT-Broker

Wenn die „main.qml“ geladen wurde, wird automatisch eine Verbindung zum Host des MQTT-Brokers hergestellt. Dabei müssen folgende Parameter beachtet werden: der Hostname, der Port, der Benutzername und das Passwort. Die Implementierung der MQTT-Schnittstelle und MQTT-Subscription wurde aus dem Beispiel des Tesla-Dashboards (https://github.com/Garzuuhl/DAIS/tree/master/Seafile/dashboard-examples) übernommen , welches zur Verfügung gestellt wurde.

### Musikspieler

Der Benutzer kann mit mehreren Buttons interagieren. Musikstücke können durch Drücken dieser Schaltflächen gestartet und gestoppt werden. Es wird die Funktion playMusic bzw. stopMusic aufgerufen. Sie aktualisiert die Bilder der Buttons, damit sie zum jeweiligen Zustand passen und spielt das momentane Musikstück ab bzw. stoppt es. Dabei wird durch einen Timer der aktuelle Zeitpunkt des Musikstücks bestimmt, konvertiert (die Zeitangabe ist in Millisekunden) und danach angezeigt. Entsprechend dieser Zeit wird die Größe eines Rechtecks verändert, d.h. ist das Musikstück bei der Hälfte der Gesamtlänge, wird auch das Rechteck halb so groß. Dadurch sieht der Benutzer immer genau, wie lange der Titel schon gespielt wurde. Zusätzlich besteht die Möglichkeit den letzten und

den nächsten Titel abzuspielen. Der Titel des Musikstücks und die Anzeige der Songlänge werden entsprechend aktualisiert.

Beim Musikspieler wurden Inhalte von BenSound (https://www.bensound.com/) verwendet. Momentan können nur die drei Lieder, die im Projektverzeichnis enthalten sind, verwendet werden. Es wurde in Erwägung gezogen, dass der Benutzer auch Lieder vom USB-Gerät , von Spotify® oder anderen Diensten laden kann. Aus zeitlichen Gründen wurde dieser Gedanke jedoch revidiert.

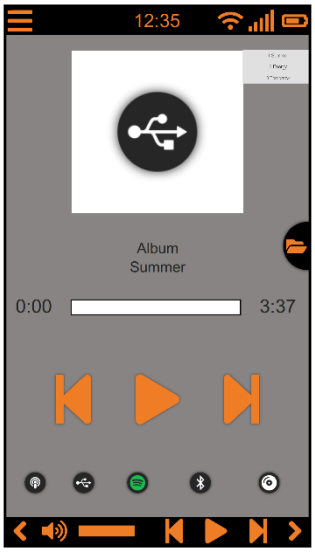


Abb. 1 zeigt den Musikspieler

## Telefon

Wie bei einem realen Telefon kann der Benutzer eine Nummer eingeben. Drückt man auf den orangenen Hörer wird ein Telefonanruf simuliert. Will man die Nummer neu eingeben, kann man mit Hilfe der „Zurück“-Taste die Nummer löschen. Momentan ist nur das Löschen der gesamten Nummer möglich. Als Zusatzfeature war ein Kontaktbuch und Favoriteneinträge gedacht.

Da, der Fahrsimulator kein Telefonnetz benutzen kann bzw. das System dieses nicht benötigt, wurde ein Telefonanruf simuliert. Nachdem man die Nummer eingeben hat erscheint ein neuer Bereich in dem drei Kreise, ein Label mit der aktuellen Gesprächszeit und ein Telefonhörer dargestellt werden. Die drei Kreise simulieren den Verbindungsaufbau zwischen den Gesprächspartnern. Durch eine Animation färben sich die Kreise orange. Nach einer festen Zeit wird das Gespräch beendet und man kehrt zur Eingabe der Telefonnummer zurück.

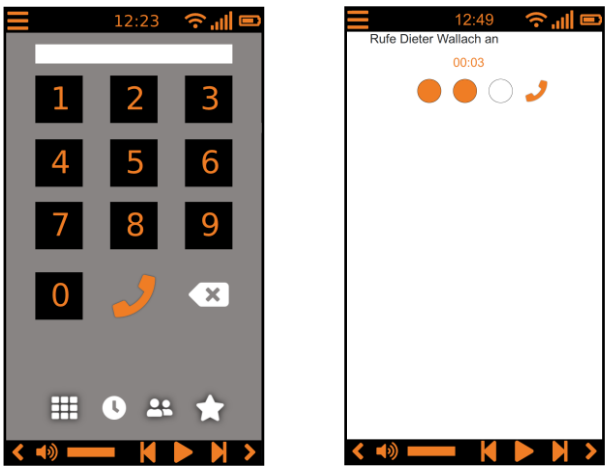


Abb. 2 zeigt das Telefonmenü und die Simulation des Anrufs

## Fahrzeuginformationen

Vom MQTT-Broker werden verschiedenste Daten wie Geschwindigkeit, Radumdrehungen, Luftdruck der Reifen usw. versendet. Diese Daten werden bei der Anzeige der Autoinformation verwendet. Der Reifendruck wird an einem Bild von einem Auto angezeigt. Der momentane Kilometerstand wird abgerufen und auch als Label gerendert. Es ist zu beachten, dass der Simulator nicht alle Werte eines realen Autos widerspiegelt, da es sich um ein spezielles Setting handelt. Die MQTT-Anbindung ist funktionstüchtig und die vorhandenen Werte werden auch übernommen.

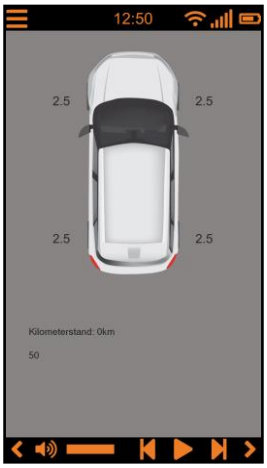


Abb. 3 Fahrzeuginformationen

## Karteninformationen und Navigation

### *Karteninhalte*

Das vierte Modul ist eine Anwendung der Klasse „Map“. Karteninhalte werden anhand eines Plugins namens „osm“, welches Kartenmaterial von OpenStreetMap (https://www.openstreetmap.de/) verwendet, geladen. Damit das Plugin funktioniert muss man einen „TileServer“ angeben. Dieser Server liefert Karteninformationen für verschiedene Anwendungsgebiete. Es gibt z.B. Karten für Radfahrer, für Wanderer und den „normalen“ Straßenverkehr. Die aktuelle Position in einer Fahranwendung (gegeben in Längen und Breitengrad) wird vom MQTT-Broker übermittelt. Diese Standortinformation wird verwendet, um die Inhalte der Karte zu aktualisieren bzw. zu positionieren. Der Mittelpunkt der Karte ist immer der aktuelle Standort. Um die Position besser darzustellen wurde ein orangener „MapCircle“ benutzt.

### *Navigation*

Ursprünglich war gedacht, dass der Benutzer einen Begriff in eine Suchleiste eingibt und die Nominatim-API (http://nominatim.openstreetmap.org) den Namen in eine Geoposition (enthält Längen und Breitengrad des Suchorts) umwandelt. Diese Information kann durch das QT-Element „Routequery“ genutzt werden. „Routequery“ benutzt die Openstreetmap-API, um eine Route zu generieren. Danach kann die Route graphisch dargestellt werden.

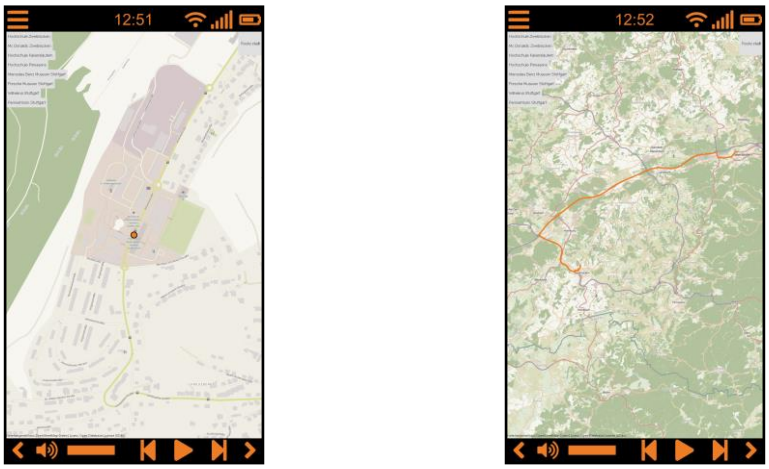


Abb. 4. Navigation und Routenplanung

## Kopf, Fußzeile und Customcontrolls

Die Kopfzeile enthält die aktuelle (lokale) Systemuhrzeit und einen Button, mit dem man das Hauptmenü aufrufen kann.

Die Fußzeile enthält die im Musikspieler implementierten Buttons im Kleinformat. Außerdem ist es möglich den Song zu muten und die Lautstärke mit Hilfe eines Lautstärkereglers festzulegen. Drückt der Benutzer auf einen der Pfeile links oder rechts gelangt man zu dem Lüfter und Heizungssteuerung.


Abb. 5: Header und Folter

## Lüfter und Heizungssteuerung

Die Stärke des Lüfters kann in drei Stufen durch „Plus“- und „Minus“-Tasten gewählt werden. Außerdem wird die Temperatur des Fahrzeuginnenraums (in Grad Celsius) dargestellt. Die Fahrer und Beifahrersitzheizung lassen sich an und ausschalten. Drückt man auf den Pfeil, der nach oben zeigt, wird ein erweitertes Kontextmenü geöffnet. Dort findet man Schaltflächen für die Front bzw. Heckheizung und Raumzirkulation


Abb. 6: Lüftersteuerung, Heizung und Temperatur

## Hauptmenü / Auswahl der Szenen

Das Hauptmenü dient als Navigationselement und kann durch das Auswählen des Burger Menüs im Header geöffnet werden. Die Applikation ist durch Hinzufügen weiterer Content Seiten erweiterbar. Man kann durch Wischgesten nach links bzw. rechts durch die Inhalte der Applikation „swipen“. Bis man jedoch an die richtige Stelle gelangt kann es unter Umständen recht lange dauern. Durch Betätigen der Buttons gelangt der Nutzer schnell zum jeweiligen Inhalt.

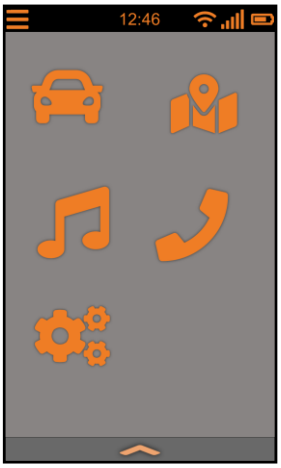


Abb. 7: Hauptmenü, Navigation durch die Applikation

## Impressum

Da der Platz im Dashboard begrenzt ist, wurde im Infotainment das Impressum angelegt. Hier wird auf die verwendete Musik, Plugins und rechtlichen Informationen eingegangen.


Abb. 8: Impressum