



CHALMERS

KANDIDATARBETESRAPPORT



Humanoid robot

Implementerad i Robot Operating System med autonom framdrivning,
ansiktsidentifiering och hälsningsfunktion

Jesper Carlsson

Ellinor Claesson

Annie Edvardsson

Christoffer Hansson

Sara Marklund

Wendy Mo

Mekatronik

Institutionen för signaler och system

Chalmers tekniska högskola

Göteborg, 2017

SSYX02-17-08

Förord

Under detta kandidatarbete, som omfattar 15 högskolepoäng över våren 2017, har vi fått enorm hjälp av vår handledare Jonas Fredriksson, som varje tisdag klockan åtta mötte upp oss för att lyssna till våra problem och hjälpa oss överkomma dessa. I framtagningen av den här rapporten har din input varit enormt värdefull. Stort tack för ditt tålamod.

Tack till Henrik Lindén, Nithin Suriac Kurien och Philip Bergström som arbetade med roboten i ett projekt före vårt kandidatarbete. Med er hjälp kunde vi snabbt sätta oss in i hur roboten fungerade.

Vi vill även tacka forskningsingenjörerna Reine Nohlborg, Göran Stigler och Jan Bragee från prototyplabbet för att vi har fått ta del av er stora erfarenhet inom tillverkning av metalledetaljer. Utan er hade vår bottenplatta inte varit i närheten av att vara så bra som den är nu.

Vi är också oerhört tacksamma för den hjälp vi fått från medlemmarna i ETA - E-sektionens Tekniska Avdelning. Med er hjälp blev designarbetet med batterikretsen helt plötsligt väldigt enkelt och stort tack för att ni lät oss använda er 3D-printer.

Den största delen av det här kandidatarbetet utfördes i RAM-labbet på Campus Johanneberg. Vi vill rikta ett stort tack till våra kurskamrater som också har arbetat där under våren. Tack för att ni har svarat på alla jobbiga frågor vi har ställt er och förlåt för att vi alltid har brett ut oss i rummet.

Sist men inte minst, tack till alla på institutionen Signaler och System på Chalmers och alla andra som på något sätt varit inblandade i det här projektet. Det här kandidatarbetet hade aldrig gått att genomföra utan er ovärderliga hjälp.

Chalmers tekniska högskola, Göteborg, maj 2017,
Annie, Christoffer, Ellinor, Jesper, Sara och Wendy

Abstract

This thesis describes the process of adding functionality to and enhancing a Meccanoid G15KS humanoid robot. The robot has been improved by replacing the original propelling motors to more powerful ones, installing batteries in order to eliminate the need of being connected to a power outlet, and by programming functionality to make it more autonomous. In its updated state, the robot moves faster than before. Additionally, the robot is equipped with facial recognition and obstacle avoiding software. When it recognizes a face, a hand wave motion is triggered.

Since this project has, to great extent, revolved around writing working code for different functions, the thesis also follows that pattern: each functionality is initially explained separately and later the different parts' correlation is presented. Some construction work has been done, these processes are described separately. The testing of the new system is then analyzed and the thesis is concluded with a discussion part. Finally, the project group's suggestions for future improvement is presented.

This project has resulted in a robot that is able to move autonomously until it detects a face, which then triggers a hand waving motion, or an obstacle, in which case the robot changes its direction in order to avoid collision. The thesis' problem statements that were established at the start of the project have been resolved.

This bachelor's thesis is written in Swedish.

Key words: ROS, robot, facial recognition, autonomy, jetson, meccanoid

Sammandrag

Denna rapport beskriver utvecklingsprocessen av en Meccanoid G15KS humanoid robot. Roboten har förbättras på flera sätt. De framdrivande motorerna har uppgraderats, batterier har kopplats in i kretsen för att eliminera behovet av strömtillförsel via vägguttag och roboten har programmerats till att röra sig framåt autonomt. Den uppdaterade versionen av roboten rör sig snabbare än den ursprungliga och den är utrustad med mjukvara för ansiktsigenkänning och hinderundvikning.

Eftersom projektet i hög grad har kretsat kring att skriva programmeringskod för olika funktioner följer även rapporten det mönstret: alla funktioner är först enskilt beskrivna och därefter följer information om hur funktionerna samverkar med varandra. Konstruktion har också varit en central del i arbetet och processen beskrivs separat. Därefter har det nya systemet testats och analyserats, vilket beskrivs i ett eget kapitel och sammanfattas i diskussionskapitlet. Slutligen presenteras projektgruppens förslag på framtida utveckling.

Resultatet av projektet är en robot som rör sig autonomt tills den upptäcker ett ansikte, då en vinkningsrutin startar eller tills ett hinder upptäcks, vilket får roboten att ändra riktning för att undvika kollision. Robotens beteende är i enlighet med de mål som utformades i projektets början och som har besvarats.

Nyckelord: ROS, robot, ansiktsigenkänning, autonomi, meccanoid, jetson

Innehåll

1	Inledning	1
1.1	Syfte och mål	2
1.2	Frågeställning	2
1.3	Avgränsningar	2
2	Teknisk beskrivning av robotens komponenter	3
2.1	Systemöversikt	3
2.2	Komponentbeskrivning av robotens utgångsläge	3
2.2.1	Meccabrain och servomotorer	4
2.2.2	Arduino UNO	5
2.2.3	Robot Operating System - ROS	5
2.2.4	Jetson TX1	6
2.2.5	OpenCV	6
2.3	Tillagd hårdvara	7
2.3.1	ZED Stereo Camera	7
2.3.2	Ultraljudssensorer - PING)))	7
2.3.3	MD23-motorkontroller och DC-motorer	8
2.3.4	Litium-polymerbatterier	8
2.3.5	Spänningsregulator	9
3	Bottenplatta	10
3.1	Krav och önskemål	10
3.1.1	Dimensionskrav	10
3.1.2	Konstruktionskrav	10
3.1.3	Övriga krav	11
3.2	Konceptgenerering	11
3.3	Slutgiltigt koncept och produkt	11
4	Systemintegration	13
4.1	Strömförsörjning	13
4.2	Kontroll av robotens armar	15
4.3	Kontroll av DC-motorer med MD23-motorkontroller	15
4.4	Integration av komponenter	17
5	Mjukvara	19
5.1	Ansiktsgigenkänning	19
5.1.1	Haar-cascade-algoritmen	19
5.1.2	Praktisk implementation	20
5.2	Topics och meddelanden i ROS	20
5.2.1	/servo	21
5.2.2	/waveDone	21
5.3	Robotens tillståndsmaskin	21
5.4	Implementation av ultraljudssensorer	22
6	Verifiering av robotens funktionalitet	24
6.1	Hastighetstest	24
6.2	Ansiktsidentifiering- och hälsningsfunktionstest	25
6.3	Hinderidentifieringstest	25
6.4	Hinderundvikningstest	26
6.5	Kombinerat framdrifts-, hinderundviknings- och hälsningstest	26

7	Diskussion	27
7.1	Utvärdering av robotens framdrivningshastighet	27
7.2	Utvärdering av hinderidentifiering och hinderundvikning	27
7.3	Utvärdering av ansiktsgenkänning	28
7.4	Utvärdering av robotens sammankopplade system	28
8	Summering och framtida rekommendationer	29
A	Konceptgenerering och eliminering för bottenplattans utformning.	32
A.1	Koncept	32
A.2	Val av koncept	34

Ordlista

Ansiktsgigenkänning Identifiering av ett ansikte digitalt, där en algoritm analyserar bilder och söker efter typiska ansiktsdrag. I detta projekt innebär ansiktsgigenkänning enbart att ett ansikte upptäcks, inte att det är kopplat till någon identitet.

Arduino UNO En elektronisk plattform, även kallad *mikrocontroller*, med öppen källkod för interaktion mellan elektroniska objekt.

Buss Ett system av gemensamma ledningar som binder samman datorkomponenter.

C++ Ett programmeringsspråk som bland annat används för programmering av inbyggda system.

CPU *Central Processing Unit*, enhet som exekverar instruktioner i en dator.

GPU *Graphics Processing Unit*, en processor som används för alla former av grafisk beräkning i en dator.

Headerfil En slags hjälpfil som används vid kompilering av andra källkoder.

I2C-buss *Inter-Integrated Circuit*. En typ av buss-konstruktion som används för att dirigera signaler.

IR *Infrared*. Elektromagnetisk strålning med längre våglängder jämfört med synligt ljus. Så kallade IR-sensorer kan användas för att bedöma ljusstyrka.

Jetson TX1 Dator framtagen i syfte att användas i robotprojekt. Datorn har en hög energieffektivitet och har en inbyggd plattform som underlättar automatisering.

Kinect-kamera Kamera utvecklad av Microsoft för spelkonsolen XBOX 360. Utrustad med RGB-kamera och IR-sensor.

Maskininlärning Område inom artificiell intelligens för att analysera mönster och konstruktion av algoritmer för att förutspå data.

MD23 En typ av motorkontroller, det vill säga en liten hjälpdator som avlastar huvuddatorn genom att styra hastighet, moment och andra kommandon hos DC-motorerna.

Meccabrain Den dator som följde med originalroboten. Anpassad för drift av servomotorer och motorer.

Meccanoid G15KS Robot som är producerad och distribuerad av Meccano. Har sedan tidigare en inbyggd Meccabrain. Det är denna robot som är grunden i detta projekt.

OpenCV Ett mjukvarubibliotek med verktyg för bland annat bildanalys.

Pulsbreddsmodulering En metod för att skapa en kontinuerligt varierbar effektmatning genom att slå på och av spänningen snabbare än vad den anslutna apparaten kan urskilja. Kallas ofta *PWM* från engelskans *pulse-width modulation*.

Ramverk Programvarumiljö som tillhandahåller funktionalitet för att underlätta utveckling av mjukvara.

RGB *Red-green-blue*. Används som grundfärger i färgblandningar för bildskärmar.

ROS Robot Operating System, ramverk för att underlätta programmering av robotar.

Ultraljudssensor Sensor som använder (för människor ej hörbart) ljud för att bedöma avstånd till objekt. Genom att skicka iväg en puls och sedan beräkna tiden det tar för pulsen att återvända efter att ha studsat mot ett objekt, kan avståndet till objektet beräknas.

Wrapper Ett stycke kod vars huvudsakliga syfte är att kalla på annan kod. Wrappern och den kod som ska anropas behöver inte vara skrivna i samma programmeringsspråk.

ZED-kamera *ZED Stereo Camera*, kamera som tack vare sina två linser kan skapa 3D-bilder med stöd för bland annat djupseende och objektidentifiering.

1 Inledning

En robot är en maskin som kan utföra en serie av komplexa uppgifter på egen hand. Robotar kan ha många tillämpningsområden och har använts inom tillverkningsindustrin under många årtionden. Med mer avancerad teknik kommer antalet möjliga uppgifter för robotar att öka. Denna utveckling syns i samhället och även i våra hem. Robotgräsklipparen är ett exempel på en produkt som många är beredda att bekosta för att slippa arbetet. En rimlig avvägning vid inköp av robotar för privat bruk är hur mycket den kostar kontra hur tråkig eller besvärande uppgiften som roboten tar över är. Det arbete roboten befriar kunden från gör den alltså värd inköpskostnaden. Däremot ute i industrin och i andra kommersiella miljöer är det främsta incitamentet ekonomi: om en robot kan utföra en arbetsuppgift mer effektivt än en människa utan att vara för dyr i inköp eller drift är det förståeligt att företaget vill ersätta personal med roboten.

Att arbetsuppgifter överläts till robotar är en trend som kommer bli mer påtaglig i framtiden. Stiftelsen för strategisk forskning bedömde år 2014 att hälften av de dåvarande arbetstillfällena i Sverige kommer att kunna överlätas till robotar innan år 2035 [1], vilket tyder på den enorma potential robotteknik har. Att över två miljoner arbetstillfällen i Sverige ska hanteras av robotar gör självklart att kraven ökar på vad robotar förväntas klara av. Här är kunskaper inom robotteknik en förutsättning för denna utveckling.

Speciellt kommer samhället de kommande åren se en stark utveckling inom autonoma robotar, alltså robotar som i stor grad klarar av att fatta självständiga beslut utifrån information från dess omgivning. Utöver ökad tillförlitlighet hos robotar är autonomi önskvärt eftersom att det avlastar nödvändigheten för människan att ständigt ha arbetet under uppsyn. Vid enkla monotona uppgifter i skyddade miljöer kan detta implementeras lätt. Tillämpningarna av autonoma robotar är många, enligt [2] är vanliga exempel robotar som används inom industrin för montering och transportrobotar på sjukhus. Självkörande bilar och kirurgiska operationsrobotar kräver väntat nog mer sofistikerade metoder för att autonomin ska fungera.

Om en omgivning innehåller många oförutsägbara faktorer, inom reglertekniken kallat *störningar*, behövs utöver ett välutvecklat datorsystem även bra beslutsfattning hos roboten för att autonomin ska fungera. Maskininlärning är således ett intressant fält att studera för att kunna utveckla och öka användningsområdena hos robotar. Algoritmer inom maskininlärning används för att förbättra robotars beslutsförmåga, vilket bidrar till att robotar blir bättre på att själva planera och resonera kring hur de bäst utför ett arbete och löser de problem som kan uppstå.

Det faller sig naturligt att människor designar robotar som efterliknar sig själva till en viss del. Exempelvis är datorseende ett bra verktyg för att implementera motsvarigheten till människans kanske viktigaste sinne, synen. Robotar som kan se och på så vis registrera dess omvärld kan använda informationen till att lättare uppmärksamma störningar i dess omgivning. Inom datorseende finns också ett element där ansiktsgenkänning ingår. Ansiktsgenkänning är fördelaktigt i situationer då ett program ska sätta igång en reaktion hos en robot när den registrerar en människa, vilket är ett delmoment för detta projekt.

För att befästa och vidareutveckla färdigheter som är viktiga inom denna gren av ingenjörsvärlden har Chalmersinstitutionen Signaler och System ett fortlöpande projekt där en humanoid robot kontinuerligt förbättras. Projektet har utgått från en robot av modell *Meccanoid G15KS* [3], som har utvecklats i samband med kurser på masterprogrammet Systems, Control and Mechatronics. Det är med utgångspunkt i det tidigare projektet, [4], som det här projektet utförs.

1.1 Syfte och mål

Målet med projektet är att vidareutveckla en humanoid robot, av modelltyp Meccanoid G15KS, som sedan tidigare redan har genomgått viss modifikation. Detta ska ske genom att förse roboten med ny hårdvara och sedan programmera denna så att roboten autonomt kan ta sig fram i en korridor med en hastighet dubbelt så stor som ursprunglig hastighet samt hälsa på mötande människor. Dessutom är ett önskemål att förändringarna ska höja robotens estetik. Syftet är att de nya funktionerna ska förse framtida projektgrupper som arbetar med roboten med en bra plattform att bygga vidare där det är enkelt att både lägga till ny hårdvara och att programmera in ny funktionalitet i den befintliga hårdvaran.

1.2 Frågeställning

För att uppnå målet har följande frågeställningar tagits fram:

- Hur ska roboten modifieras för att uppnå en hastighet som är minst den dubbla (0,5 m/s) jämfört med hastigheten i utgångsläget (0,25 m/s)?
- Hur ska roboten skilja på olika hinder, såsom människor och objekt?
- Hur kan roboten interagera med människor som den möter?
- Hur ska roboten ta sig fram genom en korridor utan att kollidera med människor eller andra hinder på vägen?

1.3 Avgränsningar

Följande avgränsningar har definierats för att testa robotens funktionalitet:

- Roboten kommer köras på plan mark. Mer komplicerade miljöer, till exempel trappor, bedömer projektgruppen som inaktuellt på grund av de tekniska svårigheter som detta funktionskrav skulle tillföra.
- Roboten förväntas inte hantera stängda dörrar eller andra hinder som kräver åtgärder utöver att roboten ska kunna förflytta sig runt dessa.
- Roboten förväntas endast identifiera och därmed interagera med människor som kommer framifrån, det vill säga i den riktning som kameran ska sitta. Med andra ord kommer roboten inte behöva hantera fall då personer närmar sig bakifrån.

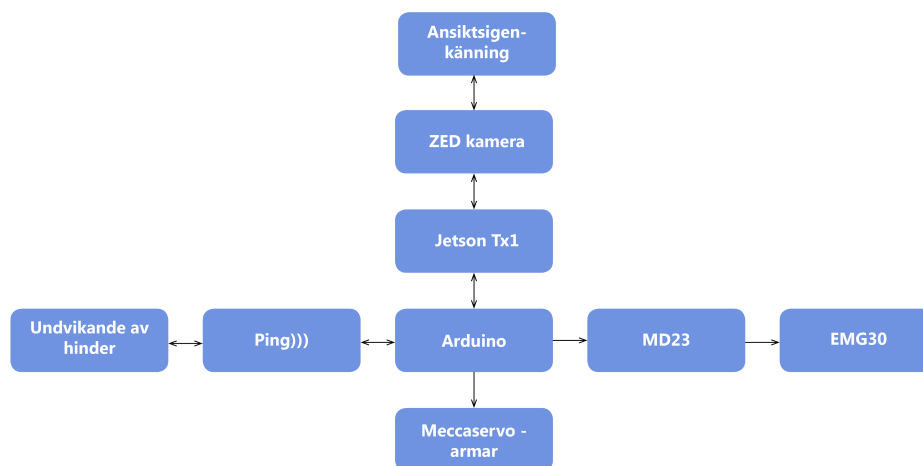
2 Teknisk beskrivning av robotens komponenter

I detta kapitel beskrivs de tekniska komponenter som använts i projektet. Först ges en översikt över robotens system vid projektets slut, därefter en beskrivning av hur roboten var uppbyggd vid projektets start, och slutligen beskrivs de komponenter som är tillagda i det här projektet.

2.1 Systemöversikt

Robotens hjärna är datorn Jetson TX1. För registrering av robotens omgivning har en ZED Stereo Camera anslutits, vilken tar bilder för vidare analys med funktioner i mjukvarubiblioteket OpenCV. Robotens rörelser styrs via ramverket Robot Operating System (ROS) som möjliggör enkel kommunikation mellan mikrokontrollers och mjukvaruprogram. Två 12 V DC-motorer styrs av en MD23-motorkontroller och används för att ge systemet framåtdrift. För att länka samman roboten med DC-motorerna har en bottenplatta konstruerats, där robotens komponenter förvaras.

I Figur 1 visas en översikt över hur det färdiga systemet är sammankopplat. ZED-kameran och mikrokontrollern, Arduino UNO, är kopplade direkt till datorn och Arduinon är i sin tur ansluten till fler komponenter. Kameran skickar ständigt information i form av bilder till datorn (Jetson TX1). När datorn identifierar ett ansikte informeras mikrokontrollern som styr armen (Meccaservoarmar) till en vinkning. Samtidigt skickar mikrokontrollern också signaler till DC-motorerna (EMG30) via motorkontrollern (MD23) att stanna. Två ultraljudssensorer av modellen PING))) kommunicerar med mikrokontrollern när den registrerar objekt i robotens närhet som då vidtar åtgärder för att svänga undan från hindret.

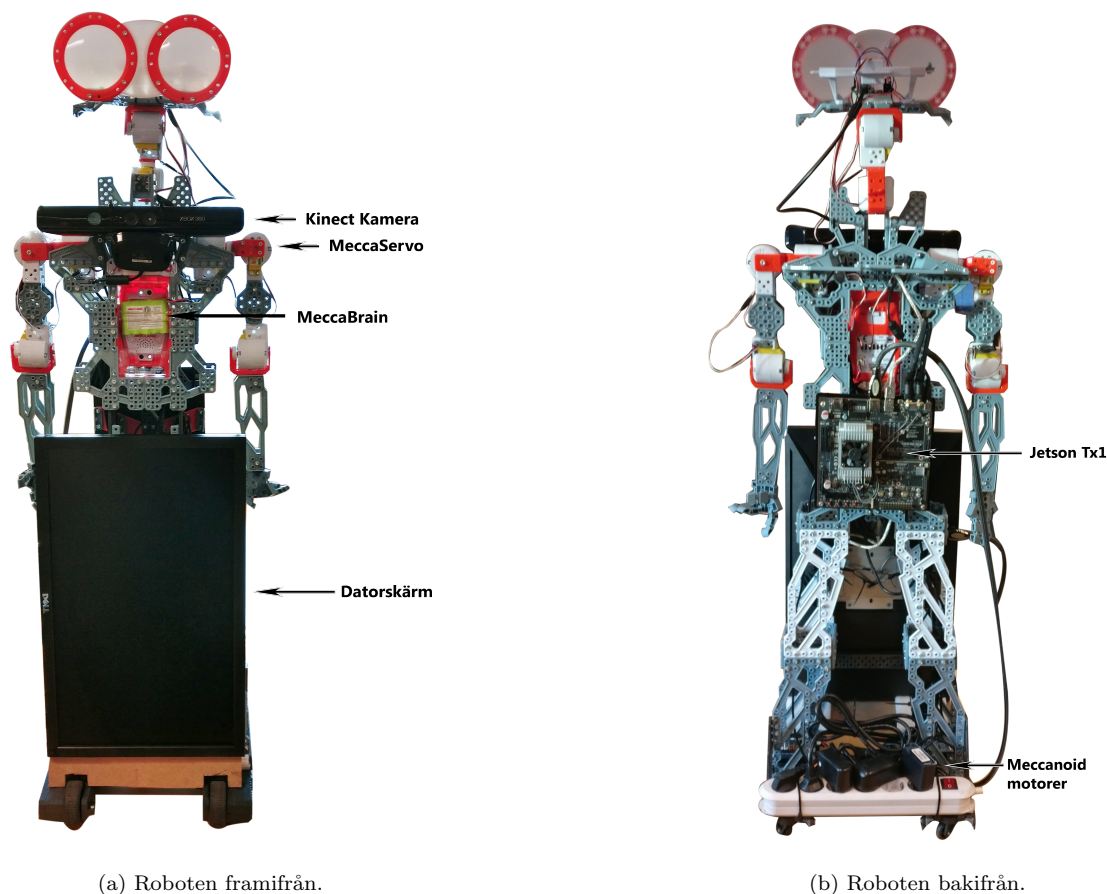


Figur 1: Ett översiktligt blockschema på systemet.

2.2 Komponentbeskrivning av robotens utgångsläge

Den robot av modell Meccanoid G15KS som detta projekt behandlar har bearbetats i tidigare kurser och projekt. Figur 2 visar roboten fram- och bakifrån så som den såg ut vid detta projekts början. Den ursprungliga Meccabraindatorn hade kompletterats med en starkare dator, Jetson TX1, men all funktionalitet var ännu inte överförd. Fötterna och huvudet styrdes av Jetson TX1 medan armarna fortfarande styrdes av Meccabrain:en.

Roboten var utrustad med en Kinect-kamera som är försedd med både IR-teknik och RGB-kamera och med hjälp av en algoritm klarade roboten av att känna igen ansikten. Skärmen som syns i Figur 2a användes för att demonstrera vad roboten ser. Roboten kunde även köras framåt långsamt med en topphastighet strax under 0,25 m/s genom styrning av en användare via ROS. Ansiktsgenkänning och framåtdrift styrdes via Jetson TX1. All kod var skriven i programspråket C++ och bibliotek, såsom OpenCV, användes för att implementera de ovannämnda funktionerna.



Figur 2: Roboten framifrån och bakifrån vid projektets start.

2.2.1 Meccabrain och servomotorer

Robotens huvud- och armrörelser styrdes ursprungligen av en Meccabrain-dator, designad med förprogrammerade uppgifter så som tal, drift och röstsstyrning. Till datorn finns även ett bibliotek med standardprogram som användaren kan nyttja för att få servomotorer tillverkade av företaget Meccano att röra sig. En robotarm består av tre sammanlänkade servomotorer där varje länk kallas för en kedja. Kodmässigt definieras en kedja av datatypen `MeccaBrain` och olika servomotorer i kedjan designas ett nummer för identifiering.

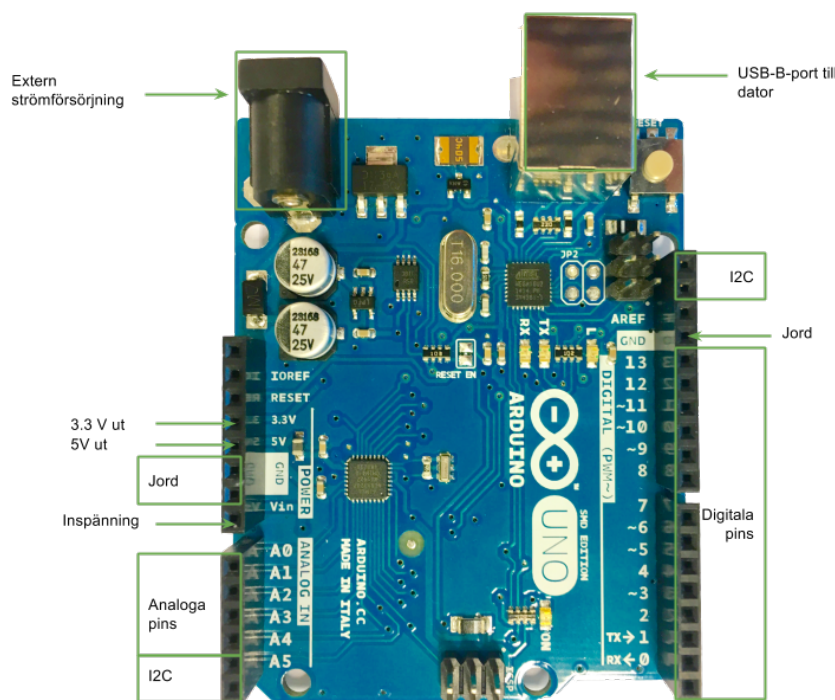
Med hjälp av standardfunktioner såsom `setPosition()` kan servomotorernas position ändras. Varje servomotor kan röra sig fritt mellan 0° och 180° . I biblioteket anges servomotorns position av ett tal mellan 0 och 255. I hjälpfilen `MeccaBrain.h` från Meccano definieras 0 som att servomotorn är roterad fullt medurs och 255 innebär fullt moturs. Varje servomotor har en inbyggd lysdiod, vilket är behjälpligt

vid felsökning. Koden kan utformas så att lysdioden lyser upp när roboten har exekverat en viss kod, vilket indikerar om kommunikationen mellan servomotorerna och mikrokontrollerkortet fungerar som den ska.

2.2.2 Arduino UNO

Arduino UNO, i kommande stycken refererat till som *Arduino(n)*, är ett mikrokontrollerkort som möjliggör enkel kommunikation mellan elektroniska enheter [5], till exempel servomotorer. UNO-modellen är lämplig för det här projektet eftersom den är robust, liten och billig samtidigt som den har tillräckligt många pins för projektets ändamål.

Programmering av mikrokontrollern sker i utvecklingsmiljön Arduino IDE. Koden kan sedan laddas upp till ett Arduino-kort via USB-B-porten, se Figur 3. Kontrollern kan drivas av en extern spänningskälla via uttaget längst upp till vänster i figuren. Kontrollern har även två pins för utspänning, en för 3,3 V och en för 5 V. Det finns sex analoga pins varav två (A4 och A5) kan användas som en I2C-buss. Arduinon har även 14 digitala pins varav sex stycken kontrolleras med *pulsbreddsmodulering* (PWM).



Figur 3: En *Arduino Uno* med olika komponenter utmärkta.

2.2.3 Robot Operating System - ROS

För att samordna koordinationen mellan robotens olika komponenter - armar, huvud och DC-motorer som driver roboten - används *Robot Operating System*. Utvecklingsteamet beskriver ROS i [6] som ett mjukvaruramverk som abstraherar de olika komponenterna till så kallade *noder*. En nod definieras som en process som kan utföra en handling eller en beräkning. Om flera noder kombineras bildas en *graf*.

Denna abstraktion möjliggör kommunikation mellan robotens delar, oavsett om de är mikrokontrollers, motorer eller annat.

Grundpelaren i ROS är *topics* - kommunikationskanaler som komponenter skickar meddelanden över. För att en nod i roboten ska kunna kommunicera över en viss topic måste typen av meddelandet som skickas vara samma som topic:ens - det går alltså inte att skicka till exempel en textsträng på en topic som hanterar arrayer. Varje topic har ett namn och en meddelandetyper, exempelvis heltal (`int`), textsträng (`string`) eller mer komplexa typer som färgkoder (`ColorRGBA`). Det är också möjligt att skicka tomma meddelanden av typen `Empty`. Med hjälp av topic:ens funktionalitet kan noderna i en graf således agera separat, något som gör att deras interna funktionalitet inte längre spelar någon roll för deras förmåga att kommunicera med varandra. De behöver inte heller vara skrivna i samma programmeringsspråk, vilket är en av ROS fördelar.

En nod som skickar meddelanden kallas *publisher* och mottagaren kallas *subscriber*. En subscriber har nästan alltid en så kallad *callback-funktion* implementerad i koden, vilket är en funktion som ligger och väntar på ett meddelande för att sedan exekveras. Figur 4 visar en allmän översikt av hur ROS kan se ut. Topic:en, */example*, har tre noder - en publisher och två subscribers. Den vänstra noden producerar meddelanden av typen textsträng och de två högra noderna lyssnar på topic:en och väntar på meddelanden från publishern. För att möjliggöra kommunikation över USB-bussen med exempelvis en Arduino används det inbyggda protokollet `rosterial`. Med `rosterial` paketeras information så att data överförs och översätts på ett smidigt sätt mellan hård- och mjukvara. Funktionen `spinOnce` används för att hantera kommunikation mellan olika noder.



Figur 4: En allmän översikt av ROS i ett system med en tre noder - en publisher och två subscribers.

2.2.4 Jetson TX1

Jetson TX1, en dator utvecklad av företaget NVIDIA, är utrustad med Linux-miljö (Ubuntu 16.04), vilket gör den kompatibel med ROS. Med kraftfull CPU (Central Processing Unit) och GPU (Graphics Processing Unit) blir datorn, som dessutom är energisnål, ett utmärkt verktyg för bildbehandling. Enligt NVIDIA [7] gör datorns höga prestanda och låga spänningsförsörjning datorn perfekt för robotprojekt. En nackdel med datorn är att den endast har ett USB 3.0-uttag, varför en USB-hubb har anslutits för att kunna hantera alla de USB-komponenter som har krävts i projektet.

2.2.5 OpenCV

OpenCV (*Open Source Computer Vision Library*) beskrivs i [8] som ett standardmjukvarubibliotek inom bland annat bildanalys, datorseende och maskininlärning. Biblioteket innehåller algoritmer för

ansikts- och objektigenkänning, produktion av 3D-modeller och många fler funktioner. I det här projektet används OpenCV främst för bildanalys av bilder som kameran registrerar för att undvika objekt och människor i en korridor. Förutom mångsidigheten i OpenCV:s biblioteket är en stor fördel dess kompatibilitet med ZED-kameran.

2.3 Tillagd hårdvara

Robotens framdrivningshastighet har ökat genom att nya, starkare DC-motorer har ersatt de ursprungliga. Ansiktsigenkänningen har förbättrats genom att Kinect-kameran har ersatts med en ZED Stereo Camera och ultraljudssensorer har implementerats för att hjälpa roboten att undvika hinder. För att strömförsörja roboten oberoende av vägguttag har en batterikrets skapats. Nedan beskrivs robotens nya komponenter i detalj.

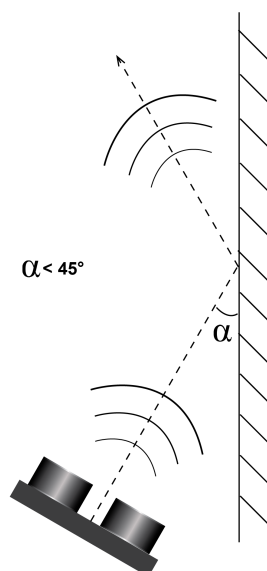
2.3.1 ZED Stereo Camera

ZED Stereo Camera är tillverkad av företaget Stereolabs. Kameran fångar bilder med hjälp av två inbyggda RGB-kameror som sitter placerade 12 cm ifrån varandra. Detta möjliggör ett djupseende på upp emot 20 m samt rendering av 3D-bilder. Kameran har en hög bildfrekvens och kan med lägre bildkvalitet ta upp till 100 bilder per sekund. Hög bildfrekvens i kombination med kamerans breda synvinkel på 110° gör enligt [9] ZED-kameran till ett utmärkt alternativ för filmning av rörelser som kan användas för bildbehandling.

Kameran ansluts till en dator via en USB 3.0-kabel. Ett antal standardprogram för filmning och bildanalys medföljer produkten. En stor fördel med kameran i detta projekt är att den är kompatibel med Jetson TX1, ROS och OpenCV. Stereolabs har själva dokumenterat och publicerat, [10], en wrapper för ROS som gör det möjligt att enkelt skicka över information från kameran till olika noder. Utöver USB-anslutningen behöver kameran ingen extern spänningsförsörjning.

2.3.2 Ultraljudssensorer - PING)))

Ultraljudssensorn PING))) är utvecklad av företaget Parallax Inc. Sensorn är ett typexempel på en komponent som är kompatibel med en Arduino. Sensorn kan enligt [11] bedöma avståndet till objekt som är mellan 0,02-3 m bort. Sensorn arbetar genom att skicka ut en ljudvåg i en frekvens som är högre än vad människan kan uppfatta och därefter räkna hur lång tid det tar för ljudet att studsas tillbaka. Avståndet till objektet kan då beräknas eftersom tiden mellan in- och utsignal liksom ljudhastigheten är känd. För att kommunicera med Arduinon ansluts sensorn till uttagen jord, 5 V och en digital signalpin. En nackdel med PING))) är enligt [12] att den enbart kan uppfatta objekt som kommer inom en vinkel av 20° framifrån. Dessutom kan inte sensorn uppfatta objekt vars yta är vinklad mindre än 45° mot sensorn då ljudet inte reflekteras tillbaka mot sensorn i dessa fall. Istället studsar ljudvågen framåt och sensorn får då ingen information om objektet, se Figur 5.



Figur 5: Illustration av en ultraljudssensor (nere till vänster i bild) som skickar iväg en puls mot en vägg. Då roboten är riktad mot en vägg med en vinkel under 45° registreras inte väggen av sensorerna.

2.3.3 MD23-motorkontroller och DC-motorer

MD23 är en avancerad motorkontroller för 12 V-motorer som hanterar upp till 3 A per motor. Motorerna som används för framdrivning i det här projektet är ett par DC-motorer av modell EMG30. DC-motorerna behöver 12 V och är lämpliga att användas till små eller medelstora robotapplikationer. DC-motorerna erbjuder feedback till användaren och är kompatibla med motorkontrollern MD23 enligt [13]. Genom styrning via MD23 kan, enligt [14], DC-motorerna köras i 255 olika hastigheter där 255 är full fart framåt, 128 är stopp och 0 är full fart bakåt.

Motorkontrollern styrs enligt [15] av en I2C-buss, *Inter-Integrated Circuit*. I2C-bussen har fyra ledare; Serial Clock (SCL), Serial Data (SDA), 5 V-input och jord. SCL synkroniserar den data som överförs till kontrollen så att DC-motorerna arbetar synkroniserat, medan SDA levererar datan och 5 V-input förser kontrollen med spänning. SCL och SDA är öppna kollektorer, vilket betyder att de bara kan generera låga ut signaler. För att få höga ut signaler måste, enligt [16], två "pull up"-resistorer parallellkopplas mellan SCL och 5 V samt mellan SDA och 5 V. En pull up-resistor fungerar som garant för att den logiska kretsen har en hög ingångsspänning oavsett omständigheterna. På så sätt vet I2C-bussen alltid vilken jordreferens den har, vilket behövs för att den ska kunna kommunicera med resterande krets.

2.3.4 Litium-polymerbatterier

Litium-polymerbatterier används enligt [17] ofta till egenbyggda elektronikenheter eftersom de är relativt lätta och kan tillverkas i många olika storlekar. Batterierna har lång batteritid och klarar hög effekt. En nackdel är dock att batterierna riskerar att fatta eld om de genom punktering exponeras för luft. Batterierna är uppbyggda av en eller flera seriekopplade celler, där varje cell har en nominell spänning på 3,7 V. Detta betyder att ett batteri med nominell spänning 11,1 V består av tre stycken

seriekopplade celler. Kapaciteten på ett batteri anges i mAh. De batterier som används i detta projekt har en kapacitet på 4200 mAh och en spänning på 11,1 V.

2.3.5 Spänningsregulator

Som nämnt ovan drivs robotens krets av två 11,1 V batterier. För att anpassa spänningen till de komponenter som ska spänningmatas med 5 V används en spänningsregulator. Den regulator som används i projektet är switchad och kan omvandla 3,2-40 V till 1,25-37 V samt leverera en utström på max 2 A enligt [18].

3 Bottenplatta

För att uppnå önskad framdrivningshastighet av roboten behövdes nya, starkare motorer. Motorbytet krävde även en ny infästningsanordning vilket ledde till bygget av en ny bottenplatta. Utöver att sammanlänka roboten med den nya framdrivningsanordningen är bottenplattans syfte att ge tyngd och därmed stabilitet till systemet samt att kapsla in robotens elektronik. I detta kapitel beskrivs de krav som tagits hänsyn till vid bottenplattans konstruktion samt hur konceptgenereringen gick till.

3.1 Krav och önskemål

En kravspecifikation, se Tabell 1, har tagits fram för att ge en strukturerad översikt över relevanta krav och önskemål som ställts på bottenplattans konstruktion. Kravspecifikationen anger riktvärden på krav och önskemål samt verifieringsmetoder för dessa.

Tabell 1: Kravspecifikation för bottenplatta.

Kategori		Krav/ Önskemål	Verifieringsmetod
Dimension	1.1 Bredd	K $330 < X < 370$	Mätning
	1.2 Höjd	K $60 < X < 150$ mm Ö $60 < X < \text{mm}$	Mätning
	1.3 Djup	K $320 < X < 370$ mm	Mätning
	1.4 Vikt	K $X < 6$ kg Ö $X < 3$ kg	Mätning
	1.5 Innervolum	K $X > 0,0063$ m ³	Mätning
Konstruktion	2.1 Tillgänglighet - öppningsbar	K	Konstruktionstest
	2.2 Möjlighet att montera roboten ovanpå bottenplattan för stabil drift	K	Konstruktionstest
	2.3 Möjlighet att montera fram- & bakhjul på samma höjd	K	Konstruktionstest
Hållfasthet	3 Bära robotens vikt under drift med försumbar deformation	K	Mätning
Störningar	4 Konstruktionen stör inte robotens huvudfunktioner	K	Drifttest

3.1.1 Dimensionskrav

Bottenplattans dimensionskrav har främst baserats på robotens utseende samt formen på den elektronik som ska byggas in. Storleken är en balansgång mellan att hålla bottenplattan så liten som möjligt för att underlätta hinderundvikning och att uppnå den stabilitet som en större platta ger. Minsta mått på bredd är avståndet mellan ytterkanterna på robotens ben vilket måste uppfyllas för att roboten ska kunna monteras direkt på plattan. Den övre gränsen är för att kunna minimera storleken för att underlätta manövrering. Djupet har begränsats av att plattan ska vara stabil och innervolumen samt höjden är sådan att den elektronik som ska få plats i lådan ryms utan att det blir för mycket oanvänt utrymme, samtidigt som framtida påbyggnationer ska få plats. Viktkravet säkerställer att motorerna orkar driva roboten med önskad hastighet på 0,5 m/s.

3.1.2 Konstruktionskrav

Konstruktionskraven är krav som satts för att bottenplattan ska vara lätt att hantera vid montering, reparationer eller justeringar av roboten samt för att hålla roboten upprätt under drift.

Kravet "Tillgänglighet - öppningsbar" innebär att användaren ska kunna göra justeringar eller tillägg till den elektronik som finns i bottenplattan utan att behöva montera loss roboten. Plattan behöver alltså ha någon form av öppning eller lucka. Kravet "Möjlighet att montera roboten ovanpå bottenplattan för stabil drift" innebär att när roboten är monterad på plattan och körs ska den vara stabil. Kravet "Möjlighet att montera fram- och bakhjul i samma höjd" säkerställer att bottenplattan är parallell med underlaget och ger därmed inte roboten en lutning som förskjuter dess tyngdpunkt i det horisontella planet vilket skulle kunna orsaka instabilitet.

3.1.3 Övriga krav

Hållfasthetskravet ”bära robotens vikt utan synlig deformation” har främst tagits med för att konstruktionen inte ska kunna förskjutas i något led. Till exempel bör inte eventuella borttagbara delar inkludera stora vertikala sidostycken då detta försvagar hållfastheten avsevärt. Kravet ”Konstruktionen stör inte robotens huvudfunktioner” innebär till exempel att bottenplattan ska vara utformad så att kablar ska kunna dras obehindrat mellan komponenter i lådan och komponenter monterade på roboten.

3.2 Konceptgenerering

Utifrån kravprofilen har koncept genererats parallellt med konsultation med forskningsingenjörerna i prototyplabbet på Chalmers. Stålplåt valdes som material för sin höga hållfasthet, seghet och svetsbarhet. Vid val av plåtens tjocklek beaktades vilken massa konstruktionen skulle få samt vilken tjocklek som är lättast att arbeta med. En tyngre bottenplatta sänker robotens tyngdpunkt vilket gör den mer stabil samtidigt som extra vikt försämrar framdrivningen av roboten. Tjockleken på plåten bestämdes till 1,5 mm vilket kommer göra bottenplattan tillräckligt robust utan att den blir onödigt tung.

Konceptgenereringen resulterade i tolv koncept med sex olika metoder för att göra bottenplattan öppningsbar och två metoder att montera roboten ovanpå. En kombination av Pugh-matriser och konsultation med forskningsingenjörerna i prototypverkstaden användes för att sälla bland koncepten. Samtliga koncept samt koncepteliminering med hjälp av Pugh-matris återfinns i Appendix A.

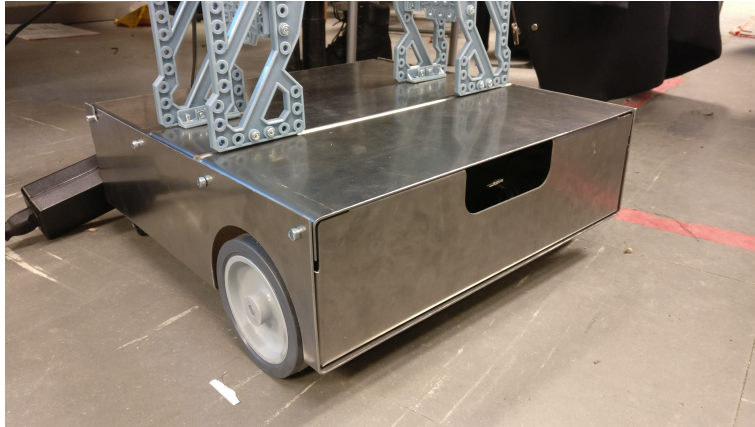
3.3 Slutgiltigt koncept och produkt

Det koncept som fick bäst resultat visas i Figur 6. Konceptet består av en grundplåt som är bockad i två sidor, vilka representerar bottenplattans höger- och vänstersida. På plåtens båda sidor är en profil utskuren där framhjulen är placerade. Bakhjulen fästes på grundplåtens bakre del, på dess undersida.

Utöver grundplåten består konceptet av två lock av bockad plåt som fästs i grundplåten med hjälp av popnitar och skruvar. Roboten står på en plåt som är bockad på alla fyra sidor för att få en så hög hållfasthet som möjligt. Denna plåt svetsas fast i mitten av grundplattan. Vid drift är utböjningen på plåten försumbar.

De sista komponenterna i bottenplattan består av dess fram- och baksida. Dessa är identiska och är konstruerade genom bockning på alla fyra sidor. Plåten har ett hål i överkant för att användaren ska kunna hålla i locket och därmed förenkla montering och demontering av denna. Slutligen lades plexiglas samt tygmatta inuti bottenplattan, dels för att förhindra elektrisk ledning genom stålplåten och dels för att dämpa vibrationer vid drift.

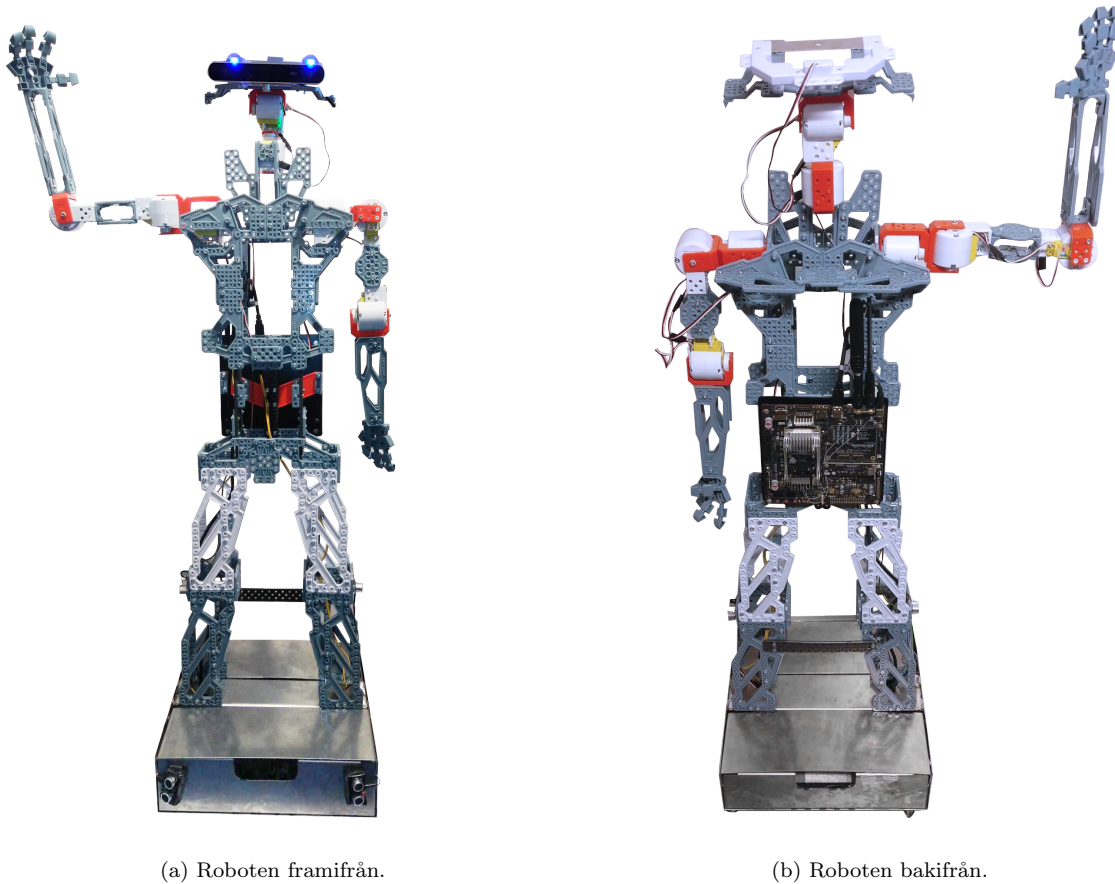
Bottenplattan uppfyller kravspecifikationen eftersom dess yttermått är 350 x 350 x 100 mm, innervolymen är 0,011 m^3 och den väger 4,32 kg. Roboten kan även monteras stabilt på bottenplattan utan att denna deformeras. I och med konceptets utformning är bottenplattan parallell med golvet, öppningsbar och stör inte några av robotens funktioner.



Figur 6: Vy av bottenplattans utsida.

4 Systemintegration

Funktionerna som tidigare styrdes av robotens Meccabrain har förflyttats till den nya datorn, Jetson TX1. Detta resulterade i att Meccabrain inte längre uppfyllde någon funktion för roboten och den har således plockats bort. Likaså har Kinect-kameran, som har ersatts av ZED-kameran, och den otympliga skärmen, som tidigare var fastskruvad på roboten, plockats bort från roboten. Den färdigkonstruerade roboten syns i Figur 7. Alla komponenter styrs nu via Jetson TX1 som sitter monterad på robotens rygg. Arduinon, som har anslutits till datorn, skickar styrsignaler till servomotorerna, ultraljudssensorerna samt motorkontrollern MD23. Robotens armar är kopplade till Arduinon så att roboten kan utföra en vinkning då ZED-kameran (som har monterats på robotens huvud) ser ett ansikte. Även DC-motorerna får signaler från Arduinon via MD23 och dessa aktiveras antingen då ultraljudssensorerna upptäcker ett hinder som roboten försöker undvika eller då roboten stannar för att hälsa på en människa.



Figur 7: Roboten framifrån och bakifrån efter att alla komponenter har installerats.

4.1 Strömförsörjning

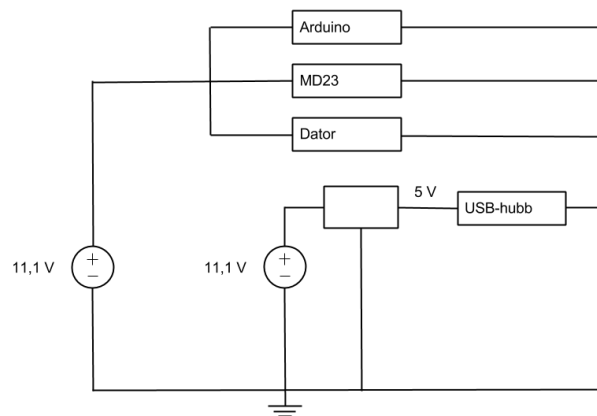
Robotens komponenter drevs ursprungligen av ström från en förgreningsdosa vilken var kopplad till ett vägguttag. Då strömkabeln påverkade robotens rörelse och därigenom försvårade användning av roboten, kopplades roboten till en portabel strömkälla i form av en batterikrets. För att bygga batterikretsen behövde de olika komponenternas spännings-, ström- och effektbehov beräknas. Den komponent

som har störst spänningsbehov är motorkontrollern MD23, som i sin tur driver DC-motorerna, vilken enligt specifikation ska matas med cirka 12 V. Batterikretsen har därför dimensionerats utifrån 11,1 V-batterier då de är tillräckligt starka för att driva MD23 med DC-motorerna. Spänningen 11,1 V kommer driva alla komponenter exkluderat USB-hubben vilken maximalt ska matas med 5 V. Tabell 2 visar en översikt över de komponenter som kräver direkt strömförsörjning och hur mycket spänning, ström och effekt de förväntas fordra.

Tabell 2: Spänningsbehov, strömbehov och effektkonsumtion för de olika komponenterna. Spänningen är bestämd från batterikretsen. * innebär att värdet är taget från komponentens specifikation och resterande värden är beräknade med Ohms lag: $P = U \cdot I$.

Komponenter	Spänningsbehov [V]	Strömbehov [A]	Effektkonsumtion [W]
Jetson TX1	11,10	1,35	15,00*
MD23	11,10	0,30*	3,33
Arduino	11,10	0,10*	1,11
USB-hubb	5,00	2,00*	10,00
Förluster i spänningsregulatorn	(5,00)	(0,22)	1,11
Totalt	43,30	3,97	30,55

Ett batteri kan leverera olika spänning beroende på hur laddat det är och USB-hubben som används i projektet fungerar bäst då den matas med en konstant spänning på 5 V. För att säkerställa att USB-hubben matas med konstant spänning används en spänningsregulator som reglerar ner spänningen från ett 11,1 V-batteri till 5 V. Den strömförsörjande kretsen består alltså av två batterier och en spänningsregulator. Ett av batterierna driver, tillsammans med spänningsregulatorn, de komponenter som kräver 5 V och det andra batteriet driver de komponenter som behöver 11,1 V. I Figur 8 visas ett schema över kretsen där det kan avläsas hur MD23, Arduinon och datorn är parallellkopplade till det ena batteriet med spänning 11,1 V. USB-hubben, som är ansluten till datorn, är kopplad till det andra batteriet efter att spänningen sänkts till 5 V via spänningsregulatorn. Samtliga komponenter jordas gemensamt för att alla signaler ska tolkas lika. Batterierna består av tre LiPo-celler med total nominell spänning 11,1 V och har en kapacitet på 4200 mAh.



Figur 8: Schema över batterikretsen som driver roboten. Kretsen förser MD23, Arduino och dator med 11,1 V och USB-hubben med 5 V. Den tomma rutan illustrerar spänningsregulatorn.

I en spänningsregulator förekommer förluster vilka också redovisas i Tabell 2. Den spänningsregulator som har använts är variabel och switchad med en effektivitet på >90% och en maximal utström på 2 A. Enligt [19] kan ekvation 1 användas för att beräkna de maximala förlusterna i spänningsregulatorn.

$$\eta = \frac{U_{UT} \cdot I_{UT}}{U_{UT} \cdot I_{UT} + P_F} \quad (1)$$

där η är effektivitetstalet, U_{UT} är utspänning, I_{UT} är utström och P_F är den totala effektförlusten i spänningsregulatorn. Om P_F löses ut ur ekvationen, enligt ekvation (2), fås effektförlusten 1,11 W.

$$P_F = \frac{U_{UT} \cdot I_{UT}}{\eta} - U_{UT} \cdot I_{UT} = \frac{5 \cdot 2}{0,90} - 5 \cdot 2 = 1,11 W \quad (2)$$

Övriga förluster i kretsen har försumrats och robotens teoretiska drifttid beräknas med hjälp av ekvation (3):

$$\frac{q}{I} = t \quad (3)$$

där q är batteriets kapacitet i amperetimmar, I är strömmen i ampere och t är tiden i timmar. I Tabell 3 redovisas den teoretiska drifttiden för varje batteri.

Tabell 3: Kapacitet, ström och teoretisk drifttid för de två olika batterierna i robotens krets.

	Kapacitet [Ah]	Ström [A]	Teoretisk drifttid [h]
Batteri för 11,1 V krets	4,20	1,75	2,40
Batteri för 5 V krets	4,20	2,22	1,89

4.2 Kontroll av robotens armar

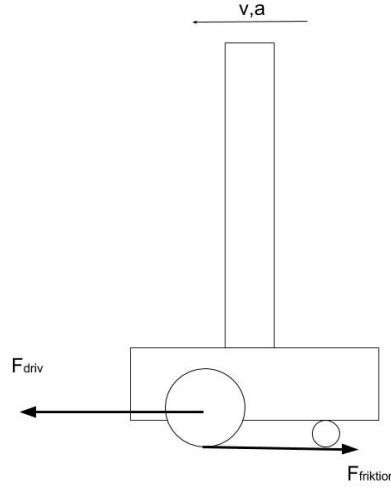
När armarna styrdes av den ursprungliga datorn, Meccabrain, kunde roboten endast utföra ett visst antal förutbestämda rörelser. För att få utökad kontroll av armarna kopplades de om från Meccabrain:en till Arduinon genom att sammankoppla mikrokontrollernas GPIO-pins och servomotor. Jord, 5 V-ut och signalkabel behöver vara inkopplade för att data ska kunna skickas och tas emot.

För programmering används Arduino IDE. Eftersom servomotorerna är specialtillverkade av Meccano inkluderas deras headerfil (`MeccaBrain.h`) för att underlätta kontrollen. Små, stegvisa ökningsar möjliggör smidig övergång mellan olika servomotorlägen. Som tidigare nämnt i kapitel 2.2.1 kan varje servomotor röra sig mellan 0 och 180°, vilket översätts till ett tal mellan 0 och 255 i koden.

4.3 Kontroll av DC-motorer med MD23-motorkontroller

Robotens ursprungliga motorer har ersatts av ett par starkare 12 V DC-motorer som styrs av motorkontrollern MD23. Motorkontrollern är i sin tur kopplad till Arduinon och användaren kan sedan skicka signaler från datorn till Arduinon. Kommunikationen möjliggör styrning samt drift av DC-motorerna på ett kontrollerat sätt. Vid programmering används ofta hjälpfiler som stöd för att underlätta användningen av inbyggda funktioner hos komponenter. För att förenkla styrningen av DC-motorerna används headerfilen `wire.h`. Eftersom Arduinon inte har tillräckligt stark utspänning för att driva DC-motorerna själv har MD23 kopplats till strömförsörjningskretsen, som beskrivet i kapitel 4.1.

För att säkerställa att DC-motorerna är tillräckligt kraftiga för att framföra roboten i önskad hastighet har förenklade beräkningar gjorts. I Figur 9 visas en friläggning med de krafter som påverkar roboten vid framåtdrift. Den stående rektangeln är en förenkling av robotens geometri. Då roboten kommer framföras i maximalt 0,5 m/s har luftmotståndet försumrats. Eventuella friktionsmoment i DC-motorn har också försumrats i beräkningarna.



Figur 9: Friläggning av roboten.

Robotens och bottenplattans totala massa, m , har uppskattats till 10 kg för att ge en marginal till framtida påbyggnationer. Rullningsfriktionskoefficienten, μ , uppskattas till 0,015. Hjulens radie är 48,5 mm. Följande värden är hämtade från DC-motorernas specifikationsblad, [20].

$$T = 0,147 Nm \quad \text{Nominellt vridmoment}$$

$$P_n = 4,22 W \quad \text{Nominell effekt}$$

Med Newtons andra lag fås följande jämvikt:

$$F_{driv} - F_{friktion} = F_{res} = ma \quad (4)$$

Då DC-motorernas drivmoment och hjulens radie är kända kan den drivande kraften från DC-motorerna beräknas från formeln $T = Fr$. Eftersom roboten har två DC-motorer fås dubbelt så stort moment:

$$F_{driv} = \frac{T_{tot}}{r} = \frac{2 \cdot 0,147}{0,0485} = 6,06 N \quad (5)$$

Rullningsfriktionen kan beräknas som $F_{friktion} = F_N \mu$ där normalkraften $F_N = mg$

$$\begin{aligned} F_N &= mg = 10 \cdot 9,81 = 98,1 N \\ F_{friktion} &= F_N \mu = 98,1 \cdot 0,015 = 1,472 N \end{aligned} \quad (6)$$

Insättning av ekvation 5 och ekvation 6 i ekvation 4 ger den resulterande kraften.

$$F_{res} = F_{driv} - F_{friktion} = 4,59 N \quad (7)$$

DC-motorns nominella effekt är känd. En tumregel är att utnyttja hälften till två tredjedelar av den nominella effekten vid konstantdrift. Effekten som utnyttjas, P_u , blir om vi väljer hälften följande:

$$P_u = \frac{4,22}{2} = 2,11 W \quad (8)$$

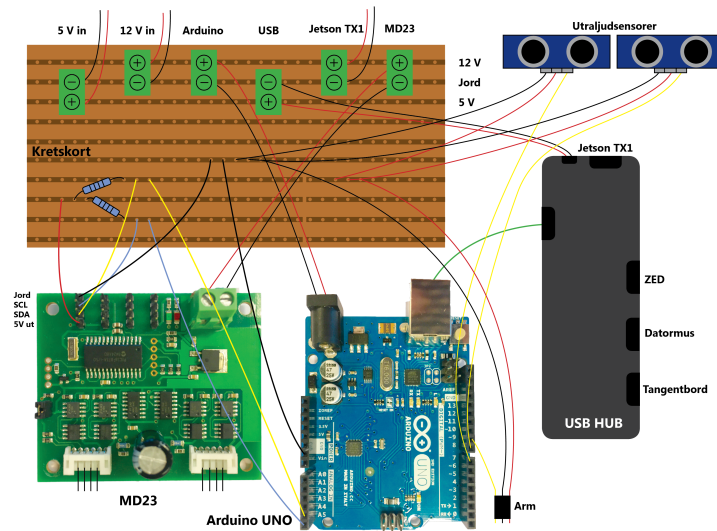
För att beräkna den vinkelhastighet, ω , roboten kan generera och därefter robotens hastighet, v , används formeln $P = T_{res} \omega$ där det resulterande momentet på varje hjul är $T_{res} = \frac{F_{res} r}{2} = 0,11 N$.

$$\begin{aligned} \omega &= \frac{P_u}{T_{res}} = \frac{2,11}{0,11} = 18,96 rad/s \\ v &= \omega r = 18,96 \cdot 0,0485 = 0,92 m/s \end{aligned} \quad (9)$$

Enligt beräkningar ska roboten under konstantdrift kunna hålla en hastighet på cirka 0,9 m/s. Även om det verkliga värdet kan variera något då förenklingar av systemet gjorts, bedöms hastigheten på 0,5 m/s kunna uppnås med god marginal.

4.4 Integration av komponenter

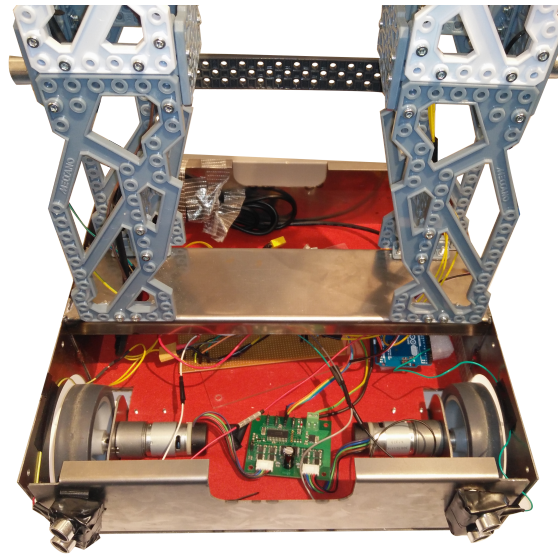
Figur 10 visar hur robotens komponenter är sammankopplade. Strömförsörjningskretsen levererar ström till Arduino, USB-hubben, Jetson TX1 och MD23-kontrollen. MD23-kontrollen är ihopkopplad med Arduino genom Arduinos A4- och A5-port, det vill säga portarna för I2C serial data (SDA) respektive I2C serial clock (SCL). Port A4 och A5 har kopplats till SDA- och SCL-porten på MD23. MD23:s port för 5 V-ut är sedan parallellkopplad med SDA- och SCL-kopplingen tillsammans med varsitt motstånd á 1800 Ω , så kallade pull up-resistorer, för att höja utsignalen. För förklaring av pull up-resistorer, se kapitel 2.3.3.



Figur 10: Översikt över koppling mellan systemets elektriska komponenter. Gul koppling representerar SDA och signalkabel, blå koppling representerar SCL, röd koppling representerar spänningskälla och svart koppling representerar jord. Den gröna kabeln från Arduino till USB-hubben är en USB-kabel.

Robotens arm kopplas in i Arduinos pin 9, 5 V-ut samt jord. De två ultraljudsensorerna kopplas in i Arduinos pin 7 respektive 8, 5 V-ut och jord. Alla komponenter i systemet har gemensam jord så att de hanterar signaler likvärdigt. Arduino är ansluten till USB-hubben för kommunikation med Jetson TX1.

För förbättrad struktur och för att minimera skador på utrustningen på grund av trassel samlas alla komponenter utom Jetson-datorn i bottenplattan. Detta bidrar även till ett förbättrat intryck av systemets estetik. Även DC-motorerna har skruvats fast på bottenplattan för att utnyttja den bockade konstruktionens böjstyvhet. Figur 11 visar hur bottenplattans inre ser ut med projektets slutgiltiga komponenter i. Bottenplattans insida har klätts in i tyg för att isolera elektroniken från stålkonstruktionen. Vissa komponenter har även monterats på plexiglasskivor för ytterligare isolering



Figur 11: Fotografi som visar bottenplattans insida med samtliga komponenter på plats. DC-motorerna är fastskruvade och botten har täckts av en matta för att undvika kortslutning på komponenter på grund av att el leds genom stålkonstruktionen.

5 Mjukvara

Ett av projektets mål är att få roboten att ta sig fram autonomt i en korridor och hälsa på mötande människor. För att möjliggöra detta är det många hårdvarukomponenter som ska samarbeta samt vara kompatibla med mjukvaran. Koden som har använts är skriven i C++ och Arduino. Delarna har sedan integrerats i ROS genom att abstrahera komponenterna till noder.

5.1 Ansiktsgenkänning

Ansiktsgenkänning används idag på många ställen - allt från brottsbekämpning till att markera vänner i bilder på sociala medier och filter i mobilkameror. Således bedrivs mycket forskning inom området och det finns ett flertal färdiga algoritmer som kan användas i egna implementationer. Några exempel är Principal Component Analysis (PCA), Local Gabor Binary Pattern Histogram Sequences (LGBPHS) och Haar Feature-based Cascade Classifiers. Projektet använder sig av den sistnämnda, härnäst refererat till som Haar-cascade. Inom bildbehandling kräver de flesta algoritmer en *feature*, vilket är en slags egenskap hos bilden som är nödvändig för att algoritmernas beräkningar ska fungera.

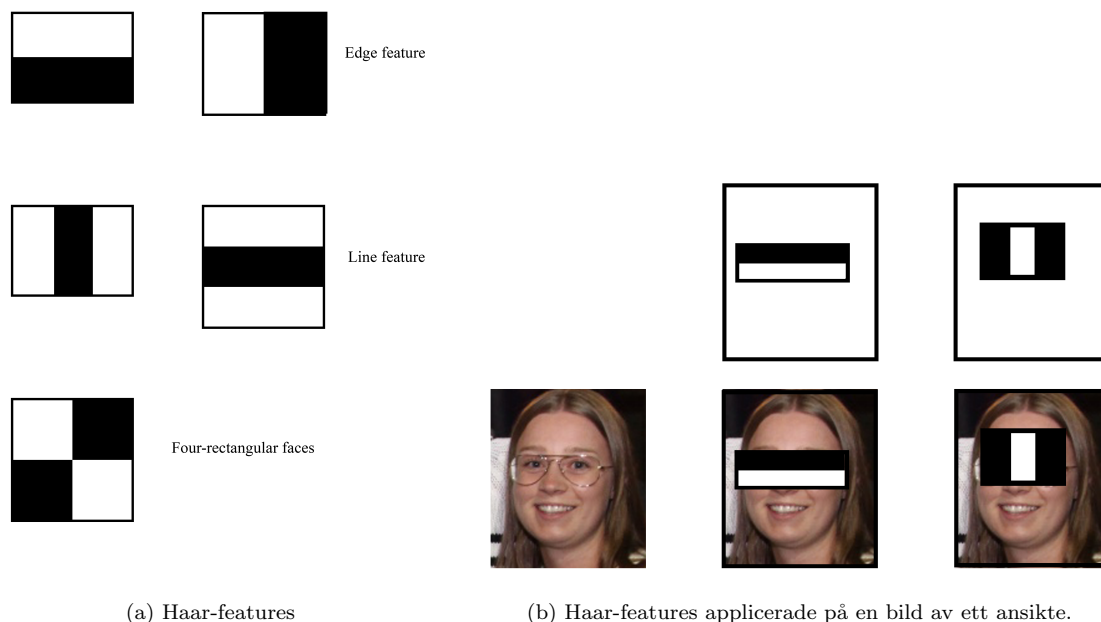
Viktigt att poängtera är att i det här projektet syftar ordet *igenkänning* i ansiktsgenkänning på att systemet identifierar objektet som kameran ser som **något** ansikte, inte att systemet känner igen och kopplar en identitet till det ansikte som registreras av kameran.

5.1.1 Haar-cascade-algoritmen

Haar-cascade är en algoritm för objektidentifiering, framtagen av Paul Viola och Michael Jones, [21]. Den är baserad på en kaskadfunktion som med hjälp av maskininlärning tränas på positiva och negativa bilder. Positiva bilder innehåller de objekt som ska upptäckas, medan de negativa fungerar tvärtom och innehåller annat än det som ska upptäckas. Algoritmen kan efter lyckad inlärning detektera objekten som den har tränats på i de positiva bilderna. Det objekt som utför objektidentifieringen kallas för en *klassificerare*.

Figur 12 visar hur algoritmen fungerar. En *Haar-feature* är en rektangel indelat i vita och svarta fält. För varje fält beräknas pixelintensiteten, vilket är ett mått på hur ljus en pixel är, och om bilden är i färg konverteras den först till gråskala. Differensen mellanfälten används sedan för att kategorisera områdena i bilden och upptäcka skillnader. Ett relevant exempel är mänskliga ansikten, där ögonpartiet oftast är mörkare än kinderna. För varje bild klassificeraren tränas på appliceras en mängd features. De features som är sämre på att urskilja de önskade objekten förkastas och de med minimal felfrekvens behålls.

För att slutföra inlärningsprocessen skapas en slutgiltig klassificerare, som är en viktad summa av flertalet svaga klassificerare. Att de kallas svaga beror på att de på egen hand inte kan användas för att klassificera en bild, utan behöver kombineras med andra. Vid 200 features kan en nittiofemprocentig klassificeringssäkerhet uppnås.



Figur 12: Illustration av hur Haar-features kan användas för att upptäcka olika delar av ett ansikte.

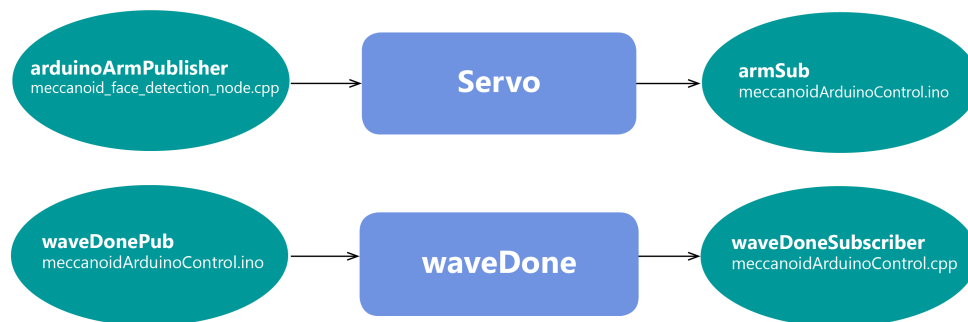
En fördel med Haar-cascade är att beräkningarna är snabba och effektiva. Som nämnt i [22] använder algoritmen tekniken *summed area table*, även kallat *integral image* i bildbehandlingssammanhang. Förenklat innebär det att istället för att beräkna värdet på varje pixel i bilden, räcker det med fyra hörnvärden på det området som Haar-features appliceras på. Dessutom används *kaskadklassificering*, där olika features grupperas och appliceras på bilden en efter en. Om det upptäcks att ett område på en bild omöjligen kan innehålla det som söks - i detta fall ett ansikte - struntar algoritmen fortsättningsvis i det området. En nackdel är dock som nämnt i [23] att algoritmen måste tränas relativt länge innan den med god säkerhet kan avgöra om en bild innehåller det önskade objektet eller inte.

5.1.2 Praktisk implementation

Ansiktsgenkänning med Haar-cascade finns som en standardfunktion i OpenCV. Som nämnt i kapitel 2.2 har det tidigare implementerats en funktion för ansiktsgenkänning. Då Kinect-kameran har ersatts av en ZED-kamera har det införts nya subscribers och publishers i ROS för att kommunikationen mellan kamera och ROS ska fungera. En ytterligare publisher som skickar ut meddelanden till arm-noden när kameran detekterar ett ansikte har lagts till för att roboten ska kunna vinka.

5.2 Topics och meddelanden i ROS

I projektet har det satts upp ett antal olika subscribers och publishers som kommunicerar med varandra över två topics - `/servo` och `/waveDone`. Sambandet mellan topics, publishers och subscribers i projektet illustreras i Figur 13. Subscribern `armSub` lyssnar på meddelanden av datatypen `Int` över `/servo`, medan `waveDoneSub` prenumererar på `Empty`-meddelanden som publiceras på `/waveDone`. Således är topics:en en väsentlig del av systemets kommunikation.



Figur 13: Översikt av robotens topics samt tillhörande publishers och subscribers.

5.2.1 `/servo`

Topic:en `/servo` sköter all kommunikation med servomotorerna i robotens armar och DC-motorerna i robotens fötter. Eftersom roboten styrs med en tillståndsvariabel har typen för `/servo` valts till ett heltal (8-bit integer, `uint8`). Det är ett naturligt val, då det blir enkelt att kontrollera vad roboten gör genom att skicka ett specifikt tal över topic:en. Via `/servo` kan publishern `arduinoArmPublisher` publicera meddelanden som finns tillgängliga för alla `Int`-subscribers att lyssna på.

När ett ansikte identifieras av kameran skickas ett meddelande innehållande heltalet 1 över `/servo`. Servomotorerna i robotens arm, som är en subscriber på `/servo`, kommer därmed vinka när meddelandet tas emot. DC-motorerna i robotens fötter är också en subscriber på `/servo`, vilket betyder att även de lyssnar på vad ansiktsigenkänningen publicerar för meddelanden och stannar när ett ansikte upptäcks.

5.2.2 `/waveDone`

Eftersom en vinkningsrutin tar relativt lång tid att utföra och kameran letar efter ansikten kontinuerligt hinner kameran registrera flera bilder av samma ansikte innan roboten vinkat klart. De ansiktsbilder som kameran registrerar när roboten är upptagen med att vinka lägger sig på kö, vilket leder till att roboten vinkar igen så fort första vinkningen är klar. Syftet med topic:en `/waveDone` är att informera noden som sköter ansiktsigenkänningen när roboten är redo för vinkning och på så sätt förhindra att kameran kör ansikten.

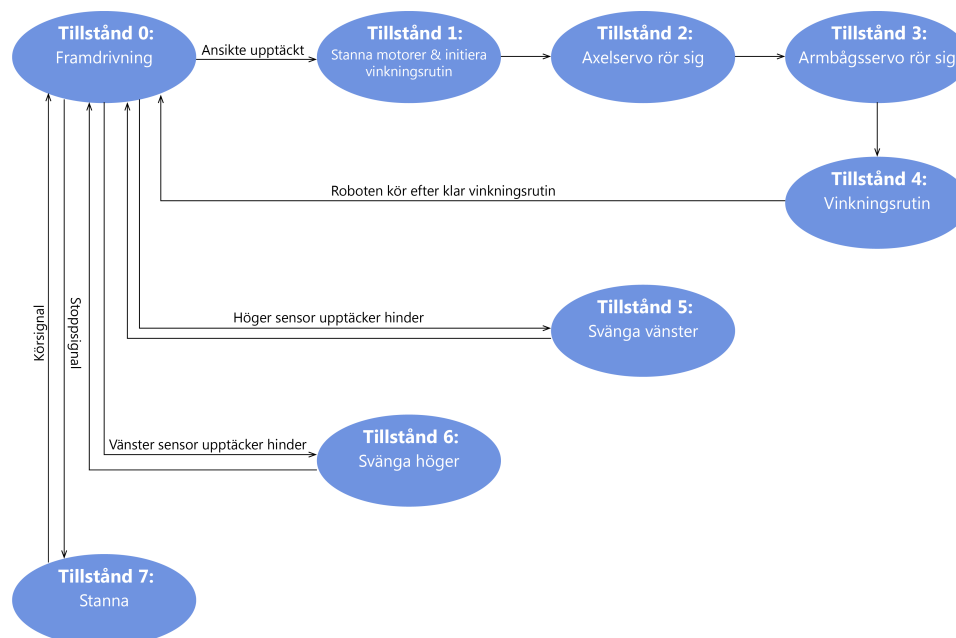
För att bestämma när roboten är redo för att vinka har en variabel av datatypen `bool`, `readyForAFace`, lagts till i koden för ansiktsigenkänningen. Variabeln sätts till `False` när ett ansikte identifieras och ett meddelande skickas till `/servo`, vilket initierar vinkningsrutinen. När vinkningsrutinen startat pausas ansiktsigenkänningen temporärt. Efter att vinkningsrutinen är utförd skickas ett tomt meddelande (`Empty`) över `/waveDone` och en callback-funktion sätter variabeln `readyForAFace` till `True`, vilket startar ansiktsigenkänningen igen.

5.3 Robotens tillståndsmaskin

Kontrollen av roboten bygger på en ändlig tillståndsmaskin. En tillståndsvariabel bestämmer vilket tillstånd roboten kan befinner sig i. Ett tillstånd kan exempelvis vara att roboten är stillastående eller att den kör framåt. Olika tillstånd måste vara unika för att de ska kunna särskiljas. Vanligtvis krävs

det även vissa förutsättningar eller villkor för att roboten ska byta tillstånd, där ett villkor kan vara att roboten endast får gå framåt efter att vinkningsrutinen är klar.

I det här projektet används heltal av datatypen `int` för identifikation av olika tillstånd. I Figur 14 visas en översikt över robotens olika tillstånd. Grundtillståndet är 7, där roboten är stillastående och skannar efter ansikten. Om ett ansikte upptäcks vinkar roboten och sätter därefter tillståndet till 0. Detta kan även göras manuellt av användaren genom att publicera ett meddelande via ROS från en kommandoterminal. När roboten befinner sig i tillstånd 0 åker den framåt och använder sig av ultraljudssensorerna för att manövrera. Om ett ansikte upptäcks publicerar ansiktsgigenkänningsnoden ett meddelande som innehåller 1 till topic:en `servo`. En callback-funktion ändrar därefter robotens tillstånd till 1. Detta stoppar framdrivningen samt initierar vinkningsrutinen. Vinkningsrutinen består i sin tur av tre olika tillstånd - upphöjning av axelservomotorn, rotation av överarmsservomotorn, samt vinkningsrörelsen med underarmen - som ändrar tillståndet till 2, 3 respektive 4. Efter en vinkning återställs tillståndet till 0, vilket startar framdrivningen igen.

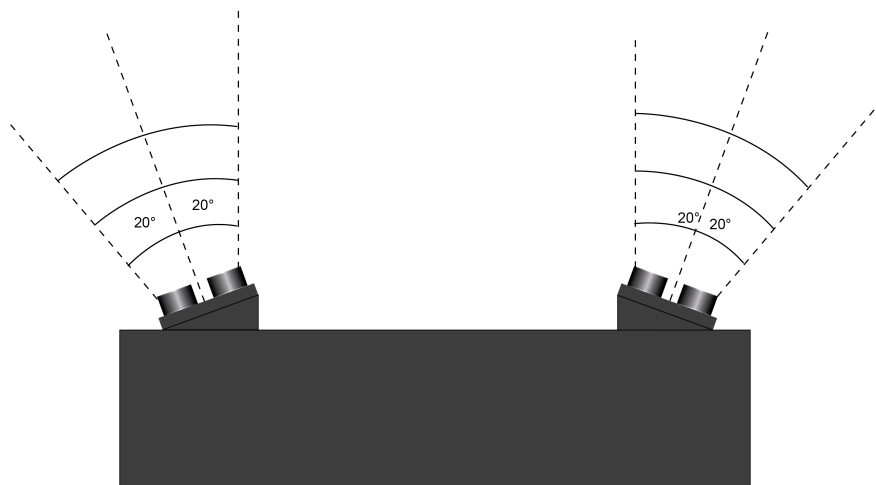


Figur 14: Översikt på de tillstånd som finns i robotens tillståndsmaskin.

5.4 Implementation av ultraljudssensorer

För att undvika att roboten kör in i väggar och andra hinder används en algoritm som bygger på inlästa avståndsvärden från två ultraljudssensorer. Det uppmätta avståndet från vardera sensor till närmsta objekt sparas i variabler och om något avstånd understiger 50 cm stoppas framdrivningen och roboten börjar svänga. Två avståndsvariabler per sensor används för att motverka att enstaka inkorrekta värden rapporteras vid de fall som ljudvågor reflekteras felaktigt, det vill säga på ett sätt där de inte återvänder raka vägen till sensorn. Roboten har även en backningsrutin implementerad. Om båda sensorerna befinner sig inom 10 centimeter från ett objekt backar roboten en kort sträcka och sätter därefter igång framdrivningen igen.

Om den högra sensorn upptäcker ett hinder svänger roboten till vänster och vice versa. Svängningen fortsätter tills det inte längre är någonting framför sensorn. Således kommer roboten att svänga 90° om den stöter på en vägg och vända helt om den åker in i ett hörn. I de fall då båda sensorerna upptäcker hinder inom 50 cm svänger roboten ifrån det hinder som ligger närmast. Sensorerna är monterade på bottenplattans framsida vilket illustreras i Figur 15. De är symmetriskt placerade cirka 25 cm från varandra och är vinklade 20° från robotens mitt.



Figur 15: Sensorernas placering på bottenplattan sett ovanifrån. Sensorerna registrerar objekt som är inom en vinkel av $0-40^\circ$ framifrån.

6 Verifiering av robotens funktionalitet

Målet med projektet var att modifiera roboten så att den kan ta sig fram i en korridor i dubbel hastigheten jämfört med före projektet, undvika hinder samt identifiera människor och hälsa på dem. För att utvärdera huruvida målen har uppfyllts eller ej har de olika funktionerna testats var för sig och slutligen tillsammans. Tabell 4 visar en översikt på vilka tester som genomfördes, kort beskrivning av testet, vilka krav som ställs för att testet ska vara godkänt samt resultatet på testet.

Tabell 4: Sammanfattning av testfall som har utförts för att verifiera att projektet har uppnått målen.

Funktion	Mål	Testmetod	Krav	Resultat
Hastighet	Roboten ska bibehålla minst 0,5 m/s i minst 10 s	Klocka tiden det tar för roboten att köra en sträcka på 10 m och därefter beräkna medelhastigheten	80% godkända tester efter minst 10 tester	Godkänt (se Tabell 5)
Ansikts-identifiering och hälsningsfunktion	Roboten ska genomföra en fysisk hälsning efter att ha identifierat ett ansikte inom kameran synfält max två meter bort	Roboten exponeras för tio ansikten från olika håll	Ansikts-identifiering och hälsningsfunktion är korrekt i minst 80% av fallen	Godkänt
Identifiering av hinder	Roboten ska med hjälp av ultraljudssensorer kunna identifiera framförvarande hinder	Ett sfäriskt objekt med 16 cm i diameter inkommer rakt framifrån i höjd med sensorerna	Identifiering innan 50 cm i minst 80% av fallen efter minst 10 tester	Ej godkänt
		Ett sfäriskt objekt med 16 cm i diameter inkommer 30° i sidled åt höger i höjd med sensorerna	Identifiering innan 50 cm i minst 80% av fallen efter minst 10 tester	Godkänt
		Ett sfäriskt objekt med 16 cm i diameter inkommer 30° i sidled åt vänster i höjd med sensorerna	Identifiering innan 50 cm i minst 80% av fallen efter minst 10 tester	Godkänt
Undvika hinder	Roboten ska identifiera och sedan undvika att köra in i stationära hinder.	Roboten kör genom en rak korridor med fem olika hinder	Undviker hinder i minst 80% av fallen	Godkänt
Kombinerad framåtdrift, hinderundvikning och hälsningsfunktion	Roboten ska undvika hinder och hälsa på människor vid framåtdrift	Roboten kör genom en rak korridor med hinder och mötande människor	Undviker hinder och hälsar på mötande människor i minst 80% av fallen	Ej godkänt

6.1 Hastighetstest

Målet med hastighetstestet var att se om roboten kan uppnå en hastighet på 0.5 m/s. En bana med start- och mållinje markerades i en korridor. Därefter startades DC-motorerna och roboten gavs en

startsträcka på ungefär en meter för att hinna accelerera till topphastighet innan mätsträckans startlinje. Tiden det tog för roboten att köra sträckan uppmättes. Testet utfördes tio gånger för att säkerställa resultatet. Med hjälp av den uppmätta datan kunde robotens medelhastighet för varje försök beräknas.

Resultatet från testet återfinns i Tabell 5 där uppmätt tid samt den beräknade hastigheten presenteras för varje testförsök. Tabellen visar även den genomsnittliga hastigheten för försöken. Enligt testet kunde roboten uppnå en medelhastighet på 0,57 m/s.

Tabell 5: Resultat från robotens hastighetstest. Tabellen redovisar tid och hastighet för varje försök samt ett beräknat genomsnittresultat.

Försök	Tid (s)	Hastighet (m/s)
1	18,81	0,53
2	17,40	0,57
3	17,56	0,57
4	17,41	0,57
5	17,33	0,58
6	17,49	0,57
7	17,51	0,57
8	17,24	0,57
9	17,37	0,58
10	17,33	0,58
Genomsnitt	17,55	0,57

6.2 Ansiktsidentifiering- och hälsningsfunktionstest

För verifikation av robotens förmåga att hälsa på människor vid rätt tillfällen designades ett test för att undersöka hur väl den känner igen ansikten. Ansiktsigenkänningsnoden initierades och roboten riktades mot en neutral bakgrund. Vid tio tillfällen visades ett ansikte för kameran. Eftersom roboten är programmerad till att vinka när ett ansikte identifieras användes den funktionen som indikation för att ansiktsigenkänningen fungerade som tänkt.

Under testet vinkade roboten i tio fall av tio när kameran exponerades för ett ansikte. Testet kan därmed bedömas som godkänt.

6.3 Hinderidentifieringstest

Ett test för att bedöma hur väl sensorerna identifierar objekt konstruerades. Syftet med testet var att avgöra från vilka vinklar som objekt kan närma sig och upptäckas av sensorerna på. En skumplastboll med diameter 16 cm fördes mot roboten från ett avstånd på ungefär två meter för att simulera ett hinder som roboten kör mot. Ultraljudssensorerna aktiverades och med hjälp av funktionen *Serial Monitor* i Arduinomjukvaran presenterades de beräknade avstånden till objektet i realtid. Båda sensorerna testades samtidigt och var placerade som illustrerat i Figur 15 under kapitel 5.4.

Under testets gång blev det uppenbart att sensorerna inte klarar av att upptäcka föremål som närmar sig roboten rakt framifrån ifall de ryms inom den blinda zon som finns mellan de två ultraljudssensorerna. Däremot registrerades objekt som inkom upp till 40° i sidled, vilket innebär att roboten har god förmåga att upptäcka och undvika hinder även från sidan.

6.4 Hinderundvikningstest

Testet genomfördes genom att hinder i form av pappkartonger placerades i en korridor. Kartongerna var av varierande storlek men inte mindre än 30 cm $\times$ 30 cm så att robotens ultraljudssensorer kunde uppfatta dem även om de befann sig rakt framför sensorerna. Målet med testet var att verifiera robotens förmåga att ta sig fram i en korridor från punkt A till punkt B utan att kollidera med hindren. Det ställdes höga krav på att mjukvaran fungerade korrekt, alltså att rätt funktioner triggas av olika händelser under testet.

Roboten klarade för det mesta av att köra i korridoren utan att kollidera med hindren eller väggarna. Däremot lyckades inte roboten ta sig fram till andra ändan av korridoren och därmed passera samtliga hinder. Istället vände roboten tillbaka eller navigerade runt i cirklar mellan hindren. Det uppstod också mindre problem vid de tillfällen då roboten kom för nära hindren och därmed förflyttade pappkartongerna från deras originalplacering i samband med undvikningen.

6.5 Kombinerat framdrifts-, hinderundviknings- och hälsningstest

För att undersöka hur väl roboten besvarar de fyra frågeställningarna utfördes ett slutgiltigt test för att se om roboten klarar av att ta sig från ena sidan av korridoren till en annan medan som ansiktsigenkänningsnoden är aktiverad. Testet genomfördes genom att hinder placerades i en korridor samtidigt som mötande människor passerade. Roboten fick sedan försöka ta sig fram i korridoren utan att kollidera med hindren och hälsa på de mötande människorna genom att vinka. Syftet med testet var att säkerställa att kommunikationen mellan de olika komponenterna fungerade som planerat.

Under testet aktiverades ansiktsidentifieringen felaktigt när ingen människa vistades inom robotens synfält vid flera tillfällen. Roboten kunde ta sig förbi hinder då sensorerna fortfarande fungerade som planerat, men eftersom att hälsningsfunktionen har prioritet i koden utfördes denna istället för att framåtdriften fortsattes. Resultatet blev att roboten mestadels stod still i korridoren och vinkade. Likt det resultat som presenteras i hinderundvikningstestet kunde roboten inte ta sig från punkt A till B i korridoren.

7 Diskussion

Tre av de fyra frågeställningar som presenterades inledningsvis har besvarats med hjälp av de tester som har genomförts. En metod för att särskilja människor och övriga objekt har presenterats och testats framgångsrikt. Roboten klarar i dagsläget inte av att ta sig fram genom en korridor autonomt och i detta kapitel diskuteras denna problematik tillsammans med resultat från de genomförda testerna.

7.1 Utvärdering av robotens framdrivningshastighet

En av projektets frågeställningar var att undersöka hur roboten ska modifieras för att dess framdrivningshastighet ska nå 0,5 m/s. I testerna som genomfördes för att verifiera detta uppmättes en medelhastighet hos roboten på 0,57 m/s och är därmed över den som var önskad. Resultatet visar att de nya DC-motorerna EMG30 är tillräckligt starka för sin uppgift.

Enligt beräkningarna skulle DC-motorerna kunna driva roboten med en hastighet på upp till 0,9 m/s om de körs på full fart, alltså 0,33 m/s snabbare än vad testerna visade. Det är dock ej önskvärt att framföra roboten i maxhastighet då det leder till instabilitet och en hög batterikonsumtion. Skillnaden mellan det uppmätta värdet och den teoretiska maxhastigheten är alltså ett aktivt val.

7.2 Utvärdering av hinderidentifiering och hinderundvikning

Testet med att upptäcka hinder blev ej godkänt för de fall då ett objekt kommer rakt framifrån. Detta beror på att det finns en blind fläck i ett fält cirka 25 cm mellan de två ultraljudssensorerna, se Figur 15 i kapitel 5.4. Blindfläcken medför att roboten inte ser små objekt som är rakt framför den. Dock har de flesta objekt som roboten förväntas passera en större bredd än 25 cm, vilket gör att den blinda fläcken inte är ett stort problem. Testet där ett objekt infördes från sidan stämde bra överens med de teoretiska beräkningarna att sensorerna kan upptäcka objekt från sidan i vinklar upp mot 40°. Genom implementation av fler sensorer hade ett bättre testresultat kunnat uppnås.

De två ultraljudssensorer som installerades under projektets gång klarar enligt hinderidentifieringstestet av att identifiera hinder snett framifrån. I dagsläget missar dock roboten ofta att upptäcka väggar när den närmar sig med liten vinkel mellan sensor och vägg. Detta beror på att ultraljudet som skickas ut studsar bort från sensorn om vinkeln mellan objektets yta och sensorn är mindre än 45°, se Figur 5 i kapitel 2.3.2.

Ett annat problem som kan uppstå är att ljudvågorna som emitteras kan reflekteras i felaktig riktning vilket medför att ljudvågorna studsar mellan objekt innan de återvänder till sensorn. Resultatet blir en missvisande avståndsbedömning och roboten börjar svänga trots att det inte finns något hinder. Detta problem åtgärdades genom att använda två avståndsvariabler per sensor. Avståndet mäts och sparas i den första variabeln och efter en kort fördröjning görs ytterligare en mätning till variabel två. Roboten börjar då endast svänga i de fall där båda variablerna är under gränsvärdet på 50 cm.

Då roboten i nuläget inte är medveten om var den befinner sig är framdrivningen stokastisk. När ett hinder påträffas svänger roboten åt ett lämpligt håll, men det finns ingen garanti för att den därefter fortsätter i ursprungsriktningen. Ett förekommande exempel är då roboten ska undvika väggar gör den ibland en för kraftig sväng och åker därefter mot motsatt vägg, vilket resulterar i ett sicksackliknande mönster. För att motverka detta kan man komplettera sensorerna med ett positioneringssystem och då implementera en mer avancerad algoritm som förutom avstånd till hinder även tar hänsyn till önskad riktning.

7.3 Utvärdering av ansiktsigenkänning

Utifrån resultat från testfallen bedöms roboten kunna känna igen ansikten. Lösningen är dock inte optimal - kameran registrerar ibland ansikten felaktigt trots att ingen människa är närvarande i robotens synfält. Den mest naturliga förklaringen är att algoritmen för ansiktsigenkänning inte är tillräckligt specifik. Till exempel kan vissa föremål innehållande två prickar och ett streck tolkas som ett ansikte.

Ansiktsigenkänning är viktig för att besvara projektets frågeställning hur roboten kan interagera med människor som den möter. När roboten upptäcker ett ansikte stannar den för att vinka och roboten har därmed interagerat med en människa. Med ansiktsigenkänningen kan roboten dessutom anpassa dess respons beroende på om den möter ett objekt eller en människa. När ett hinder upptäcks undviks det, men när en människa upptäcks stannar roboten och hälsar.

7.4 Utvärdering av robotens sammankopplade system

Överlag fungerade en Meccanoid robot bra för att uppnå projektmålen. Robotens geometri gjorde det enkelt att montera komponenter som kamera och dator på dess befintliga delar. Då servomotorerna ansluts genom jord, 5 V och signalkabel var det även enkelt att flytta styrningen av dessa från Meccabrain:en till Jetson TX1 via en Arduino UNO. Likaså kunde även bottenplattan integreras som en del av systemet på ett smidigt sätt.

Genom de avtagbara locken har bottenplattan konstruerats så att komponenter enkelt kan läggas till eller tas bort i framtiden. Bottenplattan är tillverkad i stålplåt vilket inte är optimalt ur elektrisk synpunkt eftersom det är ett ledande material. Inledningsvis skapade metallen problem för de elektroniska kretsar som var placerade inuti. Problemet löstes genom att mikrokontrollerna och kretskortet monterades på plexiglasskivor och att bottenplattans insida täcktes med matta, vilken isolerar elektroniken från stålplåten. Mattan motverkar dessutom buller som uppstår när komponenterna i bottenplattan vibrerar vid körning.

För att roboten ska kunna drivas i en korridor utan att vara beroende av en fast strömkälla kommer all strömförsörjning numera från batterier placerade i bottenplattan. Den teoretiska drifttiden för batterierna beräknades till 2,40 h för 11,1 V-kretsen och 1,89 h för 5 V-kretsen vilket stämmer bra överens med den faktiska batteritiden. Med den beräknade drifttiden kan roboten utan problem köra i korridoren tillräckligt länge för att uppfylla projektmålen.

Generellt har ROS fungerat bra för att nå projektets mål då ett flertal processer i form av noder kan styras samtidigt. Med ROS har det också varit smidigt att implementera nya eller ta bort valfritt antal noder, något som hade varit krångligare om enbart Arduino-kod använts. Kontrollen över alla komponenter i systemet och således även användarvänligheten har ökat. Dessutom har alla noder kunnat integreras till ett överskådligt system trots att olika programmeringsspråk använts.

ROS har orsakat problem vid de tillfällen då synkronisering med en hårdvarukomponent misslyckats eller när en funktion tagit lång tid att utföra. När ett kommando tar lång tid att utföra hinner servomotorer eller annan hårdvara inte slutföra sin uppgift innan nästa kommando anropas. Ett exempel på kod som tar lång tid att köra är vinkningsrutinen. Inledningsvis var hela vinkningsrutinen ihopsatt till en funktion vilken tog lång tid att utföra och ledde därmed till ovan nämnda problem. Informationen Arduinon skickade till datorn inte hann behandlas i tid när vinkningsfunktionen exekverades vilket gjorde att `spinOnce` inte anropades. Därför delades vinkningsrutinen upp i mindre delfunktioner med hjälp av en tillståndsmaskin, där varje funktion utfördes när robotens tillstånd ändrats till ett specifikt värde. Mellan varje tillståndsåndring anropades `spinOnce`, så att eventuell data som väntade på att behandlas togs om hand innan nästa funktion utfördes.

8 Summering och framtida rekommendationer

För att öka robotens hastighet har de ursprungliga framdrivningsmotorerna på roboten ersatts av ett par 12 V DC-motorer. Trots att roboten med den nya bottenplattan har blivit tyngre är DC-motorerna tillräckligt kraftfulla för att driva hela systemet framåt. Med en hastighet på 0,57 m/s kan roboten nu ta sig fram dubbelt så snabbt som den kunde vid projektets start.

Roboten undviker kollision med hjälp av två sensorer som är placerade på bottenplattans framsida, vilka varnar när hinder är i vägen. Jetson TX1 behandlar informationen och skickar signaler till DC-motorerna som svänger åt lämpligt håll. För att skilja på objekt och människa används ansiktsgenkänningsnoden. Om ett ansikte detekteras är hindret en människa och annars ett objekt.

ZED-kameran på huvudet läser av omgivningen kontinuerligt och identifierar ansikten ifall de befinner sig inom dess synfält. När ett ansikte detekteras stannar roboten och utför en hälsningsrörelse i form av en vinkning. Vinkningen är robotens sätt att interagera med de människor den möter.

Roboten kan ta sig fram i en korridor utan att kollidera med människor eller objekt. Under testerna som genomfördes framkom det dock att roboten, med dess nuvarande modifikationer, inte klarar av att ta sig från ena ändan av en korridor till den andra. En möjlig lösning på problemet är att utrusta roboten med ett positioneringssystem vilket hade besvarat sista frågeställningen fullständigt. Genom ett positioneringssystem vet roboten var i rummet den befinner sig och kan med den informationen förflytta sig autonomt på ett lämpligt sätt.

Det finns stora möjligheter till vidare utveckling av projektet där många nya funktioner kan implementeras. Syftet med arbetet har inte varit att konstruera en slutgiltig produkt, utan snarare att lägga grunden för framtida förbättringar. Ett exempel på en användbar funktion är styrning via en mobilapplikation. Nyttiga delfunktioner i appen kan exempelvis vara styrning, start- och stoppknapp för autonom framdrivning eller uppvisning av olika mätvärden från roboten.

Ett sätt att förbättra manövreringsförmågan är att implementera ett gyroskop, exempelvis Yocto-3D [24]. Användning av gyroskop förbättrar robotens förmåga att bibehålla en rak riktning då det i nuläget inte finns något som förhindrar roboten att börja åka sicksack efter att ett hinder upptäckts och undvikits. I kombination med fler ultraljudssensorer för att vidga robotens synfält borde flera av de brister som den nuvarande navigationen och objektidentifieringen har kunna åtgärdas. Två sensorer är i nuläget monterade på robotens vänster- och högersida, men de är inte inkopplade. En förutsättning för utökandet av antalet komponenter är att datorn kan hantera det ökade informationsflödet över USB-bussen och det kan krävas ytterligare en Arduino eller mikrokontroller för att hantera detta.

Referenser

- [1] "Vartannat jobb automatiseras inom 20 år - utmaningar för Sverige", Stiftelsen för Strategisk Forskning, 1st edition, 2014.
- [2] "Transportrobotar kör gods på Nya Karolinska Solna", *Robotnyheter.se*, 2016. [Online]. Tillgänglig: <http://robotnyheter.se/2016/04/27/transportrobotar-kor-gods-pa-nya-karolinska-solna/#more-32883>. Hämtad: 20 februari 2017.
- [3] "Meccanoid G15KS Personal Robot" *meccano.com*, 2017. [Online]. Tillgänglig: <http://meccano.com/product/p10945/meccanoid-g15ks-personal-robot>. Hämtad 7 mars 2017.
- [4] H. Lindén, N. S. Kurien, F. Bergström, "ROS Framework for an Autonomous Robot with Visual Guidance System", Chalmers University of Technology, Gothenburg, 2016.
- [5] "Arduino UNO & Genuino UNO", *Arduino.cc*, 2017. [Online]. Tillgänglig: <https://www.Arduino.cc/en/main/ArduinoBoardUno>. Hämtad: 7 mars 2017.
- [6] "ROS.org — About ROS", *Ros.org*, 2017. [Online]. Tillgänglig: <http://www.ros.org/about-ros/>. Hämtad: 9 mars 2017.
- [7] "JETSON TX1 DEVELOPER KIT" *nvvidia.co.uk*, 2017. [Online]. Tillgänglig: <http://www.nvidia.co.uk/object/jetson-tx1-dev-kit-se.html>. Hämtad: 9 mars 2017
- [8] "ABOUT — OpenCV", *Opencv.org*, 2017. [Online]. Tillgänglig: <http://opencv.org/about.html>. Hämtad: 7 mars 2017.
- [9] "ZED - Depth Sensing and Camera Tracking", *Stereolabs.com*, 2017. [Online]. Tillgänglig: <https://www.stereolabs.com/zed/specs/>. Hämtad: 9 Mars 2017.
- [10] A. Brehmer, "zed-ros-wrapper - ROS Wiki", *Wiki.ros.org*, 2017. [Online]. Tillgänglig: <http://wiki.ros.org/zed-ros-wrapper>. Hämtad: 9 Mars 2017.
- [11] "PING))) Ultrasonic Distance Sensor", *Parallax Inc.*, 2017. [Online]. Tillgänglig: <https://www.parallax.com/product/28015>. Hämtad: 25 april 201
- [12] Parallax Inc., "PING))) Ultrasonic Distance Sensor (#28015)", 2nd ed. 2013.
- [13] "EMG30, mounting bracket and wheel specification", *Robot-Electronics.co.uk*, 2017. [Online]. Tillgänglig: <https://www.robot-electronics.co.uk/htm/emg30.htm>. Hämtad: 9 Mars 2017.
- [14] Nicola, "MD23 - Dual 12Volt 3Amp H Bridge Motor Drive", *robotshop.com*, 2008. [Online]. Tillgänglig: <http://robotshop.com/letsmakerobots/md23-dual-12volt-3amp-h-bridge-motor-drive>. Hämtad: 7 mars 2017.
- [15] "MD25 - Dual 12Volt 2.8Amp H Bridge Motor Drive", *Robot-Electronics.co.uk*, 2017. [Online]. Tillgänglig: <http://www.robot-electronics.co.uk/htm/md25tech.htm>. Hämtad: 3 mars 2017.
- [16] "Using the I2C Buss", *Robot-Electronics.co.uk*, 2017. [Online]. Tillgänglig: <http://www.robot-electronics.co.uk/i2c-tutorial>. Hämtad: 3 mars 2017.
- [17] Brian Schneider. "A Guide to Understanding LiPo Batteries" *rogershobbycenter.com*, 2017. [Online]. Tillgänglig: <https://rogershobbycenter.com/lipoguide/>. Hämtad: 2 maj 2017
- [18] Kjell & Company. "Luxorparts Variabel spänningsregulator Switchad" *kjell.com*, 2017. [Online]. Tillgänglig: <https://www.kjell.com/se/sortiment/el-verktyg/elektronik/Arduino/moduler/luxorparts-variabel-spänningsregulator-switchad-p87896>. Hämtad: 25 april 2017.

- [19] Switching Regulator IC Series: , Utgivningsort, Land: Utgivare, år. [Online]. Tillgänglig: URL. Hämtad: datum. Rohm Semiconductors. Efternamn, "Hemsidans titel, Åbbplatsens namn, år. [Online]. Tillgänglig: URL. Hämtad: datum.
- [20] EMG30: mounting bracket and wheel specification: Robot Electronics, 2017. [Online]. Tillgänglig: <https://www.robot-electronics.co.uk/htm/emg30.htm>. Hämtad: 11 maj 2017.
- [21] Face Detection using Haar cascades, Tillgänglig: http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html Hämtad: 5 April 2017
- [22] "Computer Vision – The Integral Image"
Tillgänglig: <https://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image/> Publicerad: 3 september 2010 Hämtad: 5 april 2017
- [23] P. Viola, M. Jones "Rapid Object Detection Using A Boosted Cascade of Simple Features", Mitsubishi Electric Research Labs, Cambridge, USA, 2001.
Tillgänglig: http://wearables.cc.gatech.edu/paper_of_week/viola01rapid.pdf. Hämtad: 3 april 2017
- [24] Yoctopuce. "Yocto-3D" <http://www.yoctopuce.com>, 2017. [Online]. Tillgänglig: <http://www.yoctopuce.com/EN/products/usb-position-sensors/yocto-3d> Hämtad: 9 maj 2017.

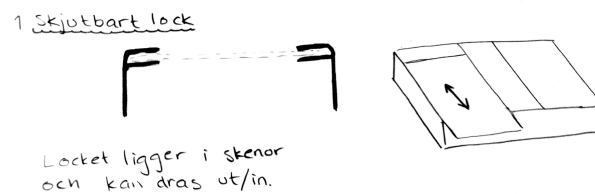
A Konceptgenerering och eliminering för bottenplattans utformning.

A.1 Koncept

Nedan presenteras de koncept som togs fram under konceptgenereringen. För varje koncept presenteras en bakomliggande idé och illustreras med en skiss.

Koncept 1 - Skjutbart lock

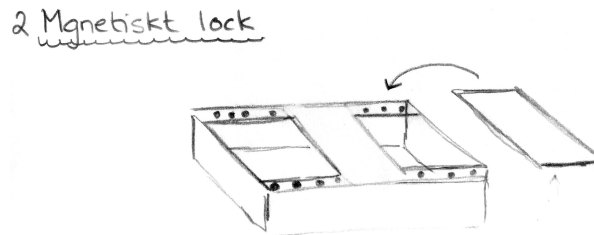
Koncept 1 som visas i Figur 16 bygger på en lösning där locket ska skjutas in igenom ett spår i bottenplattan. Detta spår skapas genom att först bocka kanten på lådan och därefter svetsa fast en list under som ska fungera som spårans underdel.



Figur 16: Skiss över koncept 1

Koncept 2 - Magnetiskt lock

Koncept 2 som visas i Figur 17 löser problemet med tillgänglighet av innehållet i bottenplattan genom ett lock som sitter fast med hjälp av magneter. Konstruktion kan tillverkas genom att bocka ner de yttre kanterna på lådan och därefter montera fast magneter på den blockade plåten.

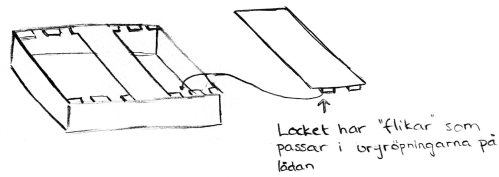


Figur 17: Skiss över koncept 2

Koncept 3 - Lock med pusselbitsfunktion

Konceptets lock är försett med tänder som passar i motsvarande hål på underdelen. Listerna med hål i fästes i underdelen, antingen med svetsning eller med förband. Tänderna som sitter fast i locket förhindrar rörelse i planets riktning, medan lockets egenvikt håller det på plats. I Figur 18 illustreras hur tänkt konceptet kan se ut.

3 Lock med pusselbitsfunktion

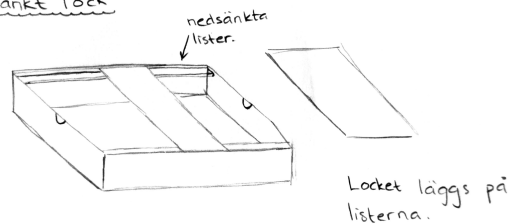


Figur 18: Skiss över koncept 3.

Koncept 4 - Nedsänkt lock

Detta koncept bygger på att lister sitter fastsvetsade längs lådans långsidor vilket visas i Figur 19. Dessa två lister ska sedan bära upp locket som hålls fast vertikalt med hjälp av gravitationen och horisontellt av lådans och mittsektionens sidor.

4 Nedsänkt lock

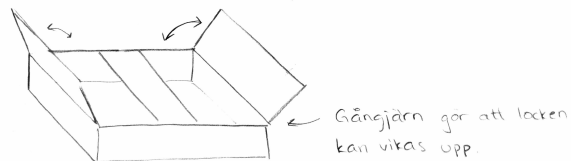


Figur 19: Skiss över koncept 4.

Koncept 5 - Gångjärn

Koncept 5 har två lock som sitter fast med hjälp av två gångjärn var. Se Figur 20.

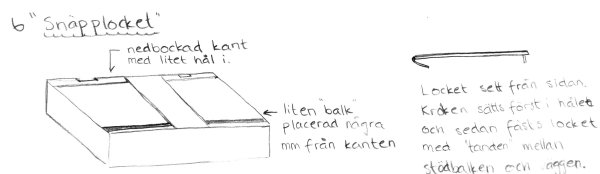
5 Gångjärn



Figur 20: Skiss över koncept 5.

Koncept 6 - Snäpplocket

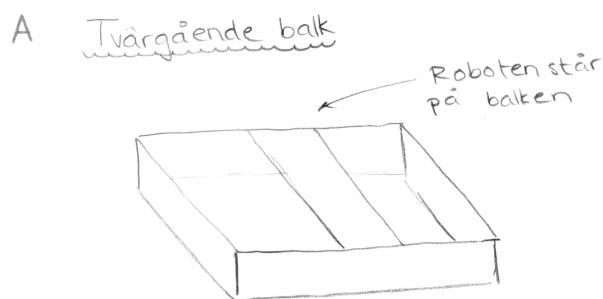
Koncept 6, som visas Figur 21, bygger på en lösning där locket är böjd i ena änden. Denna del skjuts in i ett hål i lådan och viks sedan över så att locket hamnar på plats.



Figur 21: Skiss över koncept 6.

Koncept A - Tvärgående balk

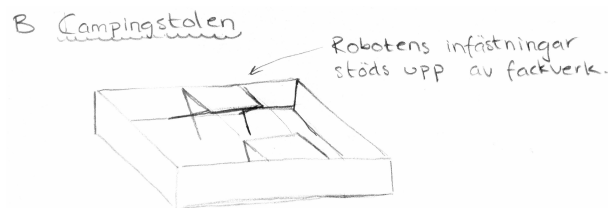
Koncept A löser problemet angående hur roboten ska fästas på lådan utan att förhindra tillgängligheten till innehållet i lådan och kan, liksom koncept B, kombineras med alla ovanstående koncept. Konceptet går ut på att en tvärgående list läggs över lådans långsidor och svetsas fast och visas i Figur 22.



Figur 22: Skiss över koncept A.

Koncept B - Campingstolen

Koncept B har två böckade plåtbitar som fäst i mitten och på insidan av lådans långsidor. Därefter kan roboten monteras med varsitt ben på respektive plåtbit. Konceptet visas i Figur 23.



Figur 23: Skiss över koncept B.

A.2 Val av koncept

Efter att dessa koncept tagits fram kunde projektgruppen direkt eliminera koncept 2 och 3. Koncept 2 togs bort på grund av sin höga tillverkningskomplexitet och koncept 3 då det fanns det en risk att magneterna som skulle hålla fast locket på bottenplattan skulle påverka elektroniken med sitt magnetfält. Nedan i Figur 24 visas Pughs matris.

Då dessa koncept var skapade togs en Pughs matris fram. Ur denna drogs slutsatsen att koncept A1, B1 och A4 var bäst. Efter de bästa koncepten var framtagna togs dessa med till prototypverkaden för vidare konsultation.

Kriterium	A1 (REF)	A4	A5	A6	B1	B4	B5	B6
Tillverkningskomplexitet	REF	+	-	-	-	-	-	-
Åtkomlighet	REF	+	-	0	+	+	+	+
Stabilitet	REF	-	0	-	0	-	0	-
Förslutenhet	REF	-	-	-	0	-	-	-
Oljud	REF	-	-	-	0	-	-	-
$\Sigma +$		2	0	0	1	1	1	1
$\Sigma 0$		0	1	1	3	0	1	0
$\Sigma -$		3	4	4	1	4	3	4
Nettovärde		-1	-4	-4	0	-3	-2	-3
Rangordning	1	2	5	5	1	4	3	4

Figur 24: Pughs matris som använts för att sälla bort lösningar. Koncept A1 är här för referens och de övriga koncepten jämförs relativt referensen för olika kriterier. Koncept A1, B1, A4 och B5 rangordnas högst.

De tre koncept som genererat högst betyg i Pughs matris presenterades för forskningsingenjörerna i prototypverkstaden. Under konsultationen diskuterades vilket koncept som är att föredra ur tillverknings-synpunkt och vilken konstruktion som kommer bli mest estetiskt tilltalande. Forskningsingenjörerna rekommenderade tillverkning av koncept A4, dock med några ändringar. Istället för att tillverka locket så att det passar i lådan kommer det istället omsluta lådans ytterkanter. För att försäkra att locket inte ramlar av då roboten accelererar eller retarderar kommer två popnitar användas.