Röststyrning av motorfordon med enkortsdatorn Raspberry Pi

Examensarbete inom Data- och Informationsteknik

ALLAN RIDHA
ÖMER SAHIN KESKIN

Institutionen för Data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2021

EXAMENSARBETE

Röststyrning av motorfordon med enkortsdatorn Raspberry Pi

En kommunikationslösning mellan enkortsdator och fordon med modulära egenskaper för anpassning av röststyrning.

ALLAN RIDHA
ÖMER SAHIN KESKIN

Institutionen för Data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2021
Examensarbete LMTX38

(Ska vara tom sida)

En kommunikationslösning mellan enkortsdator och fordon.

Examensarbete inom Data- och Informationsteknik
ALLAN RIDHA
ÖMER SAHIN KESKIN

Examinator: Jonas Duregård

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA / Göteborgs Universitet
417 96 Göteborg
Chalmers Lindholmen
Telefon: 031-772 1000

Institutionen för Data- och Informationsteknik
Omslag:
Omslaget är skapad av författarna själva.

Göteborg 2021

Sammanfattning

Denna rapport redogör för utvecklingen av en kommunikationslösning mellan enheter med hjälp av tal och textkommandon som skickas över internet. Målet med projektet är att utveckla ett program som ger möjligheten att styra ett motorfordon i en given riktning med hjälp av röst. Detta ska göras med hjälp av en enkortsdator Raspberry Pi. Programmet ska uppfylla specifika krav på tid och användarvänlighet samt följa klient-server modellen. Arbetet resulterades i en applikation som tar ett röstkommando och sedan översätter detta kommando till text, skickar över det översatta kommandot vidare genom internet till en server som sedan skickar vidare kommandot till det korresponderande fordonet. Raspberry Pi som sitter på fordonet anropar sedan korrekt metod för att utföra det kommando som begärts av användaren. Rapporten omfattar inte implementering av säkerhet eller avancerade funktioner för röststyrning såsom artificiell intelligens.

Nyckelord: raspberry pi, röststyrning, motorfordon, kommunikationslösning

Abstract

This report describes the development of a communication solution between devices using voice and text commands sent over the Internet. The purpose of the project is to develop a program that provides the ability to steer a motor vehicle in given direction using voice. This should be done using a Raspberry Pi one-card computer. The program must meet specific requirements for time and user-friendliness and follow the client-server model.

The work resulted in a software that takes in voice commands which then translates this command into text and forwards the translated command on through the internet to a server which then forwards the command to the corresponding vehicle. The Raspberry Pi sitting on the vehicle then calls the correct method to carry out the command requested by the user. The report does not cover the implementation of security or advanced voice control features such as artificial intelligence.

Keywords: raspberry pi, voice control, motor vehicle, communication solution

Förord

Denna rapport är ett examensarbete inom både hårdvara och mjukvaruutveckling på datateknikprogrammet vid Institutionen för Data- och Informationsteknik på Chalmers Tekniska Högskola.

Examensarbetet utfördes för Chalmers Tekniska Högskola och beskriver utvecklingen av en kommunikationslösning för röststyrning av små eller stora fordon genom användning av enkortsdatorn Raspberry Pi och stöd via nätet.

Särskilt tack riktas till:

Sakib Sistek, handledare på Chalmers Tekniska Högskola, för vägledning under examensarbetets genomförande.

Innehållsförteckning

Figurförteckning

Figurer:
Alla figurer som finns på rapporten har författarna själva tagit. Om inget annat angetts.

Terminologi

Användning av ord som behöver förklaring ska nämnas här. Förklaringarna kommer från ComputerSweden https://it-ord.idg.se/ord/

- **Prototyp** - Test Modell som används under utvecklingen av en produkt.
- **Bibliotek** - Ett bibliotek är en samling information som underlättar programutvecklingen. detta kan vara färdigskriven programkod, dokumentation eller egenutvecklad programkod.
- **Ramverk** - Kombination av färdigskriven kod och regler för struktur för ett program.
- **Server** - En dator som utför arbetsuppgifter i ett nätverk.(styrs inte av mänskliga användare. en server sköter sig själv).
- **Klient** - Den enhet som ansluter sig till en server för att hämta data.
- **API** - Application Programming Interface, specifikation för hur program fungerar ihop.
- **Input** - Indata, mata in, data som matar in i ett system.
- **Output** - Utdata, mata ut, resultatet på körning av ett program.
- **Port** - Anslutning, öppning
- **Port-vidarebefordran** - Applikation för översättning av nätverksadress som omdirigerar en anslutning från en adress och port nummer till en annan, även kallad *Port-Forwarding* på engelska.
- **VPS** - Virtuell Private Server, en virtuell dator.
- **HTML** - Hypertext Markup Language, ett standardspråk för hemsidor.
- **Node.JS** - Plattformsoberoende back-end språk med exekveringsmiljö i JavaScript.
- **TCP** - Transmission Control Protocol, det vanligaste protokollet för överföring av data mellan nätverk och inom internet.
- **Webbsida** – Dokument som är publicerat på webben som nås med en webbadress. En websida kan innehålla text, bild eller video.
- **TCL** – Ett vanligt skript språk som är vanligt på Unix och Linux system.
- **PMW** – Pulse Width Modulation eller Pulsbreddsmodulering på svenska.
- **Enkortsdator** – Ett litet kretskort bestående av processor, minne och input/output moduler med samma kapacitet som en dator.
- **WIFI** – Det vanligaste systemet för trådlös datakommunikation.
- **IP adress** - Internet Protocol Address, nummer som identifierar en dator som är ansluten till internet.

1. Inledning

Syftet med detta kapitel är att ge läsaren en inblick i vad frågeställningen, syftet och målet med projektet är genom fokus på röststyrning av motorfordonet.

1.1. Bakgrund

Idag är det vanligt med röststyrda och beröringsfria funktioner inom inbyggda system. Man har gett människor möjligheten kunna välja och styra funktioner såsom radio, musik eller navigering i exempelvis bilar genom röstkommandon. Många av de system som finns idag är till för att göra det lättare för användaren att göra val som inte kräver något knapptryck. Det finns moduler som är anpassade för att sköta specifika kommandon och uppdrag med bara en specifik inmatning. Det man får är ett system som inte kräver att användaren behöver flytta ögonen från aktiviteten de gör just nu utan att användaren behöver bara säga vad han/hon vill att systemet ska göra så tar systemet hand om det.

En nackdel med dagens system är att de inte har tillräcklig många funktioner. Som till exempel de flesta fordon som befinner sig på en väg behöver fortfarande ha manuell styrning för att sätta på ljuset samt de flesta fordon med radio kräver fortfarande även där manuell hantering för att starta radion. Detta kräver då att förarna släpper blicken från vägen samt att de måste köra fordonet med en hand för att göra dessa uppgifter. Detta kan då leda till att föraren inte hinner reagera på olyckor under sekunderna som föraren inte har blicken på vägen.

Ett sätt att lösa detta vore genom att ta fram en kommunikationslösning mellan en enkortsdator och ett motordrivet fordon och på något sätt implementera röststyrning på det specifika fordonet och anpassa systemet så att detta blir möjligt.

Denna kommunikationslösning kan ge funktionshindrade möjligheten att använda röststyrning för att ta sig fram med rullstol eller köra ett annat fordon som till exempel en bil. Detta sker genom anpassade kommandon som matas in genom programmet och skickas genom nätet till det specifika fordonet.

1.1. Mål

Projektets mål är att utveckla programvara för att kunna styra ett motordrivet fordon med en Raspberry Pi genom att den tar emot kommandon från en användare med en server som mittpunkt. Med hjälp av den konstruerade programvaran ska en användare kunna skicka kommandon genom röst- eller textinmatning som väljs i programmet under körning till ett fordon för att sedan få fordonet att utföra olika uppgifter.

1.2. Syfte

Syftet med detta projekt är att bevisa att det är möjligt att implementera en anpassad röststyrning med hjälp av en kommunikationslösning mellan ett motordrivet fordon, enkortsdatorn Raspberry Pi och nätet.

1.3. Frågeställningar

- Kan man utveckla ett röststyrt system för kontrollering av motorer med hjälp av en enkortsdator?

- Undersöka om detta kan användas som ett komplement till olika delar i dagens teknologi?

1.4. Avgränsningar

Detta projekt omfattar inte följande delmoment:

- Riktade och specifika säkerhetsproblem för lösningen. Däremot kommer allmänt säkerhetsproblem att nämnas.
- Artificiell intelligens såsom automatisk identifiering av individuell röst och memorering av kommandon.

2. Teknisk bakgrund

I detta kapitel beskrivs de tekniska specifikationerna på programbibliotek och elektriska komponenter som använts under projektets gång samt deras konfiguration.

2.1. Python

Python är ett programmeringsspråk som skapades under slutet av 80-talet av programmeraren Guido van Rossum i Centrum Wiskunde & Informatica som är ett forskningscentrum i Nederländerna (CWI). [1]

Första gången Guido implementerade sitt språk var 1989 på CWI. Tanken med Python var att den m.h.a. sin förmåga att generera avbrott och sin förmåga att kunna kommunicera med operativsystemet Amoeba[2] skulle vara en efterträdare till ABC[3] (Ett gammalt programmeringsspråk som var ett verktyg avsett för undervisning eller prototyping). [1]

År 1991 publicerade Guido Van Rossum för första gången sin kod till källor utanför forskningscentrumet CWI. [1]

Den senaste versionen av Python är 3.9 som lanserades 2020.[4]

2.2. Röststyrning

Röststyrning är en relativt ny teknologi där en eller flera användare kan styra en anordning med sin röst. Detta används ofta i telefoner, högtalare, hemmet eller i ett fordon. [5]

Denna teknologi fungerar genom att det finns en modul i systemet där användaren först ger ett startkommando till systemet för att göra den redo för att ta emot kommandon sedan ger användaren kommandot som denne vill att systemet ska köra och när systemet har översatt kommandot så utförts uppgiften. [5]

2.3. Server-klient modellen

En applikationsstruktur som används för att dela upp uppgifter eller arbetsbelastning mellan olika resurser, även kallat server. En klient är en tjänsteförbrukare som ansluter sig till servern för att hämta eller skicka data. Detta görs genom att klienten skickar en begäran till servern om att utföra ett arbete och returnera att arbetet är klart eller returnera resultatet på arbete. [6]

2.4. Komponenter

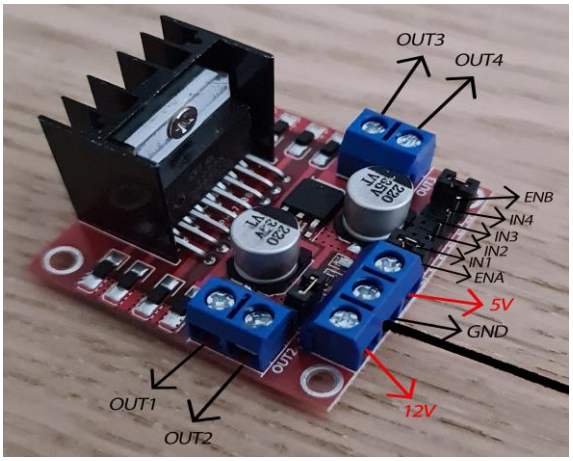
Nedan beskrivs de olika komponenter som används i detta projekt presenteras nedan.

2.4.1. Raspberry Pi

Enkortsdatorn Raspberry Pi är en liten, smart och billig dator med flertal funktioner som möjliggör programmering av applikationer och projekt för olika ändamål. Datorn kan exempelvis användas för att skapa spel, bygga en robot eller kontrollera fysiska apparater. Operativsystemet som körs är Raspbian som är ett Linux-orienterat system från Debian som utvecklades 2012. [7]

2.4.2. L298N-modul

L298N är en modul som ger än möjligheten att styra 4 likströms- och stegmotorer eller 2 motorer med möjligheten att ändra hastigheten på motorerna samt möjligheten att ändra riktning på fordonet. [8]



Figur 1: *Bilden visar modulstift layouten för L298N.*

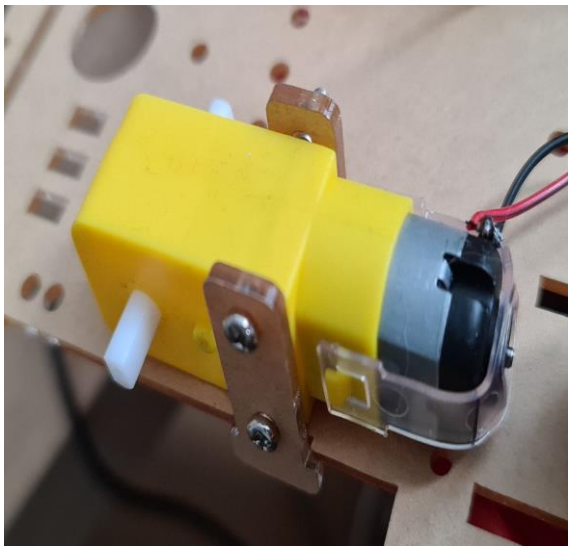
I detta projekt kombineras fyra motorer till två par motorer genom att man kopplar två par motorer till samma ut portar. Ett par kopplas från OUT2 till OUT1 och det andra paret kopplas från OUT4 till OUT3 (Se figur 1 för beskrivning på var dessa portar befinner sig). Detta görs för att ge oss möjligheten att ändra hastighet och ändra riktning genom att behandla motorerna som två par motorer istället för fyra separata motorer.

Tabell 1: L298N Modulstiftkonfiguration:

Stift namn	**Beskrivning**
IN1 & IN2	Input för första motorn. Används för att styra rotationsriktningen på motor första motorn.
IN3 & IN4	Input för andra motorn. Används för att styra rotationsriktningen på motor första motorn.
ENA	Aktiverar PWM signaler för Motor 1
ENB	Aktiverar PWM signaler för Motor 2
OUT1 & OUT2	Output för Motor 1
OUT3 & OUT4	Output för Motor 2
12V	12V input för likström
5V	Ström för logikkretsar
GND	Jord stift

2.4.3. DC3V-6 V likströmsmotor

DC3v-6 är en likströmsmotor med driftspänning på mellan 3 V ~ 6 V och lastström på 150mA (Max 200mA). Används för att omvandla likriktad elektrisk ström till rörelseenergi och därmed få fordonet att röra på sig. [9]



Figur 2: *En likströmsmotor.*

2.5. Bibliotek

Nedan beskrivs de olika biblioteken som används under projektet.

2.5.1. SocketIO

SocketIO är ett kraftfullt och användarvänligt realtids applicerat ramverk som används som verktyg för att tillföra smidiga och säkra dubbelriktade nätverksanslutningar. Det består av två delar, nämligen ett klientbibliotek som användare kan använda och som körs i webbläsaren samt ett serverbibliotek som körs på datorn med hjälp av Node.JS. Det funkar på alla plattformar och används exempelvis för chattfunktioner och streamingtjänster. [10]

2.5.2. Flask

Flask är ett webbramverk för Python som består av funktioner och tillbehör som är besläktade med webbspråk för att göra det enklare när man bygger en webbsida exempelvis en blogg eller kalenderwebbsida. Det används också för att genomföra kontrolltester för att bekräfta att webbsidan fungerar som det ska. Flask är klassificerad som ett mikroramverk vilket innebär att biblioteket inte är beroende av andra bibliotek vilket gör det lätt att använda men mer komplicerat då man måste implementera egna bibliotek. [11]

2.5.3. Selenium

Selenium är ett portabelt webbramverk som finns tillgängligt hos många programmeringsspråk och som används för att testa funktionella uppgifter i webbsidor genom ett automatiserat system som fungerar hos flera webbläsare och plattformar. Det ger också möjligheten att spela in och återkoppla felsökningar så att man smidigt kan hitta lösningar till fel i programkod. [12]

2.5.4. PyAudio

PyAudio är ett audiobiliotek som används för att lyssna och spela in ljud från en mikrofon. Med detta bibliotek kan man styra och sätta parametrar för input och output av källor på en dator. Detta bibliotek är plattformsoberoende vilket innebär att den fungerar på alla operativsystem som kan köra Python. [13]

2.5.5. Speech_recognition

Speech recognition är ett bibliotek som används för att översätta ljud. Detta bibliotek använder PyAudio (Se kap 2.5.4) för att spela in ett röstmeddelande och gör det möjligt att filtrera bort ljud som till exempel brus och sedan översätts detta med hjälp av Google Cloud Speech API. Detta API är en översättningstjänst från Google som används för att konvertera ljud från en källa såsom en mikrofon till data som sedan kan integreras för olika ändamål. Detta bibliotek stödjer även andra taligenkänningsmotorer vilket gör det smidigt för både fristående och nätbaserade applikationer. [14]

2.5.6. Tkinter

Tkinter är ett lättanvänt och inbyggt standardbibliotek i Python som används för att skapa gränssnitt. Biblioteket är plattformsoberoende vilket innebär att det fungerar på alla plattformar som stödjer Python. Alla applikationer som använder sig av Tkinter anpassas automatiskt till systemets utseende. Programmet skrivs i Python men översätts sedan till ett annat programmeringsspråk som kallas för TCL (*tickle*) som sedan ritar upp det grafiska gränssnittet. [15]

2.5.7. Övriga bibliotek

Nedan följer en rad olika systembibliotek som använts som komplement för arbetet.

2.5.7.1. Python Threading

För att kunna köra exekvering av kod med flertrådsteknik används biblioteket *Threading*. Detta bibliotek gör det möjligt att hantera körning av kod på separata trådar i ett program för att förbättra prestanda och kontrollera kodprocess. Används exempelvis när man behöver utföra flera uppgifter samtidigt eller i sekvens efter varandra då det är bättre än att köra kod i en och samma process. [16]

2.5.7.2. Python System OS

Ett systembibliotek som används för att kalla på funktioner inom ett operativsystem. Biblioteket omfattas av Python och dess standardverktygsmoduler och som ger operativsystemsberoende och avancerade funktioner för interaktion med systemet. Man kan exempelvis med hjälp av detta bibliotek hämta information om ett datasystem eller köra program. [17]

2.5.7.3. Python Time Module

Klassen används för att implementera tidskritiska funktioner i program. Den ger möjlighet att pausa, hämta körningstid på processen eller lokaltid i olika format hos en applikation. De vanligaste moduler som används är *time.sleep()* för att sätta en applikation i viloläge med argument för antal sekunder, *time.clock()* för att mäta eller jämföra processtid och *time.localtime()* för att hämta datum i olika format. [18]

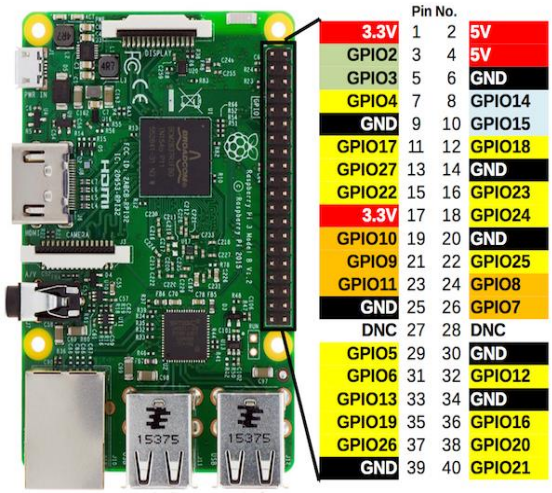
2.5.7.4. Python Sys Module

Sys är ett syskonbibliotek till Python som används för att hämta och köra kod som interagerar med exekveringsmiljön. I biblioteket så finns ett flertal funktioner för att granska Python inställningar som används i exekveringsmiljön. Kan också användas för att undersöka moduler och funktioner som Python använder sig av och som sedan kan utnyttjas och ändras för olika syften i en applikation. [19]

2.5.7.5. RPi.GPIO

RPI.GPIO är ett bibliotek som används för att kontrollera ingångs- eller utgångsdata från enkortsdatorn Raspberry Pi. Detta görs med PWM-signaler utlagda på olika stift på enkortsdatorn enligt Figur 3. [20]
Stiften som används under detta projekt är 25,24,26,21,16 och 20.

PWM-signaler även kallad för pulsbreddsmodulering är en metod för att åstadkomma analoga signaler med digitala verktyg. Detta går ut på att man ökar spänningen eller minskar spänningen. [21]



Figur 3: *Raspberry Pi:s olika programmerbara och strömsatta stift.*

Man kan genom programmering och färdig programvara styra, strömförsörja eller läsa av stiften och koppla ihop elektroniska komponenter för att skapa olika intressanta projekt. [22]

3. Metod

Under projektets gång har en blandning av vattenfallsmetoden och Scrum använts. Vattenfallsmetoden utgår ifrån att man jobbar med ett projekt genom ett så kallat flöde där man tar varje moment för sig. Detta flöde består av kravställning, analys, design, kodning, testning och slutligen drift.

En av nackdelarna med denna metod är att det kan vara svårt att återkoppla och göra förändringar i en tidigare fas av projektet som man redan har passerat. Testning görs exempelvis i slutfasen vilket egentligen borde göras varje gång man gjort nya ändringar för att säkerställa att applikationen fungerar som det ska istället för att förvänta sig ett fungerande resultat vid driftsfasen. Fördelen med detta är att projektet följer en ordning av faser som är lätt at hålla råda på. Om man har jobbat med projektet i faser från början har man fått en teoretisk bild på hur allt fungerar. [23]

Scrum är en agil metodik för att jobba med systemutveckling. Tanken bakom scrum är att man har en stor idé som man sedan delar upp i olika arbetsuppgifter baserat på hur komplicerat problemet är. När detta har gjorts delar man upp arbetsuppgifterna till olika uppdrag som man sedan bearbetar under en viss period vilket bestäms i början av projektet. Denna metodik ger möjligheten att förändra och jobba med ett projekt på lättast sätt, genom dokumentation, återkoppling och dagliga möten. [24]

Några fördelar med detta arbetssätt är att genom att man har frekventa möten kan problem hanteras och åtgärdas direkt istället för att man fastnar på problemet i flera timmar. Man delar upp ansvaret bland alla personer i laget, för att få mer arbete gjort på kortare tid vilket gör alla ansvariga för projektet och kan leda till högre motivation. Nackdelen med detta arbetssätt är att eftersom man arbetar snabbt och har snäv tidsplan så kan detta leda till att projektet blir försenat om något oförutsett inträffar. Det kan då bli svårt att planera in dagliga möten. [25]

4. Genomförande

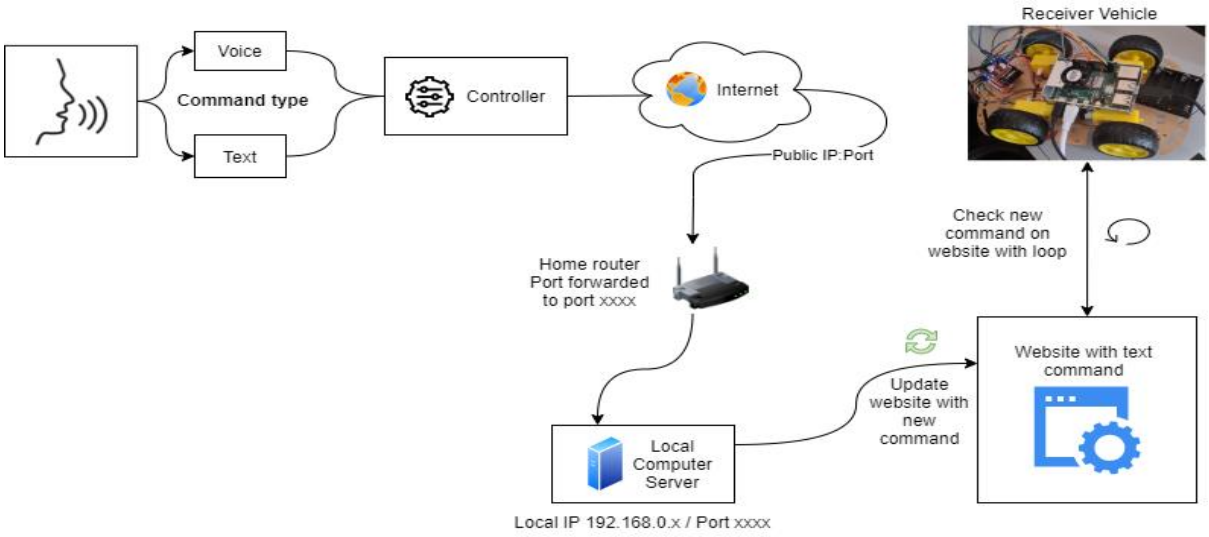
För att uppnå målet med kommunikation mellan motorfordon och enkortsdatorn måste man ta hänsyn till de förutsättningar och krav som gäller för klient-server modellen. Genom ett antal försök av implementation och med hänsyn till pålitlighet, anslutning och tid så har en sammanställning av strukturen gjorts. Slutligen redovisas alla implementations delar.

4.1. Systemstruktur

Tanken med röststyrning av motorfordon är att det ska vara åtkomligt överallt och fungera på alla fordon som har grundfunktioner för att röra sig eller utföra tjänster. Detta innebär att det måste finnas en nätverkspunkt för att systemet ska kunna komma åt Internet då kommunikationen sker trådlöst. Systemet har delats upp till 3 delar; *Server*, *Controller* och *Receiver*. Servern är riktad mot att upprätthålla en anslutning mellan både kontrollenheten och mottagaren medan de övriga två modulerna innehåller funktioner för att skicka och ta emot data såsom röst och textkommandon.

Alla skript är skriva i samma språk, nämligen Python. Servern innehåller även delar av HTML kod som används för att ge ett gränssnitt på en webbsida som man kan läsa av för att verifiera att rätt textkommando har skapats. För kommunikation valdes SocketIO som är ett bibliotek som använder Node.JS och finns i varianter för flera programmeringsspråk. Det är nära besläktat med dataöverföring protokollet TCP och kommunikationsprotokollet WebSocket fast med ett flertal extra funktioner såsom event-orienterade och dubbelriktade anslutningar i realtid som blir viktiga för system strukturen och tidskravet.

När en användare spelar in ett röstmeddelande med kontrollklassen (*Controller*) skickas detta meddelande vidare via nätet till servern. På servern körs serverklassen (*Server*) och dennes uppgift är att vidarebefordra inkommande data från kontrollklassen till webbsidan. Slutligen så läser mottagarklassen (*Receiver*) av data från webbsidan och anropar motorklassen för att utföra begärt kommando på fordonet. Detta illustreras på figur 4.



Figur 4: *Diagram på systemet*

4.2. Planering

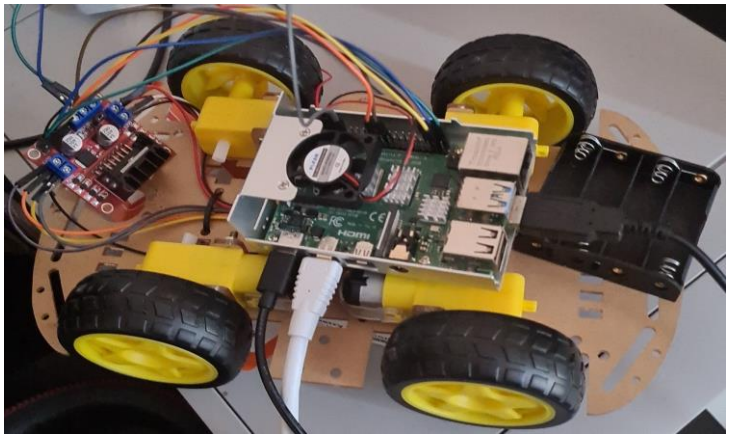
Första steget var att planera hur motorfordonet skulle kunna styras. En enkel representation av motorfordon är en minirobotbil som har plats för enkortsdatorn Raspberry Pi och som kan göra rörelser. Genom programmering i Python skulle det vara möjligt att styra anslutningsben som finns på enkortsdator och därmed kunna styra olika komponenter på motorfordonet. Man ville också kunna skriva kod som skulle vara körbart på alla möjliga operativsystem och använda sig av bibliotek som hade många funktioner för simpel nätverkshantering.

4.3. Montering

Fordonet byggdes ihop med fyra motorer (en motor per hjul) som monteras fast på en akrylskiva tillsammans med ett batteripack. Dessa motorer kopplas sedan in i L298N modulen som kontrollerar spänningen till motorerna (se Kap. 2.4.2 för mer information) och L298N modulen kopplas sedan ihop med Raspberry Pi (Se Kap 5.1 för information om vilka I/O som använts). Enkortsdatorn monteras sedan på en annan akrylskiva och dessa två akrylskivor skruvas samman.

4.3.1. Fordon nummer 1

Den första byggda minibilen består av ett fåtal grundläggande komponenter för att göra bilen till ett motorfordon. I figur 5 syns motorfordonet som består av enkortsdatorn Raspberry Pi, en L298N modul uppkopplat med konfigurationen som nämnts tidigare i kapitel 5.1 samt fyra motorer och hjul.(se figur 5)



Figur 5: *Motorfordon nummer 1.*

4.3.2. Fordon nummer 2

Lösningen har även testats på ett annat fordon från Amazon och detta visade sig fungera även för detta fordon. För att se exakta specifikationer på fordonet se [27].

I figur 6 syns fordonet och dennes komponenter.

Figur 6: *Motorfordon nummer 1. Bildkälla [26]*

4.4. Testning

Tester utfördes både lokalt och mellan olika enheter och nätverk när koden och motorfordonet var färdigt. Man ville se hur allt funkade i praktiken och om de uppfyller de nödvändiga kraven för att anses vara ett godtyckligt system för alternativa motorfordon. Det som testades var följande:

- **Responstid**: Mätning av tid mellan alla enheter. Man uppmätte tiden det tog för kontrollmodulen att skicka röst och textkommandon till servern och även hur lång tid det tog för mottagarmodellen att ta emot och utföra kommandot hos motorfordonet.
- **Mikrofon:** Test på vilken mikrofon som gav bäst översättning för att se om det spelar roll vilken mikrofon man använder som ger tydligast utdata.

5. Systemkonstruktion

I detta kapitel beskrivs de olika klasser och metoder som har skapats och använts under arbetet med bilder på källkod och förklaringar.

Systemet som har konstruerats är baserat på klient-server modellen där vi har två klienter som skickar data till varandra genom en huvudserver. Dessa klienter är mottagarmodulen (Se kap 5.2) och kontrollmodulen (Se kap 5.5). Servermodulen (Se kap 5.3) tar emot dessa kommandon och skickar till rätt mottagare.

5.1. Motor

Motor klassen används för att styra fyra motorer genom att man först initierar stiften för I/O på L298N (se avsnitt 2.2.2 för detaljerad layout för I/O) samt hur snabbt fordonet ska vara genom att man sätter hur mycket ström som motorerna ska ha tillgång till. (I detta fall används I/O 25, 24, 26, 21, 16, 20 för att kontrollera fordonet (Se figur 7). Genom att aktivera och avaktivera specifika stift förs fordonet i specificerad riktning. Till exempel om du vill svänga höger aktiverar du I/O 24 och avaktiverar I/O 26, 16 och 20.(se figur 8) Om du vill få fordonet att köra rakt fram aktiverar du I/O 24 och 16 och avaktiverar 26 och 20 (Se figur 9 samt se bilaga I för flera funktioner)

```
class Motor():
    def __init__(self,Ena,In1,In2,Enb,In3,In4):
        self.Ena = Ena
        self.In1 = In1
        self.In2 = In2
        self.Enb = Enb
        self.In3 = In3
        self.In4 = In4
        GPIO.setup(self.In1,GPIO.OUT)
        GPIO.setup(self.In2,GPIO.OUT)
        GPIO.setup(self.Ena,GPIO.OUT)
        GPIO.setup(self.In3,GPIO.OUT)
        GPIO.setup(self.In4,GPIO.OUT)
        GPIO.setup(self.Enb,GPIO.OUT)
        self.p = GPIO.PWM(Ena,100)
        self.p2 = GPIO.PWM(Enb,100)
        self.p.start(60)
        self.p2.start(60)
```

Figur 7: *Kodstruktur på motorklassen*

```
def moveRight(self):
    GPIO.output(self.In1,GPIO.HIGH)
    GPIO.output(self.In2,GPIO.LOW)
    GPIO.output(self.In3,GPIO.LOW)
    GPIO.output(self.In4,GPIO.LOW)
    sleep(2)
```

Figur 8: *Kodstruktur på moveRight funktion i motorklassen*

```
def moveForward(self):
    GPIO.output(self.In1,GPIO.HIGH)
    GPIO.output(self.In2,GPIO.LOW)
    GPIO.output(self.In3,GPIO.HIGH)
    GPIO.output(self.In4,GPIO.LOW)
    sleep(2)
```

Figur 9*: Kodstruktur på moveForward funktion i motorklassen*

5.2. Mottagare

Mottagarklassen består av funktioner som ges av Selenium API för att kunna lyssna på data som ges från webbsidan som servern håller igång (Se figur 10 för kodstruktur på mottagarklassen). Med hjälp av Selenium och Webdriver så kan man avläsa ändringar på en webbsida. När man kör programmet öppnas en ny flik i webbläsaren Google Chrome (Denna webbläsare kan ändras genom att man ändrar ”PATH” till önskad webbläsare) som är satt i *Debug mode* vilket innebär att skriptet har full kontroll över vad som sker på webbsidan. Skriptet ansluter till servern och dess HTML sida för fordonet och därefter avläser programmet var femte sekund eventuella förändringar av kommandot på webbsidan. Sker en ändring så tar den det kommando som givits och skickar sedan detta vidare till motorklassen. (Se kap 5.2 för mottagarklassen samt bilaga I för specifika metoder för motorklassen).

```
PATH = '/usr/lib/chromium-browser/chromedriver'
options = webdriver.ChromeOptions()
options.add_experimental_option("excludeSwitches", ["enable-logging"])
driver = webdriver.Chrome()
driver.get('http://83.252.170.109:1605/car1')

motor = Motor(25,24,26,21,16,20)
# Checks the site for command and then executes it
def getText():
    threading.Timer(2, getText).start() # check the site every 2s
    element = driver.find_element_by_tag_name('p')
    command = element.text
    # check what the input correspond to
    if command=="forward":
        motor.moveForward()
    if command=="backward":
        motor.moveBackward()
    if command=="right":
        motor.moveRight()
    if command=="left":
        motor.moveLeft()
    if command=="fast":
        motor.moveFaster()
    if command=="normal":
        motor.moveNormal()
    if command=="slow":
        motor.moveSlow()
    if command=="start":
        motor.start()
    if command=="stop":
        motor.stop()
    if command=="start":
        motor.start()
    if command=="off":
        motor.turnOff()
    if command=="play radio":
        print("Playing radio")
    if command=="stop radio":
        print("Stopping radio")
    if command=="blink left":
        print("blinking left")
    if command=="blink right":
        print("blinking right")

    print(command)

    driver.refresh()
getText()

```

Figur 10: *Kodstruktur för Mottagare klassen*

5.3. Server

Denna klass består av webbramverket SocketIO-Flask som används för att starta en nätverkspunkt för klienter att ansluta till. Den upprätthåller en server på lokal host på port **1605** som man sedan kan komma åt överallt på nätet med sin publika IP adress efter att man öppnat en port väg på routern. I koden initieras en hemlig nyckel tillsammans med applikationen. Den hemliga nyckeln ser till så att alla anslutningar som görs mellan server och klient är säkra och denna nyckel känner bara servern till (Se figur 11).

```python
# Will be online to display the website for navigating to right car.
from flask import Flask, render_template
from flask_socketio import SocketIO, send

app = Flask(__name__)
app.config['SECRET_KEY'] = '@%["{w>[;l`OFDu=`?^Tb9QuIEDvm2'
socketio = SocketIO(app)

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/car1')
def car1():
    return render_template('car1.html')

@app.route('/car2')
def car2():
    return render_template('car2.html')

@socketio.on('message')
def handleMessage(msg):
    send(msg, broadcast=True)

if __name__ == '__main__':
    print('Server running...')
    socketio.run(app, host='192.168.0.15', port=1605)
```

Figur 11: *Kodavsnitt på server*

5.4. Kommunikation mellan server och mottagare

När man startar servermodulen på en dator så upprätthålls en nätverkspunkt vilket ger möjlighet för fordon att ansluta sig till webbsidan som innehåller HTML sidor för styrning av fordon [Bilaga II]. Webbsidan kan nås från vilket dator som helst, och flera fordon kan därmed ansluta sig till webbsidan för att sedan invänta kommande kommando. Fordon väljer man på inställningarna hos kontrollmodulen och genom att navigera till den sida på webbsidan som motsvarar fordonet hos mottagarmodulen. Mottagarmodulen som körs på separat enkortsdator hos fordonet känner av när det sker en ändring på webbsidan, och när ett kommando dyker upp så tas detta in och läses av för att sedan jämföras och verkställas.

5.5. Kontrollmodul

Kontrollmodulen består av varierande delar kod som nästan alla utgör en viktig funktion enligt figur 12 till figur 14. Biblioteken som används är importerade på toppen av skriptet. Följande funktioner förklaras mer detaljerat i sektioner. Kontrollmodulens uppgift är att skicka kommandon till servern (Se kap 5.3) sedan skickas dessa kommandon från servern till fordonet som användaren kan kontrollera.

5.5.1. Initiering av SocketIO

Programmet inleds med genom att en SocketIO klient initieras med några standardvärden som används för att identifiera fordonet som ska ta emot kommandon. Som standard så sätts första bilen i listan tills dess att man ändrar bil på konsolgränssnittet eller i koden. Efter att händelsehanteraren har lagts till så ansluter klienten sig till servern genom publik IP och port som syns på kodavsnittet vid rad 40 och ger programmet en liten kort tid att för hinna ansluta ordentligt.

5.5.1.1. Eventhandler

I samband med initiering så sätts även olika händelser som klienten använder sig att för att identifiera beteende som sker i programmet och även för att skicka kommandot genom *my_message()*. När en klient har anslutit till servern så körs *connect()* funktionen och när man avbryter anslutningen så körs *disconnect()*. Varje funktion visar olika meddelanden beroende på vad som händer.

5.5.1.2. Anslutning

Ett booleskt uttryck som finns i SocketIO biblioteket använts för att känna av när anslutningen har gjorts. Så fort den har anslutit så skrivs konsolen ut och användaren kan då välja biltyp, skicka textkommando och göra röstinspelning.

```
Controller.py > ...
1   # Will be placed in Raspberry Pi for controlling the car.
2   import socketio
3   from os import system, name
4   import time, sys
5   import speech_recognition as sr
6   # ----------------------------------------------------
7
8   # Declarations
9   sio = socketio.Client()
10  exit = False
11  cartype = 0
12  choice = ""
13  # --------------------
14
15  # Clear console window.
16  def clear():
17      if name == 'nt': # Windows System
18          _ = system('cls')
19
20      else: # Mac & Linux Systems
21          _ = system('clear')
22  # --------------------
23
24  # Client events on the server.
25  @sio.event
26  def connect():
27      print('Connection established...')
28
29  @sio.event
30  def my_message(car, message):
31      sio.emit('message', str(car) + ':' + message)
32
33  @sio.event
34  def disconnect():
35      print('Disconnected from server')
36      time.sleep(1)
37      sys.exit()
38  # -------------------------
39
40  sio.connect('http://83.252.170.109:1605')
41  time.sleep(1) # For connection purposes.
42
43  # Choose car to operate once client has connected.
44  if (sio.connected):
45      choice = input('Select your car: 1 OR 2: ')
46      choice = int(choice)
47
48      if choice == 1:
49          cartype = 1
```

Inte viktigt för denna script.

Figur 12: *Kodstruktur på kontrollmodul*

5.5.2. Konsolgränssnitt

När användaren anslutit till servern så visas ett konsolgränssnitt på skärmen. Detta är enkelt gjord för att underlätta navigering mellan olika val. Man väljer en siffra mellan 0–11 som sedan skickar ett textkommando eller utför en röstinspelning som den översätter och skickar till servern. Man kan även avsluta programmet, och även skifta mellan de fordon man vill styra.

5.5.2.1. Röstinmatning

När röstinspelning valts så initieras biblioteket *speech_recognition* och ber användaren att prata in i mikrofonen. När ljudet spelas in begränsas även bakgrundsljudets störande inverkan som biblioteket hanterar. Efter att programmet mottagit röstmeddelandet så omvandlas det för att tolkas och skickas vidare som kommando till servern och sedan vidare till det valda fordonet.

```
while exit != True:

    # Console menu
    print (30 * '-')
    print ("  Controller Menu for car " + str(cartype) + '  ')
    print (30 * '-')
    print ("0.  Send voice command")
    print ("1.  Start")
    print ("2.  Stop")
    print ("3.  Forward")
    print ("4.  Backward")
    print ("5.  Left")
    print ("6.  Right")
    print ("7.  Normal speed")
    print ("8.  Fast speed")
    print ("9.  Slow speed")
    print ("10. Switch car")
    print ("11. Exit")
    print (30 * '-')

    choice = input('Enter your choice [1-11]: ')
    choice = int(choice)
    # ------------

    # --- Commands to send to car ---
    if choice == 0:
        r = sr.Recognizer()
        print("Please talk now into microphone..")
        with sr.Microphone(device_index = 2) as source:
            while True:
                r.adjust_for_ambient_noise(source)
                audio = r.listen(source)
                try:
                    result = r.recognize_google(audio)
                    print("You said " + result)
                    words = result.lower()
                    if (cartype == 1):
                        my_message(1, words)
                    else:
                        my_message(2, words)
                    clear()
                    break
                except LookupError:
                        print("Cant connect to Internet")
                except sr.UnknownValueError:
                        print("Please, speak more clearly")
                except sr.RequestError:
                        print("Cant connect to Internet")
```


Figur 13: *Kodstruktur på kontrollmodul*

5.5.2.2. Växling av fordon

Ett av valen som finns är att kunna byta fordon. Hos servern finns det möjlighet att växla mellan fordon. Detta sker enbart om användaren väljer kommando nummer 10 i konsolgränssnittet.

5.5.2.3. Avslutning

När man är färdig så väljer man att stänga programmet. Efter man valt *exit* så kopplas SocketIO ifrån servern och skriptet avslutas och skickar ett stoppkommando till fordonet som säger åt det att stanna, ifall det skulle vara i rörelse.

```
    # Switch car and refresh menu.
    elif choice == 10:
        if (cartype == 1):
            cartype = 2
        elif (cartype == 2):
            cartype = 1

        clear()
        print("Switching to car " + str(cartype) + "..")
        time.sleep(1)
        clear()
    # ----------------------------

    # Clear console and disconnect from server.
    elif choice == 11:
        if (cartype == 1):
            my_message(1, 'stop')
        else:
            my_message(2, 'stop')
        clear()
        disconnect()
        exit = True
    # ----------------------------
```

Figur 14*:Kodstruktur på kontrollmodul*

5.6 Kommunikation mellan kontroll och server

Genom konsolgränssnittet som körs via Raspberry Pi med kontrollmodulen skickas valt kommando till servern genom port 1605. Servern skickar sedan vidare kommandot till mottagaren för att utföra arbetet. (Se kap 5.4 för exakt beskrivning för hur detta görs).

6. Resultat

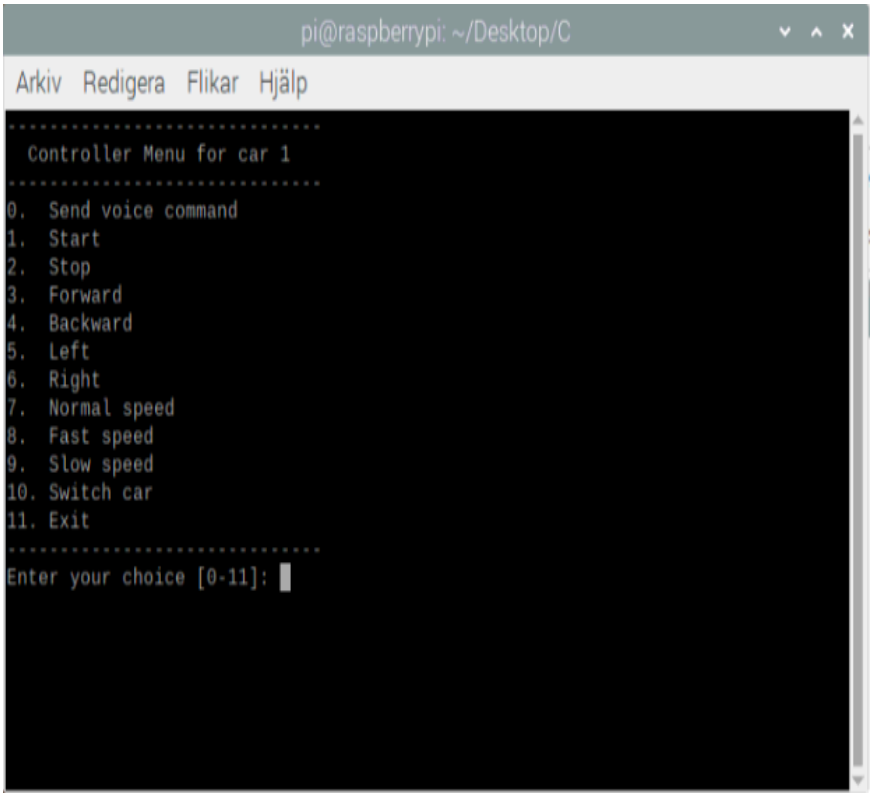
I detta kapitel redovisas de resultat som man fått fram efter användning av eget motorfordon och tilläggning av gränssnitt för kontrollering och styrning av fordonet.

6.1. Gränssnitt

Arbetet resulterade i två användarvänliga gränssnitt som redovisas nedan.

6.1.1. Kommandogränssnitt

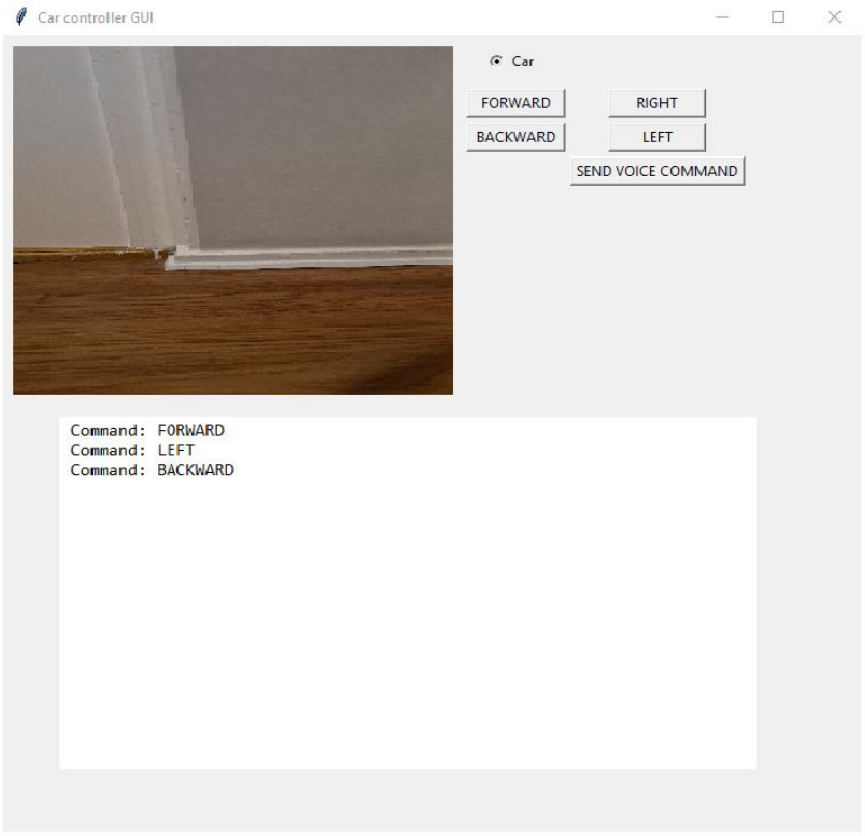
Ett av dessa användargränssnitt som detta arbete har resulterat i är en smidig applikation med användarvänligt gränssnitt som syns i kommandotolken när man kör skriptet i ett Linux system som heter Raspbian som finns hos enkortsdatorn (Se figur 15). Genom konsolen kan användaren enkelt skicka kommandon till motorfordonet såsom svängningar eller körningar, och även kunna spela in röstkommando samt avsluta och stoppa applikationen.



***Figur 15**: Kommandogränssnitt*

6.1.2. Grafiskt användargränssnitt

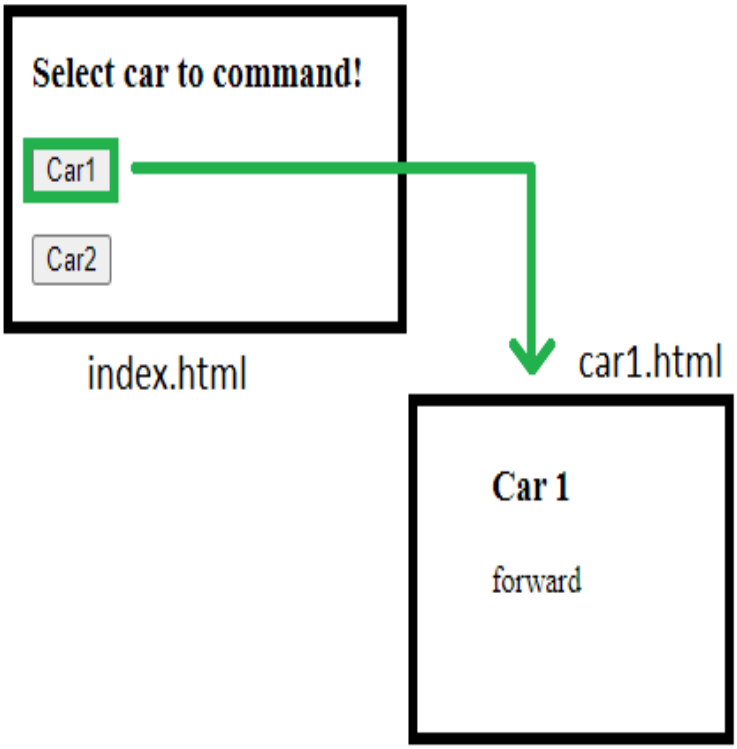
Det andra användargränssnittet som framtagits under detta arbete är en smidig applikation med ett mer estetisk tilltalande utseende än tidigare nämnda gränssnittet (Se kap 6.1.1). Med knappar som man kan klicka på för att skicka kommando till fordonet samt en inspelningsknapp för att skicka röstkommandon. Denna applikation har även ett litet fält längst ner där tidigare körda kommandon skrivs ut.
(Se figur 16)



Figur 16: *Grafiskgränssnitt*

6.2. Webbsida

Websidan skapades med hjälp av Flask biblioteket enligt kapitel 2.5.2. Den resultera i 3 sidor: *index.html*, *car1.html* och *car2.html*. Index sidan som är huvudsidan används för att se vilket kommando som körs just nu på fordonet. Detta är även webbsidan som mottagaren avlyssnar och avläser efter kommandon den ska köra. Knapparna ”Car1” och ”Car2” representerar vilka fordon du kan se kommandon på.



Figur 17: *Layouten på index.html och car1.html*

Varje sida för respektive fordon har en textruta där kommandot som togs emot visas. Man kan skicka vilken data som helst, och med hjälp av mottagarmodellen som nämnts i kapitel 5.2 kan man hämta ut data som skickas från server till html sidan. Sedan körs kommandot på motorfordonet genom Raspberry Pi.

6.3. Röststyrningen

Röststyrningen går ut på att användaren först klickar på valet för att spela in ett röstkommando. Då anropas Speech recognition (se kap 2.5.5) som sedan anropar PyAudio (se kap 2.5.4) för att göra programmet redo för att spela in ett röstmeddelande. PyAudio avlyssnar mikrofonen i tre sekunder och skickar sedan tillbaka det inspelade röstmeddelandet. När meddelandet kommer från PyAudio filtrerar Speech_recognition biblioteket bort bakgrundsljud från ljudfilen och skickar sedan den filtrerade ljudfilen vidare till Googles API där ljudfilen översätts och omvandlas till text. Textkommandot skickas sedan vidare från kontrollklassen (se kap 5.5) till servern.

Röststyrningen kan inte identifiera användaren. Detta medför till att programmet inte vet vem föraren av fordonet är och kan därmed styras av flera användare. Responstiden för röststyrningen har en kritisk tid på 8 sekunder. Detta på grund av att man inte har implementerat något sätt att verifiera kommandot på om användaren eller API ger ett giltigt kommando. I vissa fall kan information som skickats till API försvinna helt eller översättas fel som till exempel kommandot ”FORWARD” som ibland blir ”FORTH WORTH”.

Detta kan bero på att tidsintervallet (tre sekunders inspelning av röstmeddelande) är för kort och att placering av mikrofonen var för nära. Beroende på hur långt man håller mikrofonen eller vilken mikrofon man använder så fick man bättre eller sämre översättning. Vid detta tillfälle kör programmet det tidigare kommandot.

6.4. Kommunikationslösningen

Kommunikationen går ut på att alla anslutningar till och från bilen skickas genom portar som man har vidarebefordrat (I detta fall port 1605).

När användaren är redo för att skicka ett kommando ansluter klienten (Controller. se kap 5.5) till server (se kap 5.3) med hjälp av SocketIO biblioteket (se kap 2.5.1). Anslutningen börjar med att klienten försöker ansluta till servern genom port 1605. När denna anslutning är gjord skickar klienten det översatta kommandot till servern som sedan med Flask biblioteket (se kap 2.5.2) skapar HTML sidorna och Selenium (se kap 2.5.3) som gör det möjligt att visa realtids ändringar på webbsidan. Kommandot publiceras sedan på webbsidan och när detta har gjorts avläser mottagarklassen (se kap 5.2) kommandot och anropar metoden som kommandot tillhör i motorklassen (se kap 5.1) och bilen färdas i den angivna riktningen. Dessa anslutningar sker via Raspberry Pis inbyggda WiFi modul.

Denna anslutning är inte säker eftersom även andra personer kan ansluta sig till den öppna porten och skicka kommandon till fordonet och andra personer kan även ansluta sig till webbsidan med hjälp av den publika IP-adressen.

7. Diskussion

Detta kapitel beskrivs olika problem som uppkommit under projektets gång samt förslag till fortsatt arbete

7.1. Problem under körning av programmet

Under projektets gång har det uppstått flera problem som fått lösningen att vara mindre pålitlig. Dessa problem beskrivs i rubrikerna nedan.

7.1.1. Identifiering av förare

Som tidigare nämnts i kap 6.3 kan inte programmet känna igen föraren av fordonet. Programmet kör alla kommandon som kommer in genom mikrofonen som används under körning. Detta medför att om passagerarna ger ett kommando kör programmet kommandot eller om föraren pratar i telefon och programmet får en input genom mikrofonen kommer kommandot att köras. Detta är delvist löst genom att föraren för fordonet klickar på en knapp innan han/hon vill ge ett kommando till bilen men identifierings problemet kvarstår.
Detta problem går att lösa genom att man ber föraren innan han/hon kör bilen att ge ett par exempelkommandon och spelar in dessa. När användaren vill ge ett nytt kommando kan programmet använda ett annat bibliotek som till exempel Librosa eller liknande som har möjligheten att analysera ljudfilerna. Med ett sådant bibliotek kan programmet analysera om det är föraren som ger fordonet kommandot eller om det är en passagerare. Om föraren av misstag klickat på knappen bör det finnas en annan knapp som hindrar sista kommandot från ska köras.

7.1.2. Nätverksproblem

Som tidigare nämnts använder vi ett publikt nätverk. Detta medför att alla personer som har tillgång till IP-adressen kan skicka kommandon till webbsidan direkt. Bilen vet inte varifrån kommandot kommer utan bara att den ska göra så som det står på webbsidan.

Ett alternativ för att åtgärda denna säkerhetsbrist vore att installera en Virtuell Private Server (VPS) som man kan rikta sina användare till då den har högre flexibilitet, bättre säkerhet med trafikdata, är billig i drift och är helt åtskild från övriga nätverk. Man kan sedan använda sig av säkerhetsverktyg för att lägga till ytterligare skydd på servern som exempelvis kryptering av nätverksanslutningar. [28]

7.1.3. Identifiering av input data

Som nämnts tidigare i kap 6.3 kan data försvinna eller översättning bli fel. Detta kan man lösa genom att istället för att man låta PyAudio lyssna i tre sekunder, ändra programmet till att den slutar lyssna när föraren har släppt knappen. Man kan även introducera AI som har som uppgift att översätta felaktiga översättningar till det som kommandot är mest likt som till exempel ”FORT WORTH” till ”FORWARD”. Dessutom borde även en annan översättningstjänst användas istället för Google API då Speech_recognition har stöd för flera övriga översättningstjänster. (Se referens [14] för övriga översättningstjänster som Speech_recognition stödjer). Man borde även skapa en standardmetod som kör om programmet inte uppfattar kommandot. Som till exempel om programmet inte förstår ett kommando kanske fordonet borde stanna vid sidan av vägen.

7.1.4. Tidsfördröjning

Tidsfördröjningen som programmet har är ganska hög och ligger inte för tidsramen som en normal frisk person har vilket motsvarar tre sekunder från att personen har upptäckt faran till att fordonet står helt stilla. [29].

Flaskhalsen verkar vara identifiering av vad användaren säger samt att biblioteket verkar göra beräkningar i bakgrunden på ljudfilen innan den skickas vidare för översättning. På grund av dessa beräkningar verkar Googles API reagera genom att översättningen tar längre än förväntat.

Som nämnts i föregående kapitel kan man introducera AI för att identifiera felaktiga ord och justera dessa till korrekt kommando. Man kan även skapa ett nytt bibliotek från grunden som använder Googles API men inte gör onödiga beräkningar för att minska tidsfördröjningen.

7.2. Potentiella problem utanför programmet

7.2.1. Säkerhetsrisk

Eftersom programmet inte använder egna servrar och använder våra egna datorer som en server krävs det att rätt portar på routern är öppna. Detta medföljer en mängd risker för serverdatorn eftersom när man öppnar portar ber man routern att inte titta på input data som kommer genom nätverket till datorn. Detta kan användas av andra personer som har skadliga avsikter för att få tillgång till serverdatorns filer.

Detta går att lösa genom att man använder sidor som hyr ut olika servrar eller genom att man skapar en brandvägg så att ingen obehörig kommer åt systemfiler.

7.3. Gränssnitt

Nedan beskrivs det för- och nackdelar med programmets två olika gränssnitt.

7.3.1. Kommandotolken

Se kap 6.1.1 för visuell bild.

7.3.1.1 Fördelar

Fördelar med detta gränssnitt är att det inte är krävande för datorn att rita upp gränssnittet på kommandotolken. Man kan även rita upp fler användarval utan att det tar mycket plats på skärmen.

7.3.1.2 Nackdelar

Nackdelar med detta gränssnitt är att det inte är estetiskt tilltalande och saknar bildflöde.

7.3.2. Grafik

Se kap 6.1.2 för visuell bild.

7.3.2.1 Fördelar

Fördelar med detta gränssnitt är att det är mer estetisk tilltalande. Möjligheten till bildflöde finns och mer valmöjligheter på hur programmet ska se ut.

7.3.2.1 Nackdelar

Eftersom Tkinter och dess olika anrop omformas till TCL(”tickle”) kommandon och översätts av TCL finns det en risk för att hastigheten på programmet försämras detta är dock inte ett stort problem i verkligheten men problemet kvarstår. [30]

7.4. Miljö och etiska frågor

Nedan diskuteras de miljökritiska, hållbara och etiska frågorna för projektet med förslag på lösningar för framtiden.

7.4.1. Miljö

Detta projekt bidrar inte till en ökad negativ effekt på miljön som dagens fordonsindustri redan har.
Det enda projektet kan bidra med är ökade olyckor.

7.4.2 Etiska frågor

Under projektets gång har det kommit upp flera etiska frågor. Dessa frågor är klassiska frågor som dyker upp när man börjar prata om mer självkörande fordon. Självkörande fordon är fordon som inte kräver konstant input från föraren. [31]

8. Slutsats och förslag till uppdateringar

Slutprodukten är ett koncept på hur man skulle kunna göra en kommunikationslösning till ett motorfordon. Trots flera behövda uppdateringar kan slutprodukten användas till en kommunikationslösning.

För vidareutveckling av programmet bör man göra en analys på om man ska fortsätta utvecklingen med samma avgränsningar eller om man ska lägga till eller ta bort en avgränsning. Ett förslag är att man fokuserar på en smalare variant av fordon och att man försöker lösa säkerhetsriskerna som finns i projektet innan nya funktioner implementeras eller skapas.

Om man förbättrar responstiden samt anslutningen till servern och Raspberry Pi kan detta användas i olika industriella miljöer.

9. Referenser

[1] Javapoint, “Python History and Versions” [Online]. Tillgänglig: https://www.javatpoint.com/python-history. [Hämtad 2021-05-01]

[2] Andrew S. Tanenbaum, “The amoeba distributed operating System” [Online]. Tillgänglig: https://cds.cern.ch/record/400320/files/p109.pdf [Hämtad 2021-05-21]

[3] Pemberton. Steven ” ABC: The Language” [Online]. Tillgänglig: https://homepages.cwi.nl/~steven/abc/language.html [Hämtad 2021-05-21]

[4] Python ”Active python releases” [Online]. Tillgänglig: https://www.python.org/downloads/ [Hämtad 2021-05-21]

[5] Raspberrypi, “*What is a Raspberry Pi?*”. [Online] Tillgänglig: https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/ [Hämtad 2021-05-21]

[6] Syedmodassirali, ”Client-Server Model”. [Online] Tillgänglig: https://www.geeksforgeeks.org/client-server-model/ [Hämtad 2021-05-28]

[7] Engstedt. H ”Prata med prylarna – så funkar röststyrning”. [Online] Tillgänglig: https://www.radron.se/vardagskunskap/sa-funkar-roststyrning/ [Hämtad 2021-05-21]

[8] Components101, “L298N Motor Driver Module”. Apr. 2020. [Online] Tillgänglig:https://components101.com/asset/sites/default/files/component_datasheet/L298N-Motor-Driver-Datasheet.pdf . [Hämtad 2021-05-01]

[9] adafruit Industries, “DC Gearbox Motor – TT Motor 200RPM – 3 to 6VDC”. [Online] Tillgänglig: https://media.digikey.com/pdf/Data%20Sheets/Adafruit%20PDFs/3777_Web.pdf [Hämtad 2021-05-28]

[10] Lewington, Glyn. “Everything You Need to Know About Socket.IO” [Online] Tillgänglig: https://ably.com/topic/socketio. [Hämtad 2021-05-01]

[11] Pythonbasics. “What is Flask Python” [Online] Tillgänglig: https://pythonbasics.org/what-is-flask-python/. [Hämtad 2021-05-01]

[12] Guru99. “What is Selenium? Introduction to Selenium Automation Testing” [Online] Tillgänglig: https://www.guru99.com/introduction-to-selenium.html [Hämtad 2021-05-01]

[13] Joska de Langen, ”Playing and Recording Sound in Python” [Online] Tillgänglig: https://realpython.com/playing-and-recording-sound-python/ [Hämtad 2021-05-28]

[14] Amos, David, “The Ultimate Guide To Speech Recognition With Python” [Online] Tillgänglig: https://realpython.com/python-speech-recognition/ [Hämtad 2021-05-01]

[15] Amos, David, “Python Programming with Tkinter” [Online]

Tillgänglig: https://realpython.com/python-gui-tkinter/ [Hämtad 2021-05-03]

[16] Anderson, Jim, ”An Intro to Threading in Python” [Online] Tillgänglig: https://realpython.com/intro-to-python-threading/ [Hämtad 2021-05-03]

[17] Rajnis09. ”Python | os.system() method” [Online] Tillgänglig: https://www.geeksforgeeks.org/python-os-system-method/ [Hämtad 2021-05-03]

[18] Gallagher, James, ”How to Use the Python Time Module” [Online] Tillgänglig: https://www.careerkarma.com/blog/python-time/ [Hämtad 2021-05-03]

[19] GeeksforGeeks, ”Python sys Module” [Online] Tillgänglig: https://www.geeksforgeeks.org/python-sys-module/ [Hämtad 2021-05-03]

[20] Croston “RPIO documentation” [Online] Tillgänglig: https://sourceforge.net/p/raspberry-gpio-python/wiki/Examples/ [Hämtad 2021-05-21]

[21] el.st, ”Pulsbreddmodulering” [Online] Tillgänglig: https://el.st/?pulsbreddsmodulering [Hämtad 2021-05-28]

[22] Matt, “Simple Guide to the Raspberry Pi GPIO Header” [Online] Tillgänglig: https://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/ [Hämtad 2021-05-01]

[23] Nyman, Mats, , ”Agila metoder – Radikal evolution eller enkel evolution?” [Online] Tillgänglig: https://wwwwenellse.cdn.triggerfish.cloud/uploads/2016/09/agile-radikal-revolution-eller-enkel-evolution-mats-nyman-2010.pdf [Hämtad 2021-05-04]

[24] Axelsson, Mattias, ”Vad är scrum? Agil utveckling” [Online] Tillgänglig: https://www.happiness.se/artiklar/vad-ar-scrum [Hämtad 2021-05-16]

[25] Microsoft, “Så här väljer du den bästa agila metoden för ditt projekt” [Online] Tillgänglig: https://www.microsoft.com/sv-se/microsoft-365/business-insights-ideas/resources/how-to-choose-the-best-agile-methodology-for-your-project [Hämtad 2021-05-21]

[26] Amazon, ”Robotbil” [Online] Tillgänglig: https://www.amazon.se/OSOYOO-Educational-programmering-ton%C3%A5ringar-IOT-kontroll/dp/B08K2J9TQZ/ref=pd_rhf_se_p_img_1?_encoding=UTF8&psc=1&refRID=8N6BGBBATC1KTNXJNA8M [Hämtad 2021-05-01]

[27] Osoyoo, “OSOYOO Raspberry pi car V2.0 lesson 1: Basic install and coding” [Online] Tillgänglig: https://osoyoo.com/2020/08/01/how-to-use-osoyoo-model-pi-l298n-motor-driver-board-in-raspberry-pi-robot-car/ [Hämtad 2021-05-01]

[28] Colohouse, “What are the Benefits of VPS hosting” [Online] Tillgänglig: https://colohouse.com/the-benefits-of-vps-hosting/ [Hämtad 2021-05-01]

[29] Gunnarson. Lars, Svensson. Johan, “Körkortsboken”, Upplaga 33. Sveriges trafikskolors riksförbund, 2021. ISBN: 978-91-88971-15-9 [Hämtad 2021-05-01]

[30] Robinson. A,Hammond. M , “Python Programming On Win32” [Online] Tillgänglig: https://www.oreilly.com/library/view/python-programming-on/1565926218/ch20s01s02.html [Hämtad 2021-05-21]

[31] Ahlberg. J, “Hur fungerar självkörande fordon – och vilka fördelar och utmaningar finns?” [Online] Tillgänglig: https://se.ramboll.com/press/artiklar/hur-fungerar-sjalvkorande-fordon--och-vilka-fordelar-och-utmaningar-finns [Hämtad 2021-05-21]

10. Bilagor

Bilder och annat som kan tänkas läggas in i rapporten.

I. Motorklassens metoder

```
def moveForward(self):
    GPIO.output(self.In1,GPIO.HIGH)
    GPIO.output(self.In2,GPIO.LOW)
    GPIO.output(self.In3,GPIO.HIGH)
    GPIO.output(self.In4,GPIO.LOW)
    sleep(2)

def moveBackward(self):
    GPIO.output(self.In1,GPIO.LOW)
    GPIO.output(self.In2,GPIO.HIGH)
    GPIO.output(self.In3,GPIO.LOW)
    GPIO.output(self.In4,GPIO.HIGH)
    sleep(2)

def moveRight(self):
    GPIO.output(self.In1,GPIO.HIGH)
    GPIO.output(self.In2,GPIO.LOW)
    GPIO.output(self.In3,GPIO.LOW)
    GPIO.output(self.In4,GPIO.LOW)
    sleep(2)

def moveLeft(self):
    GPIO.output(self.In1,GPIO.LOW)
    GPIO.output(self.In2,GPIO.LOW)
    GPIO.output(self.In3,GPIO.HIGH)
    GPIO.output(self.In4,GPIO.LOW)
    sleep(2)

def moveFaster(self):
    self.p.ChangeDutyCycle(75)
    self.p2.ChangeDutyCycle(75)

def moveNormal(self):
    self.p.ChangeDutyCycle(50)
    self.p2.ChangeDutyCycle(50)

def moveSlow(self):
    self.p.ChangeDutyCycle(25)
    self.p2.ChangeDutyCycle(25)

def stop(self):
    self.p.ChangeDutyCycle(0)
    self.p2.ChangeDutyCycle(0)

def start(self):
    self.p.ChangeDutyCycle(60)
    self.p2.ChangeDutyCycle(60)

def turnOff(self):
    GPIO.cleanup()
```

II. HTML-sidan

```
5           <title>Car1</title>
6       </head>
7
8       <body>
9           <h3>Car 1</h3>
10          <p id="maintext"></p>
11          <script type="text/javascript">
12              var socket = io.connect('http://83.252.170.109:1605/');
13
14              window.onload = function() {
15                  if (sessionStorage.getItem('command_car1') !== null && sessionStorage.getItem('id_car1') === '1')
16                  {
17                      $("#maintext").html(sessionStorage.getItem('command_car1'));
18                  }
19              };
```

Public IP adress och port

Spara värden i hemsidan under en period eller tills man stängt fliken.

```
20              socket.on('message', function(msg) {
21                  const arg = msg.split(':');
22                  if (arg[0] == 1)
23                  {
24                      if (sessionStorage.getItem('command_car1') === null)
25                      {
26                          sessionStorage.setItem('id_car1', arg[0]);
27                          sessionStorage.setItem('command_car1', arg[1]);
28                          $("#maintext").html(sessionStorage.getItem('command_car1'));
29                      }
30                      else
31                      {
32                          sessionStorage.removeItem('command_car1');
33                          sessionStorage.setItem('command_car1', arg[1]);
34                          $("#maintext").html(sessionStorage.getItem('command_car1'));
35                      }
36                  }
37              });
38          </script>
39      </body>
40  </html>
```

Hämta värden om de finns sparade beroende på indata.