CHALMERS | UNIVERSITY OF GOTHENBURG

HMI för styrenhet i fordon

Kandidatarbete i Computer Science and Engineering

ERIC HÖRBERG
GUSTAV SVENSSON

Department of Computer Science and Engineering
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2016

KANDIDATARBETE

HMI för styrenhet i fordon

ERIC HÖRBERG
GUSTAV SVENSSON

Department of Computer Science and Engineering
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET

Göteborg, Sverige 2016

HMI för styrenhet i fordon
ERIC HÖRBERG
GUSTAV SVENSSON

© ERIC HÖRBERG, GUSTAV SVENSSON, 2016

Examinator: Christer Carlsson

ISSN 1654-4676

Department of Computer Science and Engineering
Chalmers tekniska högskola
SE-412 96 Göteborg
Sverige
Telefon: +46 (0)31-772 1000

The Author grants to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet. The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

Omslag:
Sentients demonstrationsbil där applikationen testats.

Department of Computer Science and Engineering
Göteborg, Sverige 2016

HMI för styrenhet i fordon

ERIC HÖRBERG
GUSTAV SVENSSON
Department of Computer Science and Engineering, Chalmers tekniska högskola

Göteborgs universitet
Kandidatarbete

Sammanfattning

För styrning i moderna fordon används sedan länge styrservon. För att reducera yttre faktorers påverkan utvecklar företaget Sentient ett system för styrning av styrsevot. I detta examensarbete har ett nytt HMI för deras styrsystem utvecklats. Krav på HMI:t var att det skulle implementeras på en surfplatta och om möjligt utnyttja trådlös kommunikation. HMI:t hade också krav på att vara snabbt, användarvänligt och kunna erbjuda funktioner deras tidigare HMI inte kunde. Röststyrning skulle också implementeras men detta i mån av tid.

Arbetet har utvecklats genom veckovis planering där olika delmål satts upp i samråd med Sentient. Därefter har utveckling av både applikation och kommunikation utförts. Utvecklingen har skett i Java och XML för applikationen och i C++ för utveckling till Arduinoenheten. För att testa de implementationer som gjorts har såväl fordon som CAN-lyssnare använts. För återkoppling på HMI:t så har acceptanstester utförts av Sentient och deras externa kunder.

Under de cirka två månader som arbetet utförts så har ett fullt fungerande HMI utvecklats. HMI:t består av en applikation på en surfplatta med Android. Kommunikationen till fordonets CAN-nätverk har skett via Bluetooth och en Arduinoenhet. För överföringen över Bluetooth har ett protokoll utvecklats som säkerställer säker överföring. Responsen har varit att HMI:t varit ändamålsenlig, varit visuellt tilltalande och att röststyrningen har haft potential för vidare utveckling.

De mål som varit ställda på såväl funktion som layout har uppnåtts. Både vi själva som Sentient och deras externa kunder har varit väldigt nöjda med resultatet av detta examensarbete och HMI:t används redan i Sentients testfordon och kommer vidareutvecklas för framtiden.

Nyckelord: Android, Arduino, Bluetooth, fordon, HMI, styrsystem, Sentient

Abstract

To use steering in modern vehicles power steering has been used since a long time. To reduce the impact of outer factors on steering the company Sentient has developed a new system to control the power steering. This thesis had the purpose to develop a new HMI for their system. The new HMI had requirements on that it should be implemented on a tablet and, if possible, use wireless communication. It also had requirements on being fast, usable and offer functionality not available in their former HMI. Voice commands should be implemented as time permits.

The work has been done with weekly planning where partial goals has been set each week in collaboration with Sentient after which development of both communication and application has been done. The development of the application has been done using Java and XML. The development of an Arduino unit has been done in C++. To test the implementations both a vehicle and CAN-listeners has been used. To get feedback on the HMI tests of acceptance has been made with Sentient as well as with some of their customers.

During the two month this thesis has been in work a fully functional HMI has been developed. The HMI consists of an application on a tablet using Android which uses Bluetooth to communicate with an Arduino unit that forwards the signals to the steering controller over the vehicle CAN-network. For the transmission over Bluetooth a protocol has been developed which ensures a secure transmission between the units. The system has been tested by personnel at Sentient as well by external customers to Sentient and the response has been that the HMI worked well, was appropriate, had a good layout and that the voice controller had potential for further development.

The goals that was set to both function and layout has been achieved, and like ourselves, both Sentient and their external customers has been pleased with the result of this thesis and the HMI is already in use in Sentients test vehicle and will be further developed in the future.

Förord

Denna rapport är resultatet av ett examensarbete på Dataingenjörsprogrammet på Chalmers tekniska högskola gjort av Gustav Svensson och Eric Hörberg. Examensarbetet har innefattat 15 högskolepoäng och har utförts hos Sentient med målet att skapa ett nytt HMI för deras styrsystem i testfordon.

För detta vill vi rikta ett särskilt tack till all involverad personal på Sentient som givit oss denna chans att utveckla och utvecklas till bättre och mer rustade ingenjörer men även ett stort tack till vår handledare Håkan Burden på Viktoria Institutionen/Chalmers för all inspiration och rådgivning under detta projekt.

Ordlista

Acceptanstest Test av uppfyllande av uppsatta krav

API Application Programming Interface

AOSP Android Open Source Project

CAN Controller Area Network

HMI Human Machine Interface

HW Hardware

ISM Industrial, Scientific och Medical

ISO International Organization for Standardization

LIFO Last In First Out

MVC Model, View, Controller

MVP Model, View, Presenter

SW Software

UI User Interface

XML Extensible Markup Language

Innehåll

1 Introduktion

Under tiden som människan funnits på denna jord så har det funnit ett behov av att transportera sig. Från att behöva transportera sig genom att bara använda fötterna tills nu med mer avancerade motordrivna fordon så som t.ex. fartyg, flygplan och givetvis också bilar.

Den första bilen byggdes år 1768 av Fransmannen Nicolas-Joseph Cugnot. Cugnot var verksam inom den franska armén där han först verkade som militäringenjör för att senare bli kapten för armén. Bilen, vars främsta syfte var att transportera millitär utrustning, hade tre hjul fördelade på två bak och ett fram och utrymme för 4 personer. För att drivas användes en ångmotor som hade en toppfart på 4km/h och som var femtonde minut behövdes fyllas med vatten [Lie00].

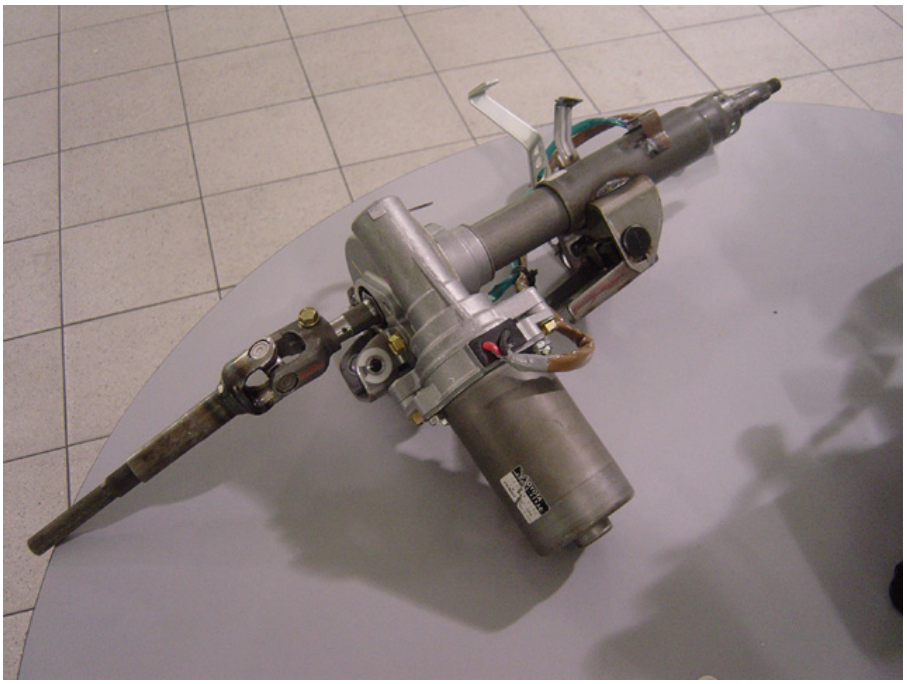
Åren gick och utvecklingen av bilarna med det. Ett problem med att köra tyngre fordon var att det krävdes en stor kraft från föraren för att styra fordonet vilket gjorde dem svårstyrda. År 1876 installerades det första styrservot av G.W. Fitts. Detta servo utnyttjade hävstångsprincipen och gav föraren av fordonet assistans av styrningen. Men det var inte förrän 1951 som styrservot kom i produktion för den kommersiella marknaden. De som satte detta i produktion var Chrysler och de använde sig av ett system som var baserat på ett utgånget patent som Francis W. Davis uppfunnit. Systemet utnyttjade hydralik för att underlätta styrningen och servot installerades i deras Chrysler Imperial där den marknadsfördes under namnet Hydraguide [Pat10].

I moderna styrserveron kan elektriska motorer användas för att förstärka kraften från ratten till hjulen. För att styra motorerna finns det mjukvara som ger signaler baserat på olika insignaler från bilen (se figur 1.1). Sentient är ett företag baserat på Lindholmen i Göteborg som utvecklar sådana system.

Sentient har sex anställda och startades 2009 med en vision om att utveckla teknik för fordon och i synnerhet för autonoma fordon. I vissa moderna fordon finns system som inte bara förstärker styrningen utan även justerar styrningen beroende på hastighet, underlag eller andra parametrar. Denna typ av teknik är vad Sentient har utvecklat.

Systemet som Sentient utvecklat heter S+ Premium Steering, även kallat SPS. Det är byggt för att reducera yttre faktorers påverkan på styrsystemet. Detta för att ge föraren maximal styrkänsla och låta föraren hålla fullt fokus på vägen istället för att parera inverkan från de yttre faktorerna. För närvarande ligger styrsystemet i serieproduktion hos en av de största lastbilstillverkarna i världen.

Sentient har en testbil där det ordinarie styrsystemet är utbytt så att de olika systemen som utvecklats kan



Figur 1.1: En isärmonterad styrenhet från en Toyota Prius. Själva servomotorn syns i mitten av bilden. Staget till hjulen går ner till vänster och rattstången är den som går upp till höger. Foto: JMPerez, CC-BY-SA 3.0.

demonstreras. Systemen kan demonstreras genom att exempelvis koppla in en joystick eller andra funktioner. För att välja i vilket läge styrsystemet ska demonstreras och för att visa aktuella inställningar finns i bilen ett enkelt HMI. Detta består av en tvåraders skärm samt tre knappar för att utföra olika kommandon till styrsystemet. Interiören i bilen med deras HMI visas i figur 1.2.

1.1 Samhällsaspekter

Ett välutvecklat HMI kan förbättra den mänskliga miljön i fordon. Det kan ge en mer positiv upplevelse och en ökad användarvänlighet. Potentiellt kan det också reducera smärtor i till exempel axlar, vilket är ett utbredande problem i dagens samhälle. Detta genom den assistans som styrsystemet ger föraren av fordonet.

1.2 Syfte/mål

Sentient har uttryckt en önskan om att kunna använda ett mer funktionellt och användarvänligt HMI i sina testfordon. Deras önskningar har varit att tydligare kunna se aktuella inställningar och övrig information av fordonet. Utöver detta skulle mer avancerade inställningar för styrenheten kunna ändras via gränssnittet. De ville dessutom att systemet skulle kunna styras via röstkommandon likt Siri på Apple-enheter. Siri, vilket är ett program, där användaren via ett röstkommando frågar enheten om att utföra en viss uppgift [Inc16]. Funktionen med röstkommandon kommer att ses över men den funktionaliteten kommer endast att implementeras i mån av tid. Därför kommer detta ej att vara ett krav för examensarbetet.

Detta examensarbete går ut på att skapa ett nytt HMI för demonstration av styrsystemet. Detta genom att ersätta det befintliga HMI:t med en surfplatta, vilket underlättar ändring av inställningar och ger en tydligare presentation av information för användaren. För att systemet ska vara framtidssäkert ska systemet dessutom vara väldokumenterat och utbyggbart. Detta i form av att komponenter enkelt skall kunna bytas ut och att funktionaliteten skall vara modifierbar för framtida behov från Sentients sida.

Applikationen på surfplattan ska kunna koppla upp sig mot fordonets styrenhet för att läsa aktuella

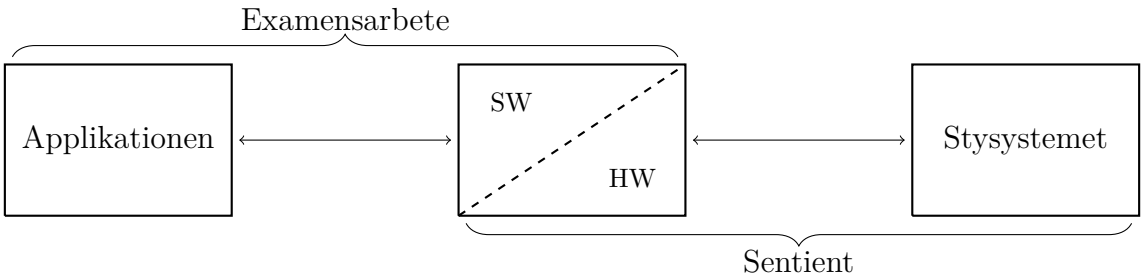
Figur 1.2: Inne i Sentients testbil med deras befintliga HMI:t som sitter vid vindrutan.

inställningar samt ändra inställningar i styrenheten. Denna uppkoppling ska ske trådlöst samt vara snabb utan några större fördröjningar.

Utöver detta är syftet med examensarbetet att vi ska lära oss om praktisk utveckling ute i näringslivet inom data- och informationsteknik, om olika tekniker användbara för vårt arbete samt hur man utformar och skriver vetenskapliga rapporter.

1.3 Avgränsningar

Projektet ska vara utbyggbart och väldokumenterat. Med detta menas att funktioner och grafiska komponenter enkelt skall kunna modifieras och läggas till utefter behov. Röststyrning kommer att implementeras och utvecklas i mån av tid. Övriga funktioner kommer ej att vara specificerade. Endast mjukvaruutveckling skall göras i detta projekt och hårdvaran tillhandahålls av Sentient (se figur 1.3).



Figur 1.3: Arbetsfördelning gentemot Sentient.

1.4 Problemformulering

Målet med projektet är att skapa en applikation till en surfplatta. Applikationen skall kunna styra inställningar i Sentients system för styrning av fordon, vara väldokumenterad samt utbyggbar för Sentients framtida behov. Applikationen ska både ersätta och överträffa nuvarande system bl.a. avseende på grafik, möjligheter till inställningar samt möjlighet för röststyrning. Applikationen skall ha ett enkelt och användarvänligt gränssnitt och möjlighet till att enkelt spara och ändra information, så som texter och felkoder.

Applikationen har även som krav att ge snabb respons så att eventuella statusmeddelanden från styrenheten visas inom 200 ms.

2 Teknisk bakgrund

I detta avsnitt beskrivs de olika tekniker och verktyg som har varit centrala för hur detta projekt har byggts upp och fungerar. Presentation av de olika kommunikationsmetoder som använts inom detta projekt kommer att förklaras. Dessa är CAN och dess protokoll för kommunikation i fordonet och Bluetooth för den trådlösa kommunikationen till surfplattan. I övrigt kommer hårdvara, mjukvara samt de olika metoder som använts att presenteras så som Arduino och programmeringsmönstret MVP.

2.1 Bluetooth

Bluetooth är en form av standard för trådlös kommunikation som har hand om uppkoppling mellan olika tekniska enheter. Dessa enheter kan vara hörlurar som kopplar upp sig mot mobiltelefoner eller en surfplatta som kopplar upp sig mot en Arduinoenhet. Bluetooth låter sedan dessa kommunicera fritt emellan varandra. Totalt kan mellan två och åtta enheter vara uppkopplade samtidigt [Blu16].

Historien bakom Bluetooth har sin början på Ericsson 1995 där den svenske Lundingenjören Sven Mattisson och hans holländske kollega Jaap Haartsen fick i uppdrag att utveckla ett koncept för trådlös kortdistans kommunikation mellan mobiltelefoner. Konceptet blev snabbt uppmärksammat och 1997 gick Intel med Jim Kardach, chef för teknisk utveckling, i spetsen in i projektet och redan i maj året efter lanserades tekniken under namnet Bluetooth. Namnet Bluetooth hade de tagit från den danska kungen Harald Blåtand som regerade mellan år 940 och 986 [Sve16].

Bluetooth använder sig av radiovågor och en master/slave princip för att kommunicera mellan olika enheter. För att kommunikation skall kunna upprättas måste enheterna innehålla ett antennutrustat Bluetooth-chip och mjukvara för att koda av kommunikationen som skickas mellan dem. För att länka ihop enheterna måste enheterna först paras. Detta görs genom att enheterna utbyter en nyckel, så kallad link key, som är likadan i båda enheterna. Är dessa kriterier uppfyllda kan uppkoppling mot varandra göras och information kan börja utbytas [Blu16].

Tekniken använder ISM-bandet som ligger på frekvenser mellan 2,400 och 2,485 GHz. Vid användning producerar Bluetoothchipen våglängder som är bundna till detta frekvensspektrum som speciellt är skapat för kortdistans kommunikation. Men det är inte bara Bluetooth som använder sig av dessa frekvenser. Även t.ex. trådlösa telefoner och andra typer av trådlösa enheter kan ligga på detta band. För att inte flera enheter skall ligga på samma frekvens samtidigt, vilket kan leda till att signalerna stör ut varandra, hoppar Bluetooth mellan frekvenserna 1600 gånger/sekund. Detta gör att signalerna inte kan störa ut någon annan signal i mer än 1/1600 sekund [Bou14].

Bluetooth är skapat för kortdistans kommunikation och har som lägsta krav att klara en distans på ca 10 meter. Det är sällan kommunikation kan ske över större distanser än 100 meter[Bou14]. Bluetooth 2.0 kan överföra data med hastigheter kring 700 kb/s och i de nyaste protokollen, vilket är version 4.2, med flera Mb/s [Kew04].

Fördelen med att använda Blutooth som kommunikationsstandard är att det redan finns implementerat i miljonstals enheter, kräver lite ström och är billiga att producera [Blu16].

2.2 CAN

Moderna fordon består av en stor mängd komponenter där många är i behov av att kommunicera information mellan varandra. Därför utvecklades på 1980-talet ett system som möjliggör sådan kommunikation. Systemet fungerar utan att det finns någon värdenhet som styr kommunikationen och utan mängder av kablar kopplade mellan alla komponenter som behöver kunna kommunicera. Systemet som utvecklades av Robert Bosch Gmbh, även kallade Bosch, lanserades 1986 och går under namnet CAN. Systemet är sedan 1993 en ISO-standard (ISO 11898) med den nyaste versionen från 2015 [CAN16b][15a].

CAN medger datahastigheter på upp till 1 Mb/s på avstånd upp till 40 m och upp till 125 kb/s på avstånd upp till 500 m. För att skicka data mellan de olika komponenterna använder sig CAN av två virade kopparledningar, vilka är seriekopplade mellan de olika komponenterna [15b].

Meddelanden på CAN kan skickas på två sätt, antingen med ett extra långt adressfält (29 bitar) eller med ett normalt adressfält (12 bitar). CAN-meddelanden skickas synkroniserat där meddelanden först innehåller en startbit och därefter en adress vilken läses samtidigt som den skickas. Om meddelanden med olika adresser

skickas samtidigt kommer enbart meddelandet med lägst adress transmitteras. Detta sker genom att den som sänder meddelande med en högre adress slutar skicka när den detekterar att ett annat meddelande med lägre adress samtidigt skickas. Efter de första 11 bitarna av adressen skickas en bit som avgör om meddelandet är ett längre meddelande eller ett normalt meddelande. Meddelandena innehåller därefter hur många byte data de innehåller, vilket är ett tal mellan noll och åtta [Rob91].

För kontroll av meddelanden så att överföringen säkerställs används en 15 bitars kontrollsumma. Detta för att reducera sannolikheten att korrupta och felaktiga meddelanden skickas och tas emot på CAN-nätverket [CAN16a].

2.3 Arduino

Arduino är ett företag som sedan 2005 erbjuder billiga microkontrollerenheter till vilka det finns en stor mängd extra komponenter som kan anslutas [Bar16]. Bland annat enheter för kommunikation via Bluetooth och CAN.

Arduinos microcontrollerbaserade kretskort kan programmeras med C/C++ och kan kopplas till en dator via USB [Ard16a]. För att programmera den finns det ett gratisprogram som kan laddas ner. I programmet kan kod skrivas, kompileras och föras över till enheten. Traditionellt sett har korten varit baserade på Atmels mikroprocessorer med 8, 16 eller 32 bitar, men sedan 2015 finns även kort med andra processorer. Den enklaste Arduinoenheten på marknaden utanför USA är Genuino Uno [Ard16b]. Den är baserad på en 32 bitars Atmelprocessor (ATmega328) och har 14 digitala in-/utgångar samt 6 analoga ingångar.

Genuino Uno har inbyggt hårdvarustöd för seriell kommunikation som används till USB-ingången. För kommunikation över de andra digitala in-/utgångarna finns ett mjukvarubibliotek som heter SoftwareSerial. Detta ser programmeringsmässigt likadant ut som den seriella kommunikationen med hårdvarustöd. Den enda skillnaden är att biblioteket SoftwareSerial används istället för biblioteket Serial, som används för den hårdvarubaserade seriella in-/utgången [Ard16c].

För att kommunicera via Bluetooth från en Arduinoenhet kan ett kretskort med en HC-06 kopplas in. Detta är en liten enhet som kan kommunicera seriellt med en ingång och en utgång [Sty13].

För att kommunicera på CAN kan en så kallad CAN-sköld kopplas in. Detta är ett kretskort som placeras ovanpå Arduinoenheten med hjälp av stift och som då sitter likt en sköld över Arduinoenheten. På kretskortet finns en kontakt där CAN-nätverket kan kopplas in. CAN-skölden kopplas in på alla Arduinons in-/utgångar men använder endast sex av dem. Resten av in-/utgångarna blir som en förlängning av Arduinoenhetens in-/utgångar [Spa16]. En enhet med både CAN-sköld och Bluetoothenhet visas i figur 2.1.

2.4 HMI

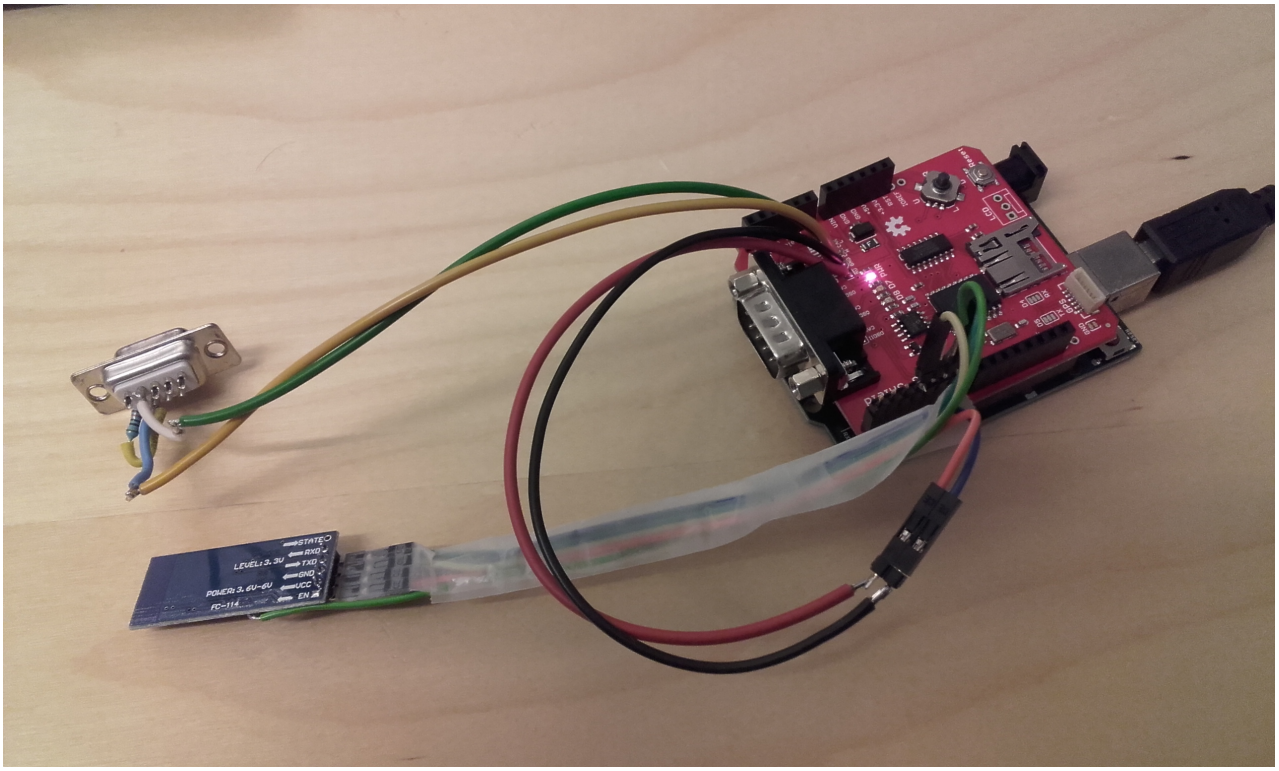
Likt kommunikation mellan människor där information utbytes och löften ges behöver människan, speciellt i dagens tekniksamhälle, kunna kommunicera på ett enkelt sätt med maskiner. Kommunikationen måste ske på ett tillfredsställande och enkelt sätt, där information och interaktion kan utläsas och utbytas likt en dialog mellan oss människor. Det kan ske genom att använda ett visuellt gränssnitt, såsom en surfplatta, där information flödar ut till användaren och där användaren använder sig av skärmen för att kommunicera med maskinen.

HMI är ett sådant gränssnitt som låter användare interagera med maskiner. Gränssnittet består av hårdvara och mjukvara där signaler mellan maskinen och gränssnittet skickas och omvandlas antingen till information för användaren eller kommandon för maskinen. Detta ger, för användaren, en bättre bild av vad som sker i systemet och gör det enklare för användaren att kontrollera systemet som körs av maskinen. Detta genom att information och grafik uppdateras i realtid utefter de signaler som skickas mellan HMI:t och maskinen [Tec16].

Förutom funktion skall HMI:t vara utformat på ett ergonomiskt sätt så att användaren både fysiskt och psykiskt lätt skall kunna förstå och använda detta som medium [Tec16]. Några av fördelarna med att använda ett HMI är ökad användarvänlighet, minskad risk för misstag, ökad effektivitet etc.

Andra fördelar med att använda sig av ett HMI är att genom informationen, som skall vara enkelt accessbar, kunna dra enkla och snabba slutsatser. Denna information kan vara i form av temperaturer, varvtal, förbrukning etc. och hjälper användaren att fatta klokare och snabbare beslut när systemet körs [Ste16].

Genom att kunna få ut snabb och precis data kan detta leda till ekonomiska och mer hållbara fördelar. Detta då det blir möjligt att enklare se hur systemet agerar och fungerar i olika situationer. Därefter kan systemet optimeras via t.ex. olika styrdon. Kan denna information dessutom sparas kan bättre beslut fattas utifrån all uppsamlad data [Ste16].



Figur 2.1: Arduinoenheten med CAN-sköld och Bluetoothenhet.

2.5 MVP

System som består av olika delar kan modeleras för att ge en bättre översikt av vad de olika delarna i systemet ska ha för ansvarsområden. En sådan modell kan vara MVP vilken bygger på MVC [Pot96]. MVP står för Model-View-Presenter och bygger på tre olika delar precis som MVC. Delarna är dels en modell (Model), vilket är enheten som definierar datan som ska behandlas. Dels är det en vy (View), som är enheten vilken användaren ser och interagerar med. Slutligen är det en presenterare (Presenter), vilken skiljer sig åt från kontrolleraren i MVC-mönstret. Skillnaden består i att en kontrollerare har styrfunktion i systemet till skillnad från presenteraren som enbart vidareförmedlar information. Detta genom att presentera informationen från vyn för modellen på lämpligt sätt och på samma sätt med informationen från modellen som då presenteras för vyn. En skiss över detta system visas i figur 2.2.

MVP har sitt ursprung i en önskan att skilja på data och interaktion [Pot96]. Det som i MVC handlar om modellen har önskats skiljas från kontrolleraren och vyn. Gränssnittet mellan de två delarna blir en presenterare som ersätter kontrolleraren. Därmed kan man också minska interaktionen som går direkt mellan modell och vy.

Figur 2.2: Uppbyggnad av MVP (Model-View-Presenter).

2.6 Android

I Oktober 2003 startade Andy Rubin, en före detta Appleingenjör, företaget Android Inc. Företaget, som de första två åren inte ens hade en produkt, finansierades helt av Andy Rubin själv och hade som vision att skapa framtidens mjukvara till smarttelefoner. Han fokuserade på att mjukvaran skulle vara lätt att koppla upp mot webben och använda en sådan miljö så att utvecklare lätt skulle kunna bygga sina egna applikationer. Detta genom att ge utvecklarna tillgång till ursprungskoden. 2005 kom det stora genombrottet då Google, som behövde en plattform till sin satsning på den mobila marknaden, köpte företaget och anställde Andy och hans team för vidare utveckling av plattformen [16b]. Idag står Android för cirka 84 % av operativsystemen till mobiltelefoner [Sta16].

Den första telefonen som använde sig av Android som operativsystem var HTC telefonen Dream, vilken släpptes 2008. Där användes Android 1.0 [16a]. Sedan dess har en rad uppdateringar kommit och de har gått ifrån versions numret som primärt namn till namn associerade med godsaker. Android finns idag i både mobiltelefoner, surfplattor och smarta klockor [And16c].

Android är känt som ett operativsystem byggt på öppen källkod. Vem som helst, från stora företag till hobbyprogrammerare, kan ladda ner källkoden till operativsystemet och bygga egna versioner av operativsystemet. När en ny version av Android är släppt för marknaden släpps källkoden genom AOSP, ett socialt nätverk där man kan ladda upp versioner av Android och ladda ner andras versioner [16a].

Androidapplikationer utvecklas huvudsakligen i Java. Vilket Google rekommenderar då deras API:er huvudsakligen är designade för Java. Dock går det att skriva Androidapplikationer i andra språk, såsom C och C++, men stöden för dessa språk är begränsade. UI:t byggs huvudsakligen genom XML som sedan kan modifieras genom Java-koden [Sim16].

För interaktion använder Android bland annat så kallade aktiviteter och fragment. En aktivitet är en komponent som ger användaren ett fönster, vilket oftast täcker hela skärmen. I aktiviteten kan användaren interagera men de komponenter som finns skapade i aktiviteten. Varje aktivitet har en egen livscykel. Vid byte av aktivitet, t.ex. när en vy byts och en ny aktivitet presenteras, förstörs eller pausas den tidigare aktiviteten. Pausas aktiviteten läggs denna i en så kallad back stack som fungerar som en LIFO-kö i minnet. När användaren sedan går tillbaka, genom att t.ex. trycka backpå telefonen eller surfplattan, hämtas den förregående aktiviteten och kallar på funktionen onResume som ingår i dess livscykel [And16a].

Ett fragment liknar en aktivitet. Den har en egen livscykel precis som en aktivitet och den hanterar sina egna komponenter som finns i fragmentet. Ett fragment fungerar som en subaktivitet och måste finnas inbäddad i en aktivitet. Förstörs eller pausas en aktivitet så förstörs eller pausas även dess fragment. En stor fördel med att använda fragment är att de är enkla att återanvända och återskapa, eftersom de har en egen livscykel och innehåller egen logik. Det går att använda flera olika fragment inuti en aktivitet. Detta gör att det är möjligt att enkelt dela upp skärmen i behållare och på så sätt enkelt strukturera upp layouten utefter de behov som finns [And16b].

För utveckling av Android kan en rad olika program användas. Det mest kända och mest använda är Android Studios. Det är baserat på Jetbrains IntelliJ IDEA, vilket är en redigerare och kompilator för Java. Android Studios är gratis och är speciellt utformat för utveckling av Androidapplikationer [Stu16].

3 Metod och genomförande

Vi har under detta arbete använt en typ av Scrum-metod [Scr16] där vi planerat veckovis och där mål och arbetsuppgifter har delats ut. Detta har antingen skett i början på varje vecka eller i anslutning till att Sentient velat ha ny funktionalitet i applikationen. Sedan har en utvärdering gjorts i slutet på varje vecka.

Utvecklingen har bedrivits i nära kontakt med Sentient på plats i deras lokaler. Detta för att kunna få snabb feedback på vårt arbete. Även testbilen, en Toyota Yaris, har funnits vid detta kontor vilket har underlättat vid skarp testning.

Innan arbetet påbörjades gjordes en grov planering med veckovisa milstolpar vilken redovisas i tabell 3.1.

För att få en så snabb utveckling av applikationen som möjligt har utvecklingen av surfplatta och Arduinoenhet skötts parallellt, med start från dag ett. För att fördela arbetet har vi delat upp ansvaret så att en har ansvarat för Arduinoenheten och en för surfplattan. För att välja vem som skulle ansvara för vilken del av utvecklingen togs erfarenhet och kunskap in som parametrar. Vi har dock hjälpt varanndra då detta har behövts och då i synnerhet i utvecklingen av applikationen som varit huvudsyftet med projektet. För utvecklingen på Arduinoenheten har vi haft hjälp av personal på Sentient som haft stor erfarenhet av att arbeta med sådana system. Utvecklingen av Androidapplikationen har bedrivits i Android Studios och för Arduinoenheten har kod skrivits i Arduinos egna utvecklingsmiljö.

Arbetet har utvärderats löpande. Detta har i huvudsak skett med feedback från anställda på Sentient, men även med hjälp av CAN-lyssnare och genom testning i Sentients demonstrationsfordon. Det har gjort att vi har kunnat testa att ett tillfredsställande beteende uppnås i gränssnittet, att kommunikationen fungerar mellan de olika lagren/enheterna och att styrsystemet reagerar på de signaler som skickas och tas emot.

Applikationen har även visats upp för Sentients externa kunder vilka har varit placerade både inrikes och utrikes. Detta för att acceptanstesta applikationens användbarhet samt för att få feedback för framtida utveckling.

Tabell 3.1: Veckoplanering innan projektets uppstart.

Kalendervecka	Innehåll
14	Möte, uppstart
15	Val av surfplatta och kommunikationsform. Under veckan kommer vi ha behov av att få hjälp med hur kommunikationen kan ske gentemot bilen. Vi behöver också få hjälp med hur befintlig kommunikation fungerar. Samt att ordna en kommunikationsenhet mellan surfplatta och bil.
16	Fixa kommunikationen mellan surfplatta och bil. Under veckan kommer vi ha behov av att testa och implementera kommunikationen till och från bilen.
17	Fixa kommunikationen mellan surfplatta och bil. Under veckan kommer vi ha behov av att testa och implementera kommunikationen till och från bilen.
18	(Kristi himmelsfärd) Skicka kommandon mellan surfplatta och bil
19	Skicka kommandon mellan surfplatta och bil. Under veckan kommer vi ha behov av att bekräfta att kommunikationen sker så som tänkt.
20	Ordna en enklare prototyp
21	Testning
22	Finjustering och utvärdering
23	Deadline inför opponering
24	Opponering

4 Vald teknik

Innan valen av plattform och kommunikationsform gjordes, undersöktes via vilket medium som kommunikationen skulle skötas, vilken hårdvara som skulle användas och för vilket operativsystem applikationen skulle utvecklas.

För trådlös kommunikation valdes Bluetooth eftersom detta är en känd teknik både för oss och för Sentient och en teknik som vi visste var möjlig att implementera. Det enda trådlösa alternativet vi såg skulle vara att använda Wi-Fi, men det var inget som kom på tal av Sentient och utvärderades därför inte. Ett annat alternativ till Bluetooth hade kunnat varit att använda trådbunden kommunikation som var säker och som vi visste skulle fungera. Det beslutades att vi skulle försöka med Bluetooth och endast om det av någon anledning inte skulle fungera tillfredsställande skulle trådbunden kommunikation användas.

Däremot diskuterades vilken typ av surfplatta och operativsystem som skulle användas. Alternativet till Android var att utveckla på en Apple-enhet med IOS som operativsystem. Detta borde rent funktionsmässigt fungera lika bra men var inget som vi vidare har fördjupat oss i. Dels hade vi tidigare erfarenheter av att skapa applikationer på Android, vilket innebar att inlärningstiden kunde kortas ner och dels att det finns röstigenkänning i Android, vilket kan användas av andra applikationer även utan någon Internetanslutning. Vi hade också upplevt att Android är mer öppet för utveckling. Detta genom att låta oss som utvecklare modifiera komponenter utefter behov och ge oss tillgång till stora delar av systemets funktioner. Något som skulle vara positivt då potentiella komplikationer lättare kan lösas. Därför föll valet på att använda Android då utveckling av applikationen fortare kunde påbörjas och resultaten snabbare kunde utvärderas.

När valet av operativsystem var klart utvärderades vilken surfplatta som skulle användas. Då Sentient ville ha en surfplatta med en skärmstorlek på 7 tum föll valet på en Asus ZenPad C 7.0. Denna uppfyller de krav vi hade, så som mikrofon och Bluetooth[Asu16]. Den är också designmässigt tillfredställande och låg inom vår budget.

Valet av hårdvara för kommunikationslagret föll på en Arduino, men det diskuterades även om att använda en Rasberry Pi. Raspberry Pi överträffar Arduino i processorkraft och därmed snabbhet samt att den är ungefär lika billig. Men även här föll valet på den hårdvara som vi hade mest erfarenhet av och som även de anställda på Sentient hade erfarenhet av och som de visste fungerar.

Flera olika sorters design med knappar placerade på olika platser och med olika lager provades ut. Även olika färgkombinationer och typer av bilder provades. Då inget tidigare HMI hade motsvarande möjligheter kunde vi arbeta under ganska stor frihet och själva välja vad vi tyckte var snyggt och hur applikationen grafiskt skulle vara utformad.

5 Resultat

Under detta projekt har ett HMI utvecklats, vilket består av en applikation på en surfplatta, baserad på operativsystemet Android, som kommunicerar med styrsystem i fordon (se figur 5.1). Utöver detta har även utveckling av kommunikationen mellan de olika enheterna behövt utvecklas. Detta har gjorts med hjälp av en Arduino Uno som sköter kommunikationen mellan fordonet och surfplattan. Kommunikationen görs via Bluetooth och fordonets CAN-nätverk. Utöver detta så har ett protokoll för överföring inom kommunikationen över både CAN och Bluetooth tagits fram.

I projektet har vi använt ett MVP-mönster. Fordonet fungerar som Model, då det är härifrån all data skickas och tas emot. Arduinoenheten fungerar som en Presenter då denna sköter kommunikationen mellan surfplatta och bil. Applikationen på surfplattan, där datan presenteras, fungerar som View.

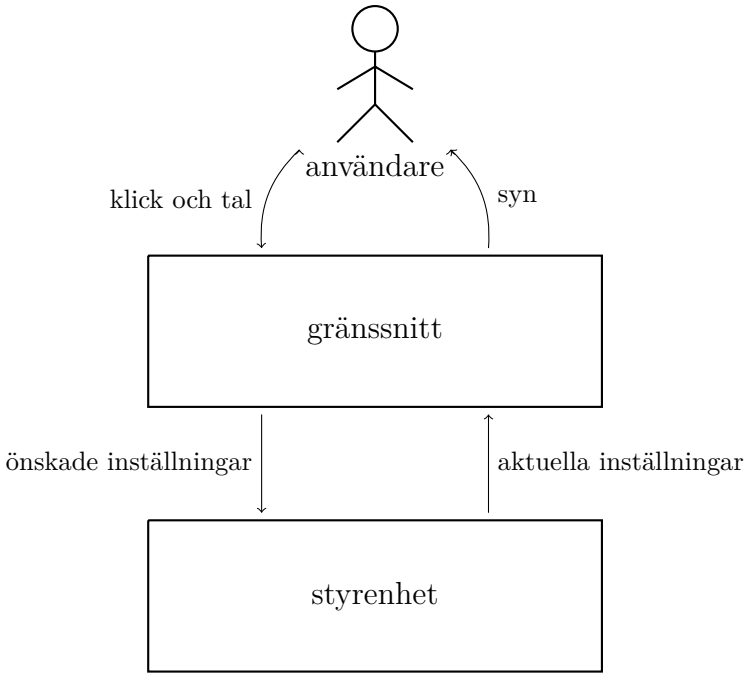
5.1 Kommunikation

För överföring mellan Arduinoenheten, som kopplats till fordonets CAN-nätverk, och applikationen på surfplattan används Bluetooth. Detta möjliggör trådlös kommunikation med, för projektet, tillfredsställande hastighet.

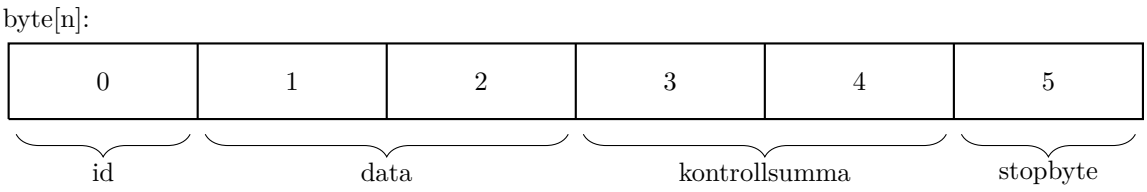
För överföring på Bluetooth, där meddelanden kan korrumperas, skapades ett protokoll med en fix längd på sex byte. En byte används för identifikation av meddelandet, två byte för data, två byte för checksumma och en för stopbyte (se figur 5.2). Identifikationsbyten används för att avgöra vad för information de två byten med data innehåller och hur denna data ska avkodas. De två byten med checksumman beräknades med hjälp av fletcher16 från identifikationsbyten och de två byten med data. Stopbyten utgörs av Carriage return, vilket krävs för att meddelandet ska skickas från Arduinoenheten.

Meddelandena som inkommer på fordonets CAN-nätverk har upp till åtta byte information, som därför måste delas upp i upp till fyra meddelanden.

De olika data som överförs från fordonets styrsystem är aktuella inställningar, vilka dels utgörs av åtta tal inom intervallet 0-100 samt av upptill 32 booleska värden. Dessutom skickas aktuell status som ett värde inom intervallet 0-3 och vid fel en två byte lång felkod.



Figur 5.1: Uppbyggnad av vårt HMI. Gränssnittet i bilden är det vi har utvecklat som utgörs av flera lager. Från styrenheten och uppåt av: Arduinoenhet, Bluetooth, applikation, skärm och mikrofon.



Figur 5.2: Diagram över innehållet i paketen som skickas över Bluetooth.

På CAN-nätverket skickas informationen för två olika meddelanden. Båda meddelandena skickas med ett intervall på 25 ms. Det innebär att det i snitt är 12,5 ms mellan varje mottaget meddelande. Det ena meddelandet innehåller de åtta olika talen med en byte per inställning. I det andra används de första två byten för aktuell status, därefter fyra byte med de 32 booleska inställningarna och sist två byte med felkod vilken endast läses om aktuell status är i felläge.

Datan som inkommer på CAN skickas vidare över Bluetooth med olika id. Om datan som inkommer skiljer sig gentemot förra meddelandet skickas det omedelbart. Annars skickas meddelandena periodiskt med 50 ms mellan varje meddelande med ett visst id. Meddelandet med aktuell status som endast innehåller något av värdena 0-3 byttes ut mot värdet 4 om inga meddelanden inkommit på CAN-nätverket under mer än 100 ms.

Ett meddelande från applikationen utgörs av begärda förändringar av inställningar av ett tal eller ett booleskt värde. Det skickas över Bluetooth med olika id beroende på om det är ett tal eller ett booleskt värde som ska förändras. I meddelandet används den första byten för att avgöra vilken inställning som begärs och den andra för att meddela vilket värde som begärs. På CAN skickades detta som ett meddelande med tre byte. Den första byten används för att begära en förändring av booleska värden där 6 bitar används för att ange vilket värde som ska ändras och två bitar används för det nya värdet. De två andra byten används för att ändra tal där en byte används för att ange vilket tal som ska ändras och en byte för att ange det nya talet. Om ingen ändring begärs för booleska värden och/eller tal skickades enbart ettor vilket innebär en felaktig inställning som ignoreras.

På CAN skickas meddelanden vid begärda ändringar omedelbart och därefter periodiskt med 25 ms mellan varje meddelande. På Bluetooth skickas på samma sätt men med 50 ms mellan varje meddelande.

5.2 Applikation

Applikationen som utvecklats kommer att presenteras här. Det inkluderar uppbyggnaden av applikationens olika funktioner, hur rösthanteringen fungerar och är uppbyggd samt hur layout är utformad och med vilka hjälpmedel som applikationen har byggts upp med.

5.2.1 Uppbyggnad av applikation

Applikationen körs på en surfplatta baserad på operativsystemet Android. De krav som ställts på surfplattan är att den ska klara av att hantera dataöverföring via Bluetooth, ha en skärm på 7 tum och att mikrofon finns.

Applikationen är uppbyggd med ett MVC-mönster. Den innehåller en aktivitet och olika fragment, vilka utför olika kommandon beroende på användarens knapptryckningar eller vad användaren via röst säger till applikationen att göra. Aktiviteten fungerar som controller i MVC-mönstret och sköter kommunikationen mellan fragmenten och underliggande klasser, så som hanteringen av kommunikationen via Bluetooth och hanteringen av röststyrningen. All inkommande och utgående kommunikation till och från Arduinoenheten tas emot respektive skickas via en särskild Bluetoothklass. Inkommande kommunikation skickas vidare från Bluetoothklassen till en handler, vilken fungerar som en egen tråd, i aktiviteten. Handlern sorterar ut dessa signaler och utför, utefter vilken signal som tagits emot, ett byte av vy eller dylikt. Utgående signaler skickas från fragmenten via aktiviteten till Bluetoothklassen som i sin tur skickar ut dessa via Bluetooth till Arduinoenheten.

Applikationen fungerar som en avspegling av styrsystemet. Visning av hur de olika komponenterna presenteras bestäms utifrån de olika signaler styrsystemet skickar till applikationen. Till exempel ställs inga knappar i läget pääm inte en signal som bekräftar detta inkommit till applikationen från styrsystemet. Samma sak gäller de reglage som används. Signaler skickas via Bluetoothklassen till fordonet vid byte av önskat värde, men reglagen ställs inte förrän värdet blivit godkänt av fordonet och det skickats tillbaka till applikationen. När

olika inställningar görs i applikationen skickas signaler utan att något genast ändras i vyn. Det görs först när inställningen är gjord i systemet och systemet meddelat det. Vid normal användning går detta så fort att det upplevs som direkt. Ett flödesdiagram över hur signalerna skickas visas i figur 5.3.

För att veta vilken komponent som avses har varje komponent tilldelats ett unikt id som hör ihop med en viss egenskap i styrsystemet. Vid varje interaktion, som leder till att en förfrågan görs mot styrsystemet, skickas detta id, samt vilken förfrågan som applikationen vill göra. Exempel på detta är om användaren vill slå på eller av en funktion eller modifiera ett värde i inställningarna.

5.2.2 Röststyrning

För att hantera röststyrning används Googles röstfunktion, vilken finns inbyggd i Android. Genom att lyssna på röstanrop skapar den en sträng som utnyttjas för att skicka rätt signaler för rätt kommando genom att matcha mot två redan förbestämda mappar innehållande ord. Mapparna som använts är HashMaps och dessa har fyllts med ord som används för att matcha mot de ord som fås genom röstfunktionen. Den första HashMappen innehåller kommandoord och den andra HashMappen innehåller styrord vilka skickas om inkommande röststräng matchar ett ord i varje mapp.

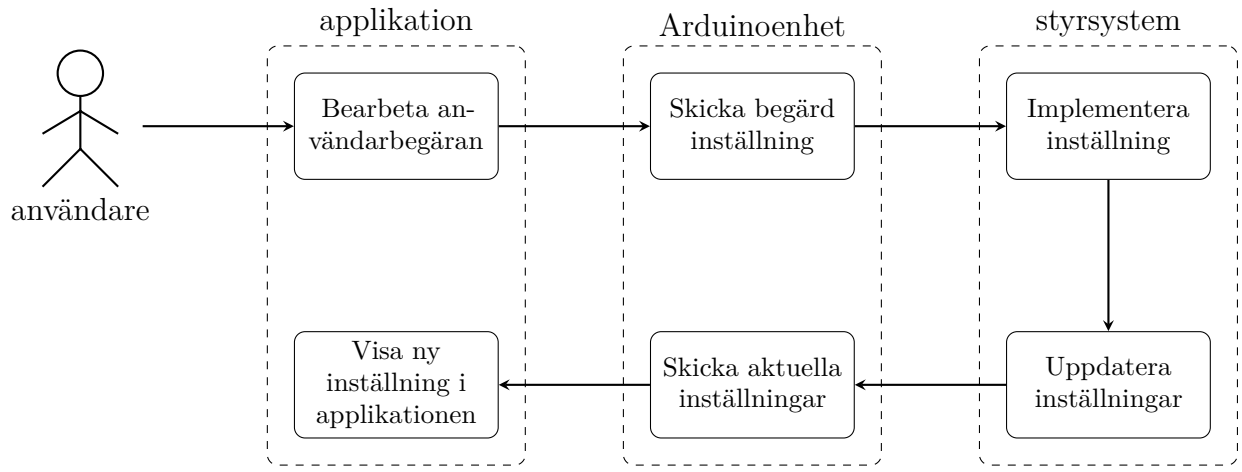
För att hantera röststrängen bryts den ner till ord genom att klippa meningarna vid varje mellanslag och orden läggs i en lista. Listan gås igenom för att först se om något av orden matchar de förbestämda kommandoorden, för att sedan se om något av de nästkommanade orden matchar något av de förbestämda styrorden. Om dessa två kriterier uppfylls returneras en sträng till aktiviteten som, via Bluetooth, gör en förfrågan till styrsystemet om att ställa sig i det läge som önskats. Om kriterierna inte uppfylls returneras null och ingen förfrågan skickas.

Ett problem som uppkommit i samband med röststyrning är att sagda ord lätt blir fel och tolkas som ord som liknar de som pratas in. För att få en mer tillförlitlig röstfunktion har därför många närliggande ord implementerats i mapparna där de returnerar samma sak som de avsedda orden.

5.2.3 Layout

Layouten är skapad via XML. Med hjälp av inbyggda komponenter och komponenter som behövts modifieras, bland annat för att hantera olika id, byggs layouten upp. För placering av komponenterna används olika så kallade containers.

Layouten är uppbyggd med en fast ram som täcker surfplattans övre del. Där styrs navigation och vilket innebär att man kan byta vy mellan bland annat röstigenkänning, information och inställningar (se överst i figur 5.4). Denna ligger hela tiden kvar och följer med oavsett vilken del av programmet som är aktivt. Övriga delen av skärmen består av olika fragment som ändras beroende på de signaler som bilen, via CAN-bussen, skickar ut eller av användarens interaktion med de knappar och skjutreglage som finns. Alla inställningar i



Figur 5.3: Flödesdiagram som visar händelserna när en användare använder vårt HMI. Enligt MVP motsvarar här applikationen vyn, Arduinoenheten presenteraren och styrsystemet motsvarar modellen.

Figur 5.4: Plattan med applikationen. Plattan är en Asus ZenPad C 7.0.

applikationen styrs av bilens signaler, som ställer applikationens knappar och skjutreglag till de värden som ges ifrån bilen, medan användarens interaktion skickar en begäran, i form av en signal, till bilen om att få ändra inställningar i denna. Godkänns denna begäran sätts komponentens värde till det önskade. Signaler skickas direkt när skjutreglage och krysslådor används och efter en sekunds nedtryck när knapparna för byte av läge används. Detta för att byte av de olika konfigureringsfragmenten skall kunna visas och användas utan att behöva byta läge i fordonets styrenhet.

Själva inställningsdelen av applikationen består av två olika vyer. Första vyn sköter inställning av vilken typ av enhet som styr fordonet, till exempel om fordonet skall styras med joystick eller normalt, och vilken sorts inställning som denna enhet skall ligga i. Detta presenteras via olika knappar och kryssrutor som kan sättas på och av.

Den andra vyn är till för inställningar av styrenheten (se figur 5.4). Dessa inställningar kan ändras genom att trycka på de knappar med förbestämda värden som finns eller att själv bestämma vilka värden som skall finnas genom att dra i de olika skjutreglage som finns.

För att förenkla designen och för att inte störa när surfplattan utsätts för g-krafter valdes en fast liggande layout. I övrigt har layouten försökts hållas så enkel och tydlig som möjligt så att det även vid körning skall vara möjligt att göra enklare inställningar utan att behöva fokusera på skärmen allt för länge. Designen är framtagen i samråd med Sentient.

5.3 Arduinoenhet

För kommunikation mellan styrenhet i bilen och applikationen utvecklades en enhet baserad på Arduino. Arduinoenheten är en Genuino Uno. Den är utrustad med en CAN-sköld baserad på chippet MCP2515 och en Bluetoothenhet baserad på en Linvor HC-06. Enheten kan kommunicera med styrenheten via CAN och med surfplattan via Bluetooth. Arduinoenheten är programmerad med C++ i Arduinos utvecklingsmiljö. För användning av CAN-skölden används biblioteket MCP2515_lib [Fow14] och för användning av Bluetoothenheten används biblioteket SoftwareSerial. Kommunikationen med Bluetoothenheten är satt till 57 600 bps som är den högsta hastighet Genuino Uno klarar av [Ard16c].

På grund av CAN:s relativt höga hastigheter och att bufferten på MCP2515 endast kan hålla två meddelanden används en huvudloop där inkommande meddelanden på CAN läses in efter varje annat kommando som tar mer än 100 µs. Att sända meddelanden på CAN uppmättes ta cirka 280 µs och mottagande uppmättes ta cirka 200 µs. Kommunikationen via Bluetooth är för att sända ett meddelande ca 1100 µs och för mottagande genom att meddelande tas emot bytevis cirka 100 µs. Övriga kommandon i enheten är i storleksordningen 10 till 30 µs. Detta innebär att enheten kan läsa meddelanden med 1,5 ms/meddelande som långsammast.

Då meddelanden på CAN kan komma i avsevärt högre takt än vad en Arduinoenhet kan hantera, används MCP2515:s hårdvarufilter så att endast relevanta meddelanden behövs läsas och så att bufferten inte fylls med irrelevanta meddelanden.

Arduinoenheten är inbyggd i en plastbox som enkelt kan kopplas in i fordon och strömförsörjas via CAN eller USB (se figur 5.5).

5.4 Acceptanstest

Applikationen har testats av och visats upp för såväl Sentient som externa kunder. Sentient har redan från början varit nöjda med det som levererats och varje vecka kommit med återkoppling och nya idéer och funktioner som sedan utvecklats vidare. Utvecklingen har även gjort att Sentient själva har vidareutvecklat sitt system utefter de idéer de fått från HMI:t och expanderat funktionaliteten i systemet.

Allt efter att applikationen haft högre funktionalitet än det befintliga HMI:t har krav önskats på ökad funktionalitet där Sentient då också, i och med att vår applikation utvecklats, fått öka möjligheterna i sin styrenhet att kunna göra motsvarande ändringar. Sentient har uttryckt entusiasm i att ha ett kompetent HMI som kan visa vad deras system gör och de har också genomgående uttryckt att applikationen uppfyller och övergår deras förväntningar. Applikationen har också sedan den haft högre funktionalitet än det befintliga HMI:t ersatt det så att Sentient slutat uppdatera det och enbart förlitar sig på vår applikation.

Sentients personal har testat HMI:t under färd både med röststyrning och med andra inställningar vilket gett dem direkt feedback på hur styrsystemet reagerar på vissa inställningar. Sentient kommer också att skicka

Figur 5.5: Arduinoenheten inbyggd i en låda för inkoppling i fordon med de två ingångarna synliga. CAN-ingången syns längst ner i bilden och USB-ingången till höger.

det till kund i Italien för testning i deras fordon.

Ett stort test gjordes i mitten av projektet då Arduinoenheten samt surfplatta med applikation skickades ner till en av Sentients mest etablerade och största kunder i Italien där utformning och funktion, förutom den röststyrda delen, visades upp för bl.a utvecklingschefer och VD för ett utlåtande och för att få återkoppling för framtida utveckling. Utlåtandet var positivt och alla parter som konceptet visades upp för var intresserade och såg gärna att detta koncept utvecklades vidare så att detta skulle kunna användas skarpt i deras fordon. Då detta är ett nytt koncept för hur systemet skall styras gavs inga direkta önskemål på hur produkten skulle kunna utformas och utvecklas utan detta kommer att vara något som de i framtiden kommer att kunna ge återkoppling på då produkten är färdigutvecklad och testad i fordon.

Ett annat test av HMI:t gjordes av fyra personer hos en stor kund till Sentient. Testet gick väldigt bra och alla funktioner som testades fungerade på ett tillfredställande sätt. Återkopplingen som gavs var genomgående bra. Att applikationen hade proffsig layout, inställningarna fungerade som förväntat och röststyrningen hade stor potential var några av lovorden som gavs ut. De påpekade för vidare utveckling att röststyrningen borde kunna bli säkrare om applikationen endast kan välja mellan en viss uppsättning kommandon istället för att som nu först göra om talet till en sträng som sedan jämförs med olika förbestämda kommandon.

6 Diskussion

Målet har varit att med en effektiv metod skapa ett HMI som kan uppfylla ställda krav i problemformuleringen. Fokus här i diskussionen är därför kring vårt val av metod, vårt val av lösning, vilket resultat vi uppnått och vilken potential vi ser inför framtida utveckling.

6.1 Metodval

Arbetet har både gett oss bredare teknisk kunskap och en större inblick i hur näringslivet fungerar. Vi har i detta projekt utvecklat en produkt av värde. Detta är något vi är väldigt nöjda med. När valet av projekt skulle göras så föll det ganska naturligt att välja detta projekt då vi båda lockades av den stora friheten i att skapa något nytt där kreativitet låg som huvudfokus och där en produkt skulle utvecklas. Detta har också genomsyrat vårt arbete där vi relativt fritt har fått välja metoder och layouter för att sedan få återkoppling på dessa. Något som också har styrt arbetet har varit vilka faktiska funktioner som ska kunna sättas på och av samt vilka värden som skulle kunna ändras.

En lärdom vi har tagit till oss är att vi i början av projektet spenderade mycket tid på att implementera en applikation som kunde ersätta det befintliga systemet fast med en utökad funktionalitet och med en snyggare och lättare presentation. När denna applikation med tillhörande kommunikationsenhet var utvecklad till den grad att den fungerade att använda i fordonet framkom nya idéer. Idéer som innebar att en helt ny version av applikationen började utvecklas och den första applikationen skrotades helt. Mycket tid gick här till spillo. Men å andra sidan fanns stora delar av grunden för ny utveckling av applikationen redan där och genom strukturerad kommunikation där scrummetoden användes kom vi fort igång med den nya versionen. Hade vi gjort om projektet borde vi lagt lite mer tid på att lära oss det befintliga systemet och därefter direkt börjat utveckla det nya vilket då skulle inneburit att vi även om den första tröskeln med mer inläsning hade blivit högre till slut hade sparat tid. Tid som hade kunnat användas till ytterligare utveckling av HMI:t.

Det har genom hela projektet funnits svårigheter med att kunna testa vårt system skarpt. Detta då systemet varit beroende på Sentients system och att alla ändringar i kommunikation mellan fordonets styrsystem och vår applikation har behövts implementerats parallellt. Då Sentient haft mycket att göra i andra projekt och vi endast behövt inrikta oss på detta projekt har det ibland blivit att vi kommit olika långt i utvecklingen och därför har testning ej kunnat utföras i den grad som behövts. Detta i kombination med nya önskemål av funktioner, från personal på Sentient, har gjort att vi aldrig haft något färdigt att kunna testa. När de implementerat kommunikationsprotokoll i styrsystemet har vi jobbat med de nya funktionerna som då inte kunnat testas. Vad som dock hjälpt har varit att de använt sig av beprövade protokoll och därmed haft i stort sett felfri kommunikation så att all testning enbart har behövts felsökas på det vi jobbat med. Dessa svårigheterna tror vi hade blivit än värre om vi inte hade arbetat på plats hos dem.

Själva arbetet hos Sentient tycker vi har fungerat bra. Sentient som företag har ett spännande koncept och många intressanta och roliga idéer som vi fått ta del av. Att vi fått chansen att vara en del av detta tycker vi har varit roligt. Personalen har varit mycket tillmötesgående. Det har aldrig varit något problem att få hjälp. De har förutom att dela med sig av sin tekniskkunskap även givit en inblick i teknikutveckling och allt som hör till detta. Något som varit mycket uppskattat och givande.

6.2 Teknisk lösning

Tekniska beslut har tagits i samråd med Sentient. Valet av Bluetooth gjordes tidigt och var enkelt. Det på grund av mängden enheter som använder Bluetooth och det faktum att det uppfyllde de på datahastighet som ställts. Då systemet endast skulle användas i en testbil trodde vi inte eventuella problem med störningar eller avlyssningar skulle vara sådana att någon mer avancerad teknik skulle krävas utöver en säker och korrekt signalöverföring. Vi upptäckte vid testningen dels att enstaka meddelanden korrumperades, dels att en del meddelanden trunkerades. Då det är signaler som bestämmer inställningarna i ett fordon som ska köras är det synnerligen viktigt att inga felaktiga signaler skickas och att inga oväntade inställningar utförs. Visserligen finns felkontroll i styrsystemet så att felaktiga inställningar ska vara omöjliga att utföra, men det är ändå mycket viktigt att inga meddelanden överförs med fel. För att undvika detta valdes en 16-bitars kontrollsumma. Detta innebär en risk att felaktiga data kan tolkas som korrekt är ungefär en på 65 tusen med slumpad data

vilket vi ansåg vara tillräckligt lite. Ungefär samma kontrollsummelängd (15 bitar) används också på CAN där det fungerat bra och varit pålitligt.

Arduino, vilket i vårt projekt valdes att användas, har varken speciellt mycket processorkraft eller minne vilket gör det svårt att använda dynaminska längder för dataöverföringen. Därför valde vi en fix längd av överförd data till två byte samt en byte för identifikation av datan. Då detta fungerade bra gjordes inga ytterligare försök att förbättra överföringen. Eftersom datan som överförs på CAN dessutom måste omkodas någonstans, till den data som skulle ställas in och visas, ansågs det vara lättast att göra detta med hjälp av Arduinoenheten. Då kan kommunikationen över Bluetooth hållas enhetlig och det är onödigt att anpassa den överföringen till hur datan skickas på CAN. Det förordades också av de anställda på Sentient som har lättare att sätta sig in i vår kod på Arduinoenheten, där de har stor erfarenhet, än i vår kod på Androidenheten som de tidigare inte arbetat med.

6.3 Utvärdering av resultat

Ett krav var att HMI:t skulle vara snabbt. Då Arduinoenheten inte är så snabb som vi önskat och då meddelanden på CAN kan komma i mycket höga hastigheter gjorde vi flera mätningar för att försöka optimera koden. Arduinoenheten kan enligt de mätningarna läsa meddelanden från CAN med en hastighet av ungefär fem meddelanden per millisekund om det vore det enda den gjorde. Det är inte tillräckligt vid maximal belastning på CAN men mycket snabbare än vad som behövdes för de två meddelandena som med filtret inkommer inom en tidsperiod på 25 ms. Även om detta systemet skulle missa ett meddelande skulle ett nytt inkomma efter ytterligare 25 ms. Att skicka ett meddelande på Bluetooth tar drygt en millisekund vilket var mycket snabbare än det tidigare HMI:t som krävde cirka 4 millisekunder för att uppdatera dess display. Så då det tidigare HMI:t fungerat bra och vårt är bättre ansågs alla tider bra och vårt HMI vara tillräckligt snabbt. Vid tester där signaler skickades från applikationen till styrenheten och tillbaka så kändes det omedelbart och var klart bättre än de ställda kraven på 200 ms.

Layouten av applikationen har bemöts mycket positivt av personal på Sentient. Vid användning av applikationen upplevs direkt respons i fordonet vilket också tydligt visar på hur interaktionen mellan fordon och förare knutits närmare. Detta var omöjligt med det HMI:t de tidigare hade eftersom man då var tvungen att bläddra i menyer innan olika inställningar kunde göras.

De acceptanstester som gjorts mot såväl Sentient som externa kunder har gått över förväntan. Återkopplingen vi fått från kunder till Sentient har varit genomgående positiv både vad det gäller det grafiska som funktionella. Den enda negativa återkoppling som gavs var angående röststyrningen som skulle kunna utvecklas vidare bland annat med snabbare och felsäkrare igenkänning. Detta är något som vi själva funderat över. Vi hade önskat att även få mer konstruktiv feedback. En anledning till att i stort sett endast positiv feedback har getts har vi funderat över och vi tror att då det tidigare HMI:t varit ett mycket stort lyft jämför med inget alls och då detta är ett motsvarande steg har det varit svårt att se något annat än enbart positiva delar. Detta har gjort att kunder tidigare varit nöjda med att få fler lysdioder och nu när ett helt nytt koncept för hur styrsystemet skall styras dykt upp har detta nästan blivit överrumplade och små eller stora brister eller förändringar därför vid detta tidiga stadie försummats eller inte upptäckts. Något som summerar denna tes är som en på Sentient uttryckte det: ”Hade vi frågat dem hade de endast önskat fler lysdioder”.

6.4 Framtida utveckling

Under arbetet har en hel del nya idéer om framtida funktioner växt fram. Både avseende vidareutveckling av redan befintliga funktioner och ny funktionalitet. En del av dessa idéer kommer här att presenteras.

Fordonet kan hamna i ett så kallat failsafe-läge vilket indikerar att något gått fel i systemet. Som det är nu så visas denna felkod som en kod som skickas från styrsystemet i ett meddelande som poppar upp. Koden sparas också i en textfil tillsammans med datum och tid. Tyvärr så kan användaren, i de flesta fall, inte veta vad felet betyder utan att kolla upp detta på annat håll. En idé var att koppla samman alla felkoder med förklarande strängar om vad felet verkligen betyder och sedan visa dessa beroende på vilket fel som inträffat. Detta för att lätt kunna felsöka systemet och därmed spara tid.

En annan idé som växt fram är att man genom att koppla upp sig mot ett fordon få upp vyn för precis det fordon som man är uppkopplad mot. Eftersom varje fordon har olika funktioner och därför olika layout skulle man behöva göra en applikation för var och en av fordonen så som bil, traktor, lastbil etc. En lösning på detta hade varit att bygga upp var och en av vyerna i applikationen och beroende på vilken signal som fordonet

skickar ut byta denna vy till den som matchar just det fordon som körs. En annan lösning hade varit att man manuellt via till exempel inställningarna kunnat byta till det fordon som man för tillfället kör. Detta hade gjort att man endast hade behövt en applikation för testning av alla fordon.

Andra områden som kommer att behövas utvecklas i framtiden är bland annat röststyrningen som idag fungerar och byter läge beroende på vilken indata surfplattan får men den är bara en prototyp för hur det skulle kunna fungera. Skall den fungera bättre så behövs denna funktion utvecklas genom att t.ex. göra röstfunktionen mer tolerant gentemot felsägningar/närliggande ord som ibland blir fel men även att kanske utveckla så att applikationen alltid lyssnar på föraren och att inga knapptryckningar behövs för att starta röstfunktionen.

7 Slutsats

Målen vi hade i detta arbete var dels att skapa ett HMI som skulle kunna kommunicera med styrenheten i Sentients demonstrationsfordon, dels att detta HMI skulle fungera på ett tillfredställande sätt. Säkert och snabb kommunikation med styrsystemet i ett fordon var ett krav för att HMI:t skulle kunna användas i skarp miljö.

För att lösa detta utvecklades en applikation som körs på en surfplatta. Applikationen kommunicerar via Bluetooth med en enhet baserad på Arduino till vilken vi har utvecklat mjukvaran. Arduinoenheten kommunicerar också över CAN med fordonets styrsystem och skickar och tar emot olika typer av begäran.

Utvärdering har gjorts av såväl oss själva som av Sentient och deras kunder. De har uttryckt hur nöjda de är visar att vår lösning fungerar väl och uppfyller ställda krav på såväl kommunikation, funktion och grafisk presentation.

Dessutom hade vi som mål att lära oss mer om olika lämpliga tekniker vilket är något vi upplever att vi gjort.

Referenser

[15a] *ISO 11898*. The International Organizatio for Standardization, 2015.

[15b] *ISO 11898-3:2006*. The International Organizatio for Standardization, 2015.

[16a] *Android Early Days*. 2016. URL: http://www.androidcentral.com/androids-early-days (hämtad 2016-05-24).

[16b] *Android Pre-History*. 2016. URL: http://www.androidcentral.com/android-pre-history (hämtad 2016-05-24).

[And16a] Android. *Activities*. 2016. URL: https://developer.android.com/guide/components/activities.html (hämtad 2016-05-27).

[And16b] Android. *Fragments*. 2016. URL: https://developer.android.com/guide/components/fragments.html (hämtad 2016-05-27).

[And16c] Android. *The Android Story*. 2016. URL: https://www.android.com/history/#/marshmallow (hämtad 2016-05-24).

[Ard16a] Arduino. *Arduino*. 2016. URL: https://www.arduino.cc/ (hämtad 2016-05-24).

[Ard16b] Arduino. *Arduino UNO & Genuino UNO*. 2016. URL: https://www.arduino.cc/en/Main/ArduinoBoardUno (hämtad 2016-05-24).

[Ard16c] Arduino. *SoftwareSerial Library*. 2016. URL: https://www.arduino.cc/en/Reference/SoftwareSerial (hämtad 2016-05-12).

[Asu16] Asus. *ZenPad C 7.0*. 2016. URL: https://www.asus.com/Tablets/ASUS_ZenPad_C_70_Z170C/specifications/ (hämtad 2016-08-23).

[Bar16] H. Barragán. *The Untold History of Arduino*. 2016. URL: http://arduinohistory.github.io/ (hämtad 2016-05-24).

[Blu16] Bluetooth SIG Inc. *Bluetooth technology basics*. 2016. URL: https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics (hämtad 2016-05-13).

[Bou14] B. Bourque. *This is how Bluetooth works, and no, it's not by magic*. 2014. URL: http://www.digitaltrends.com/mobile/how-does-bluetooth-work/#:Xyve-MzLig+VgA (hämtad 2016-05-25).

[CAN16a] CAN in Automation. *Cyclic redundancy check (CRC) in CAN frames*. 2016. URL: http://www.can-cia.org/can-knowledge/can/crc/ (hämtad 2016-05-16).

[CAN16b] CAN in Automation. *History of CAN technology*. 2016. URL: http://www.can-cia.org/can-knowledge/can/can-history/ (hämtad 2016-05-16).

[Fow14] C. J. Fowler. *MCP_CAN_lib*. 16 jan. 2014. URL: https://github.com/coryjfowler/MCP_CAN_lib (hämtad 2016-05-12).

[Inc16] A. Inc. *Siri*. 2016. URL: http://www.apple.com/ios/siri/ (hämtad 2016-05-24).

[Kew04] G. Kewney. *High speed Bluetooth comes a step closer: enhanced data rate approved*. 2004. URL: http://www.newswireless.net/index.cfm/article/629 (hämtad 2016-05-26).

[Lie00] J. H. Lienhard. "The Engines of Our Ingenuity: An Engineers Looks at Technology and Culture". 2000. ISBN: 9780195135831.

[Pat10] D. Patrascu. *History of the Steering Wheel*. 2010. URL: http://www.autoevolution.com/news/history-of-the-steering-wheel-20109.html (hämtad 2016-05-24).

[Pot96] M. Potel. MVP: Model-View-Presenter The Taligent Programming Model for C++ and Java (1996). URL: http://www.wildcrest.com/Potel/Portfolio/mvp.pdf (hämtad 2016-05-24).

[Rob91] Robert Bosch GmbH. *CAN Specification 2.0*. 1991. URL: http://www.bosch-semiconductors.de/media/ubk_semiconductors/pdf_1/canliteratur/can2spec.pdf (hämtad 2016-05-24).

[Scr16] Scrum Alliance. *MCP_CAN_lib*. 2016. URL: https://www.scrumalliance.org/why-scrum (hämtad 2016-05-18).

[Sim16] G. Sims. *I want to develop Android Apps – What languages should I learn?* 2016. URL: http://www.androidauthority.com/want-develop-android-apps-languages-learn-391008/ (hämtad 2016-05-24).

[Spa16] Sparkfun. *CAN-BUS Shield*. 2016. URL: https://www.sparkfun.com/products/13262 (hämtad 2016-05-24).

[Sta16] Statista. *Statista*. 2016. URL: http://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/ (hämtad 2016-05-24).

[Ste16] E. Stern. *7 benefits of integrating human-machine interfaces, historians*. 2016. URL: http://www.controleng.com/single-article/7-benefits-of-integrating-human-machine-interfaces-historians/2b8802e26c7cbb7c3fff68ff064fff7f.html (hämtad 2016-05-16).

[Stu16] A. Studios. *Meet Android Studio*. 2016. URL: https://developer.android.com/studio/intro/index.html (hämtad 2016-05-24).

[Sty13] E. Styger. *Using the HC-06 Bluetooth Module*. 2013. URL: https://mcuoneclipse.com/2013/06/19/using-the-hc-06-bluetooth-module/ (hämtad 2016-05-24).

[Sve16] A. L. Svenolof Karlsson. *The History Of Bluetooth*. 2016. URL: http://www.ericssonhistory.com/changing-the-world/Anecdotes/The-history-of-Bluetooth-/ (hämtad 2016-05-18).

[Tec16] Techopedia. *Human-Machine Interface (HMI)*. 2016. URL: https://www.techopedia.com/definition/12829/human-machine-interface-hmi (hämtad 2016-05-13).