Kandidatarbete EENX15-18-12

Autonom städrobot

Framtagning av dellösningar till en städrobotsprototyp

FANNIE HENRIKSSON, JOHANNA HENRIKSSON WENNSTRÖM, KEVIN JOHANSSON BILLSKOG, ALICE KARLSSON, RASMUS NORSTRÖM

Handledare: Jonas Sjöberg
Examinator: Jonas Fredriksson

CHALMERS

Institutionen för Signaler och System
Chalmers tekniska högskola
Göteborg, Sverige, 2018-05-14

Abstract

Autonomous cleaning robots are currently not adapted for industrial settings. This report examines the possibility to develop integrated partial solutions for an autonomous cleaning robot prototype suited for industries. The prototype will be operating in an industrial facility that belongs to *Resource for Vehicle Research at Chalmers*. Four partial solutions are developed; propulsion, obstacle avoidance, dirt detection and global navigation. The propulsion consists of programming on the chosen platform *Husqvarna Research Platform* with the software *Robot Operating System*. The obstacle avoidance consists of wiring and programming of ultrasonic sensors. The dirt detection is done with a camera and uses image analysis to decide if the floor of the facility is dirty. Global navigation includes positioning and consists of a neural network which makes recognition of objects in the facility possible. To verify the set up requirements of the partial solutions several validation tests are arranged. The prototype consists of all the partial solution except for the global navigation which works separately. The prototype fulfils a majority of the predetermined requirements with some difficulty with obstacle avoidance and global navigation. The prototype can be autonomously propulsed with well functioning dirt detection. The prototype can be further improved and many functions can be added in the future.

Sammandrag

Autonoma städrobotar är idag inte anpassade för industrilokaler. Den här rapporten undersöker möjligheten att utveckla samverkande dellösningar till en autonom städrobotsprototyp dimensionerad för industrier. Prototypen ska verka i en industrilokal tillhörande *Resource for Vehicle Research at Chalmers*. Fyra dellösningar utvecklas; framdrivning, hinderundvikning, smutsdetektion samt global navigering. Framdrivningen består av programmering av den valda plattformen *Husqvarna Research Plattform* med programvaran *Robot Operating System*. Hinderundvikningen består av koppling samt programmering av ultraljudssensorer. Smutsdetektionen genomförs med hjälp av en kamera och bildanalys för att avgöra om golvet i lokalen är smutsig. Global navigation innefattar positionering och består av ett neuralt nätverk som möjliggör igenkänning av objekt i lokalen. Samtliga dellösningar genomgår valideringstester för att verifiera uppsatta krav. Prototypen består utav alla dellösningar förutom global navigation som fungerar separat. Dellösningarna uppfyller en majoritet av alla uppsatta krav där bristområden finns inom hinderundvikningen och den globala navigationen. Prototypen kan framdrivas autonomt med välfungerande smutsdetektering. Prototypen har stor förbättringspotential och många funktioner kan i framtiden läggas till.

Innehåll

1

Inledning

Robotar finns överallt i dagens samhälle och de senaste åren har autonoma robotar blivit allt mer populära eftersom tekniken blivit mer förfinad. En autonom robot är en robot som agerar självständigt genom att lösa problem och uppgifter utan mänsklig kontroll, *se* [1]. Städrobotar i form av autonomt gående dammsugare blir allt vanligare i svenska hem men dessa städrobotar är inte dimensionerade för industrilokaler.

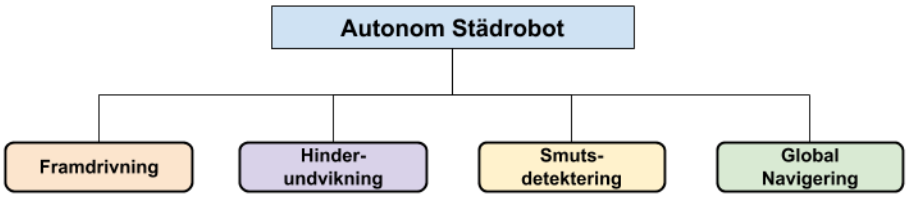
Det finns olika typer av städmaskiner som förekommer i dagens industri. För att städa industrilokaler används oftast skurmaskiner, sopmaskiner eller en kombination av dem båda. Maskinerna kan vara åkbara eller handdrivna men samtliga varianter kräver manuell styrning. Att städa en stor industrilokal är både tids- och resurskrävande och i dagsläget är automatiserade skur-, sop- eller kombiskurmaskiner inte etablerade, *se* [2] *och* [3].

1.1 Syfte

Syftet är att utveckla dellösningar till en autonom städrobotprototyp dimensionerad för industrier. Samtliga dellösningar skall valideras och om möjligt implementeras, sammansättas och samverka i den färdiga prototypen.

1.2 Problemformulering

Städrobotsprototypen skall endast verka i en känd industrilokal. Lokalen sätter krav på prototypens utformning och arbetssätt. Utmaningarna som prototypen ställs inför i lokalen har identifierats och delats upp i fyra delproblem, *se figur* 1.1.



Figur 1.1: De fyra delproblem till prototypen som skall lösas.

1.2.1 Beskrivning av prototypens arbetsmiljö

Prototypen skall vara verksam i en lokal tillhörande Resource for Vehicle Research at Chalmers, även kallat Chalmers fordonslabb, *se bilaga* A för planritning. Forskningsfordon trafikerar lokalen frekvent och drar in mycket grus och därmed behöver lokalen städas ofta. Gruskornen är i genomsnitt ca 2 mm^3 stora. Det förekommer också ofta avklippta buntband på golvet. Golvet i labbet är ljusgrå målad betong. Miljön i lokalen är väldigt dynamisk då bilar, människor, bord, stolar, sladdar och verktyg flyttas runt kontinuerligt. *Se figur* 1.2. Det finns även många trånga utrymmen i lokalen.

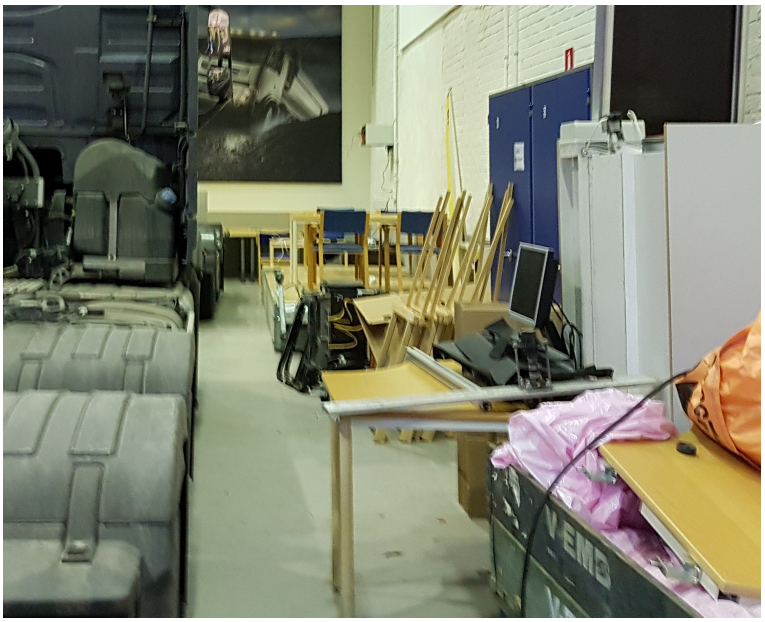
Figur 1.2: I fordonslabbet finns det många objekt, både fasta och rörliga som prototypen måste ta hänsyn till.

1.2.2 Delproblem I: Framdrivning

En autonom prototyp skall kunna framdrivas självständigt. Teknik som möjliggör detta inkluderas under delproblem I: Framdrivning. Med autonom framdrivning menas att prototypen självständigt utan manuell styrning kan ta sig fram. En framdrivande plattform måste kunna ta emot styrinput och agera därefter. Svårigheter med autonom framdrivning är att de sammankopplade systemen och kommunikationen som styr framdrivningen måste vara pålitlig. Det är också viktigt att plattformen inte utgör någon fara och har säkerhetsystem för att förhindra eventuella olyckor.

1.2.3 Delproblem II: Hinderundvikning

En autonom prototyp bör inte köra in i objekt. Teknik som möjliggör hinderundvikningen inkluderas under delproblem II: Hinderundvikning. Svårigheterna med hinderundvikningen i lokalen är att hänsyn måste tas till rörliga och fasta objekt. Prototypen bör inte gå för nära hinder och undvika att fastna i trånga utrymmen eller passager, *se figur* 1.3.



Figur 1.3: Det kan finnas många trånga utrymmen i labbet beroende på hur många fordon som befinner sig i lokalen samtidigt.

1.2.4 Delproblem III: Smutsdetektering

För att effektivisera robotens städning bör den kunna identifiera grus som finns i lokalen under förhållanden likt *figur* 1.4 och buntband likt *figur* 1.5. Teknik som möjliggör detekteringen inkluderas under delproblem III: Smutsdetekering. De största utmaningarna ligger i att skapa en anpassad lösning som inte klassificerar föremål eller skuggor som smuts.

Figur 1.4: Prototypen bör kunna identifiera grus som smuts i dessa förhållanden.

Figur 1.5: Exempelbild av buntband som bör kategoriseras som smuts.

1.2.5 Delproblem IV: Global Navigering

För att optimera prototypens framdrivning är det fördelaktigt att prototypen är medveten om sin position i rummet. Teknik som möjliggör den globala navigeringen inkluderas under delproblem IV: Global Navigering. De största utmaningarna är att välja och utforma en dellösning som ger en tillräckligt noggrann positionering genom att utnyttja den kända miljön som prototypen är verksam i.

1.3 Avgränsningar

På grund av tidsbegränsning togs det ingen hänsyn till exempelvis uppstädning eller tömning av smuts i prototypen, utan endast dellösningar till delproblemen I-IV undersöktes. Utöver de verktyg som tillhandahölls av Chalmers tekniska högskola

och Chalmers fordonslabb finns en begränsande budget på 5 000 kr. Då prototypen inte ska tillverkas i stora volymer lades ingen fokus på prototypens estetiska design eller materialval som är lämpligt vid massproduktion. Ingen hänsyn togs heller till prototypens ljudnivå.

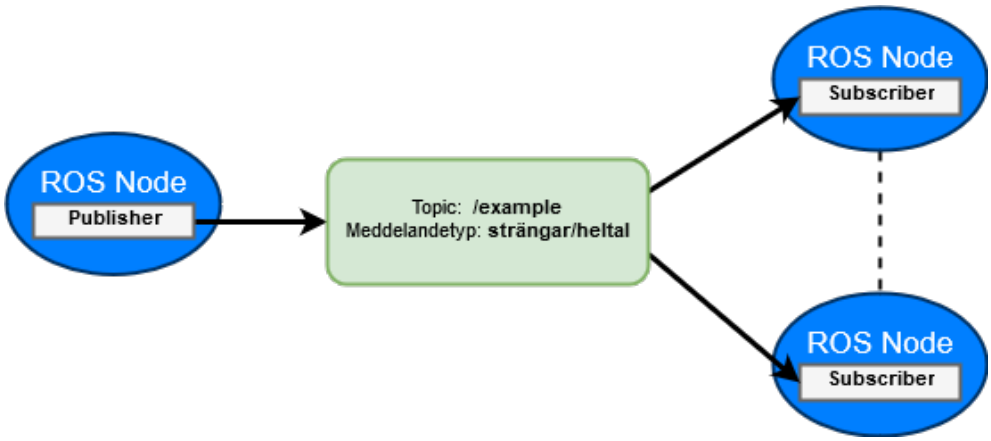
2

Tekniska Specifikationer

Detta kapitel ger bakgrundsinformation kring den teknik som används för att lösa delproblemen I-IV och den bakomliggande teorin. Hård- och mjukvaruteknik presenteras.

2.1 Framdrivning: Robotics Operating System

Robotics Operating System (ROS) är ett projekt med öppen källkod bestående av en samling programvaruramar som underlättar kommunikationen mellan mjuk- och hårdvara i robotprogram. ROS är utrustat med verktyg, färdiga bibliotek och är kompatibelt med simuleringsprogram som Gazebo. Tack vare den öppna källkoden är ROS välanvänt och väldokumenterat inom forskning. Vid körning av program i ROS så startar användaren skriptfiler, kallat ROS-Nodes, som kommunicerar med varandra genom ROS-Topics. Dessa topics har en meddelandetyp bestående av exempelvis strängar eller heltal. Varje nod kan prenumerera och publicera information till flera ROS-Topics och samtidigt exekvera kod, *se figur* 2.1. ROS-Nodes är oftast skrivna i C++ eller Python men är kompatibelt med fler programmeringsspråk, *se* [4] *och* [5].



Figur 2.1: Illustrering av kommunikation mellan ROS-Nodes via ROS-Topics. Noderna till höger i figuren prenumererar på meddelanden från noden till vänster.

2.2 Framdrivning: Husqvarna Research Platform

Husqvarna Research Platform (HRP) är en Husqvarna gräsklippare, modell 430X, som är speciellt framtagen för forskning kring autonoma robotar. Roboten är utrustad med mjukvaran ROS Kinetic, integrerat batteri, tillhörande laddstation och moderkort. Plattformen har också integrerade sensorer som känner av bland annat kollisioner och lyft. På motorerna finns det pulsgivare som läser av rotationen på hjulen. Plattformen har en skärm som visar bland annat batterinivå, inställningar och varningar, *se* [6]. Dimensionerna av HRP presenteras i *figur* 2.2, *se* [7].

Dimensioner HRP	
Ytkapacitet	3200m²
Batterityp	Li-ion
Laddningstid	65 min
Högsta Ljudnivå	58 dB
Max sluttning	45 %
Maximal lutning	24 °
Vikt	13.0 kg
Storlek LxBxH	72x56x31 cm
Normal drifttid en laddnig	135 min

Figur 2.2: Husqvarnas gräsklippare modell 430X med tillhörande dimensioner.

Till HRP:s operativsystem tillkommer en del ROS-Nodes där noden *am_driver_safe* är ett gränssnitt till gräsklipparens hårdvara via serieport. De topics som noden behandlar gör det möjligt att kontrollera robotens status och ge roboten kommandon, *se tabell* 2.1. Noden *hrp_teleop* är ett styrgränssnitt som kommunicerar med *am_driver_safe* där tangentbordstryckning används för att styra plattformen på 9 olika sätt samt ställa in linjär- och rotationshastighet i både simulering och i verkligheten.

Tabell 2.1: ROS-topics som noden *am_driver_safe* behandlar.

Publicerade topics	**Beskrivning**
/odom	Position av robot given pulsgivardata
/loop	Status av begränsningskabelssensorer
/sensor_status	Status av kollisionsensorer
/wheel_encoder	Antalet pulser lästa från roboten
/battery_status	Spänning och ström för Li-Ion batteriet i roboten
Prenumererade topics	**Beskrivning**
/cmd_vel	Given hastighet till roboten (Linjär- och vinkelhastighet)
/cmd_mode	Givet kontroll-läge

2.3 Hinderundvikning: Ultraljudssensorer

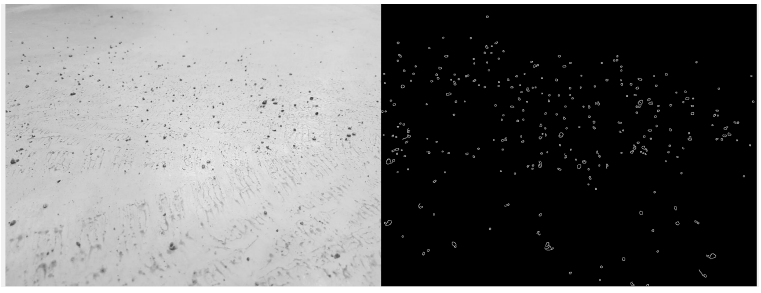
Ultraljudssensorer skickar ut en ljudpuls med en vald frekvens över 20 kHz. Ljudpulsen träffar hinder i vågens väg och reflekteras tillbaka till sensorn. Eftersom pulsens hastighet är känd kan tiden för reflektionen användas för att beräkna distansen till objektet, *se* [8] *och* [9]. Sträckan beräknas enligt

$$s = \frac{t \cdot v}{2} \tag{2.1}$$

där s är sträckan till hindret, v är ljudets hastighet i luft och t är den totala tiden.

2.4 Smutsdetektion: Kantdetektion med Cannymetoden

Kantdetektion är ett sätt att automatiskt karaktärisera och klassificera objekt i bilder. För att detektera kanter i bilder används Cannymetoden, *se figur* 2.3. Kanter ger en skarp förändring i bildens intensitet och genom att undersöka bildens gradient kan kanter upptäckas. Cannymetoden använder sig av två olika tröskelvärden som anger hur skarp förändringen ska vara för att indikera en kant. Metoden resulterar i en god detektion eftersom både skarpa och svaga förändringar detekteras i bilden. Dock registreras kanterna endast om de svaga förändringarna återfinns sammankopplat med en skarp förändring. Detta tillvägagångssätt är mer precist än andra kantdetektionsmetoder med endast ett tröskelvärde, *se* [10].



Figur 2.3: Cannymetoden applicerad på en bild med grus.

Cannymetoden beskrivs mer utförligt i följande steg *enligt* [10] *och* [11]:

1. Två filter appliceras på bilden för att minska störningar och falska kanter. Det första filtret är ett Gaussfilter enligt följande:

$$G(L) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{L^2}{2\sigma^2}} \tag{2.2}$$

   där L är filtrers längd och σ är standardavvikelsen.

Det andra filtrets funktion är derivatan av det första filtret enligt:

$$G'(L) = \frac{dG(L)}{dL} \tag{2.3}$$

När filtren har applicerats får man en ny version av bilden som består av två matriser. Matriserna representerar gradienterna i x respektive y-led för bildens pixlar.

2. Magnituden av gradienterna beräknas i varje pixel (i, j) enligt:

$$MagGrad_{i,j} = \sqrt{|\frac{dG(x_i)}{dx}|^2 + |\frac{dG(y_j)}{dy}|^2} \tag{2.4}$$

3. Magnituderna normaliseras genom att dela varje pixels magnitud på den största magnituden vilket ger värden mellan noll och ett enligt:

$$MagGrad_{i,j} = \frac{MagGrad_{i,j}}{MagGradMax} \tag{2.5}$$

4. Tröskelvärden bestäms. Det höga tröskelvärdet samt tröskelkonstanten är förinställda parametrar. Det låga tröskelvärdet bestäms enligt:

$$Tröskelvärde_{låg} = Tröskelvärde_{hög} \cdot Tröskelkonstant \tag{2.6}$$

5. För magnituder vars värde är mindre än det låga tröskelvärdet sätts värdet till noll i matrisen. Magnituder vars värde är större än det höga tröskelvärdet kallas skarpa kanter och ansätts värdet ett i matrisen. De magnituder vars värden ligger emellan det höga och låga tröskelvärdet kallas för svaga kanter. Om någon av en svag kants åtta kringliggande pixlar har en skarp kant, sätts dess värde till en etta, annars noll.

2.5 Global navigering: Deep learning

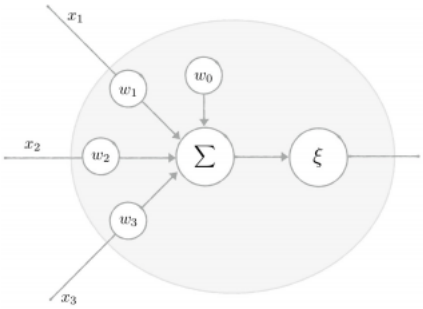
Deep learning är välanvänt inom bilanalys. Genom att simulera den mänskliga hjärnans design, arbetssätt och funktion skapas artificiell intelligens. Syftet med artificiell intelligens är att imitera hjärnans förmåga att lösa problem, dra slutsatser och lära in ny kunskap, *se* [12]. Maskiners förmåga att bearbeta och förstå data utan förprogrammering kallas för maskininlärning. Genom att träna maskiner med hjälp av stora mängder data kan maskinen få en egen uppfattning och fatta välgrundade beslut. Ju mer indata maskinen får desto bättre blir besluten, *se* [13].

Deep learning är en typ av maskininlärning som använder sig av simulerade neurala nätverk för att behandla och bearbeta indata. Med tiden kan maskinen fatta mer intelligenta beslut, *se* [13]. Genom att skapa en abstraherad modell av den mänskliga hjärnan kan ett artificiellt bildanalyssystem som kan lära sig att känna igen objekt

skapas. Exempelvis om nätverket tränas med bilder på katter så kommer systemet att lära sig att hitta och känna igen karaktärsdrag på kattbilderna. När programmet sedan testas på en ny bild så letar nätverket efter de inlärda karaktärsdragen och om de hittas så är det med största sannolikhet en katt på testbilden.

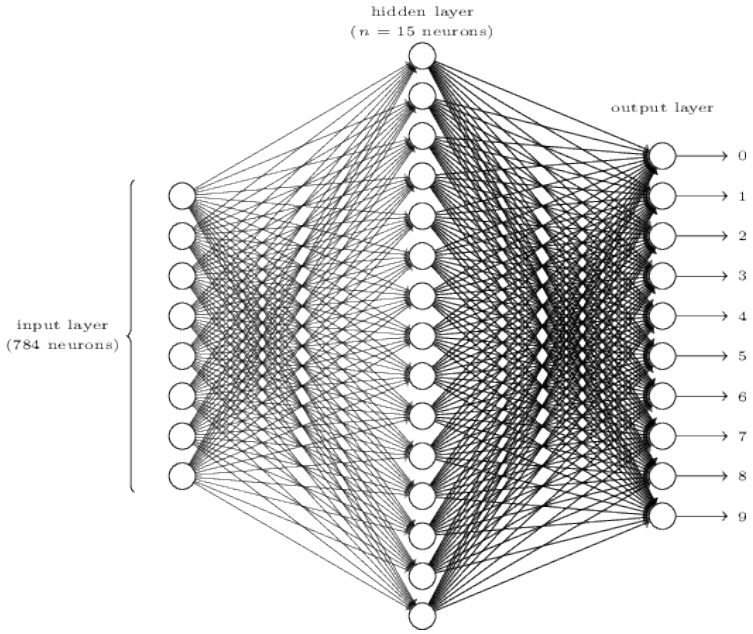
2.5.1 Uppbyggnad och träning av simulerade neurala nätverk

Ett simulerat neuralt nätverk består av simulerade neuroner, noder, som länkas samman till ett nätverk. Neuronen kan simuleras som en booleansk funktion. För att simulera axonerna och synapserna i hjärnan multipliceras insignalen till en nod med olika vikter, *se figur* 2.4. När det neurala nätverket tränas kalibreras vikterna kontinuerligt, *se* [14].



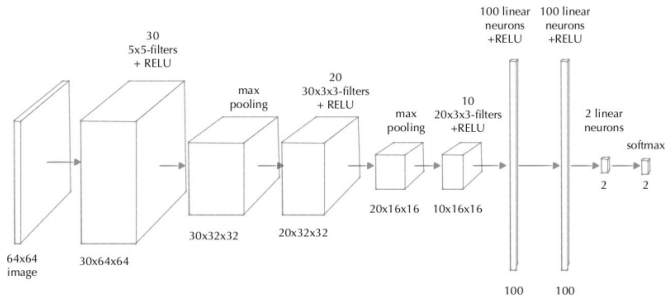
Figur 2.4: Exempel på en simulerad neuron med insignaler x_1, x_2, x_3 och olika vikter w_0, w_1, w_2, w_3 som summeras ihop till en utsignal ξ.

De simulerade neuronerna delas in i olika lager där varje lager bara får indata från det tidigare lagret, *se figur* 2.5. Därmed är det bara det första lagret som får indata från hela bilden. Arkitekturen på nätverket består utav olika kombinationer av lager. Lagren har olika funktioner som till exempel convolutionlager eller maxpoolinglager, *se* [14].



Figur 2.5: Exempel på neuralt nätverk med tre lager och 784 neuroner som behandlar den ursprungliga indatan, *se* [15].

Ett convolutionnätverk är ett neuralt närverk som ofta appliceras inom bildanalys. Convolutionnätverket ser bilder som ett tredimensionellt objekt uppbyggt av matriser. Neuronernas vikter bygger upp ett matrisfilter i två dimensioner. Den tredje dimensionen specificerar färg. Skalärprodukten mellan vikterna och indatan beräknas och en extra viktningsterm adderas. Genom att applicera maxpoolingfilter reduceras dimensionerna på 3D-matrisen och bara den viktigaste informationen skickas till nästa lager med hjälp av ett så kallat fully connected layer som kopplar ihop rätt neuroner i de olika lagren, *se figur* 2.6 och *se* [14].



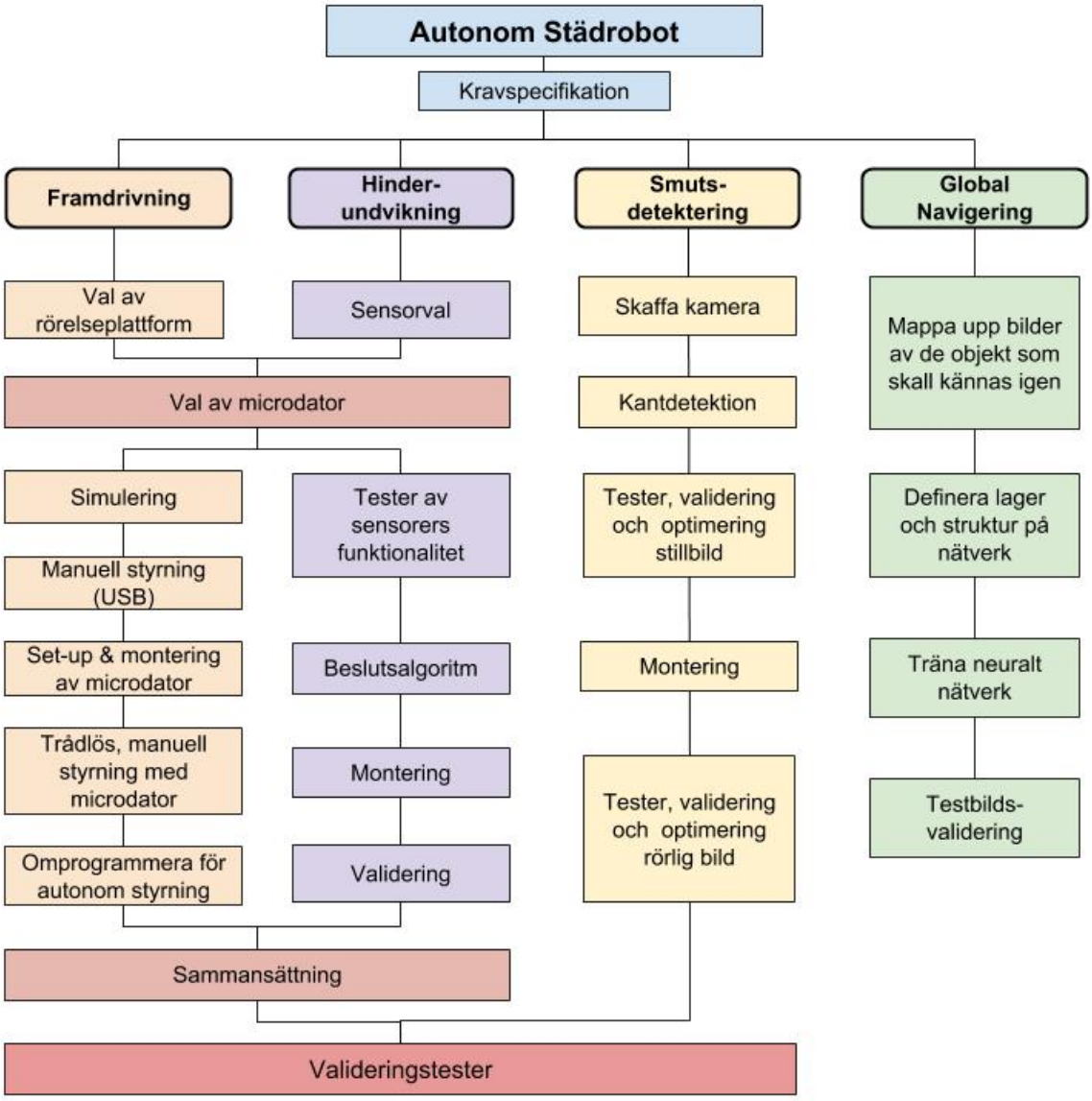
Figur 2.6: Exempel på ett convolutionnätverk, *se* [14].

För att träna det neurala nätverket ges nätverket känd testdata med kända svar. Nätverket tar en liten bit av bilden i taget och låter det passera genom filtren. Sannolikhetsberäkningar används för att justera vikterna och minimera fel med hjälp av den stokastiska gradienten. Denna process upprepas tills hela bilden har passerat alla filter. Ju fler filter nätverket består av desto bättre blir nätverket på att känna igen mönster på tidigare okända bilder, *se* [14].

3

Metod

Konstruktionen av lösningar till delproblemen I-IV och utvecklandet av mjukvara för prototypen beskrivs. För att finna möjliga lösningar på delproblemen genomfördes först litteraturstudier. Utifrån detta skapades ett gemensamt tillvägagångssätt för att lösa delproblemen till prototypen, *se figur* 3.1. Detaljer och samtlig kod kan granskas här: *https://github.com/bustroll/Kandidatarbete*.



Figur 3.1: Flödesdiagram över tillvägagångssättet för att angripa delproblemen till prototypen. Notera att dellösningen till den globala navigeringen verifieras separat från övriga dellösningar.

3.1 Uppsättning av krav

Kravspecifikation utvecklades efter problemformuleringen, *se avsnitt* 1.2.5. För att få en prototyp som kan ta sig runt i en industrilokal sattes maxgränser på storleken, *se tabell* 3.1. Uppsatta krav och önskemål för dellösningarna I-IV ses i *tabellerna* 3.2, 3.3, 3.4 respektive 3.5.

Tabell 3.1: Uppsatta dimensionskrav för städrobotsprototypen.

Dimensionering	
Vikt <100 kg	Krav
Bredd <1 m	Krav
Längd <2 m	Krav
Höjd <1 m	Krav

Tabell 3.2: Krav och önskemål gällande framdrivning av prototypen.

Framdrivning	
Beskrivning	**Krav / Önskemål**
Larma vid <20 % batteri	Krav
Trådlös framdrivning	Krav
Nödstopp	Krav
Kunna åka framåt & bakåt (ca 5 km/h)	Krav
Kunna rotera	Krav
Kapacitet för städning av 80 % av den fria ytan	Önskemål
Kunna signalera batterinivå	Önskemål
Kunna åka hem vid lågt batteri	Önskemål

Tabell 3.3: Krav gällande hinderundvikning.

Hinderundvikning	
Undvika kollision med fasta objekt med en höjd över 2 cm och en bredd på minst 15 cm. Stanna >20 cm innan kollision.	Krav
Undvika fasta objekt med ett överhäng på 2-50 cm och en bredd på minst 15 cm. Stanna >200 cm innan kollision.	Krav
Undvika rörliga objekt med en höjd över 2 cm och en bredd på minst 15 cm. Stanna >50 cm innan kollision.	Krav
Undvika rörliga objekt med ett överhäng på 2-50 cm och en bredd på minst 15 cm. Stanna >50 cm innan kollision.	Krav
Dubbla sensorsystem.	Krav
Larma och stängas av om prototypen fastnat.	Krav

Tabell 3.4: Krav och önskemål gällande detektering av smuts i lokalen.

Smutsdetektering	
Beskrivning	**Krav / Önskemål**
Detektera grus med en kornstorlek på minst 2 mm^3	Krav
Detektera grus under rörelse (5 km/h)	Önskemål
Inte detektera bromsspår som smuts	Önskemål
Inte detektera sladdar som smuts	Önskemål
Inte detektera skuggor som smuts	Önskemål
Detektera grus på 0,5 m avstånd	Önskemål
Fungera i varierande ljusstyrka	Önskemål
Detektera vatten (stillastående)	Önskemål

Tabell 3.5: Krav och önskemål angående prototypens globala navigering.

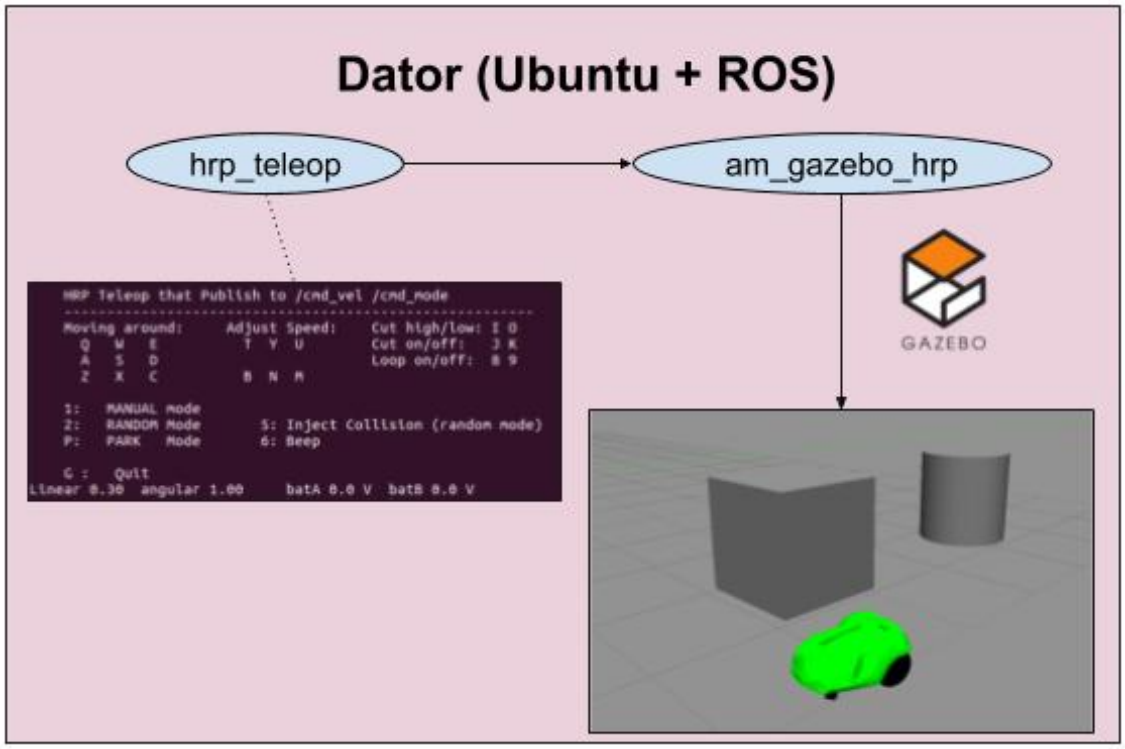
Global Navigering	
Beskrivning	**Krav / Önskemål**
Ej åka ut ur verkstaden	Krav
Möjlighet att utnyttja insamlad data för att effektivisera städningen	Önskemål
Navigera till specifika områden	Önskemål

3.2 Framdrivning

För att möjliggöra prototypens framdriving valdes den modifierade Husqvarna gräsklipparen Husqvarna Research Platform (HRP) som rörelseplattform. Motiveringen bakom plattformsvalet var att plattformen uppfyller en stor mängd önskemål och satta krav ur den framtagna kravspecifikationen som exempelvis nödstopp och förmågan att kunna signalera batterinivå.

3.2.1 Simuleringar av framdriving med ROS

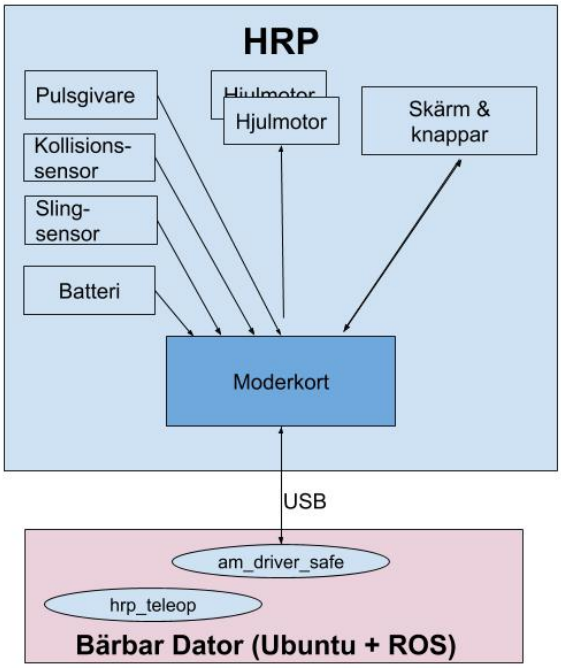
För att simulera och verifiera HRP:s framdrivning och skrivna ROS-program före praktisk implementering användes simuleringsprogrammet Gazebo som endast är kompatibelt med Linux. Exempelvis testning av manuell styrning via tangentbordskontroll skedde med styrgränssnittsnoden *hrp_teleop* som kommunicerar med Gazebo genom noden *am_gazebo_hrp*, *se figur* 3.2.



Figur 3.2: Simulering av styrning med *hrp_teleop* (tangentbordskontroll) tillsammans med programvaran Gazebo.

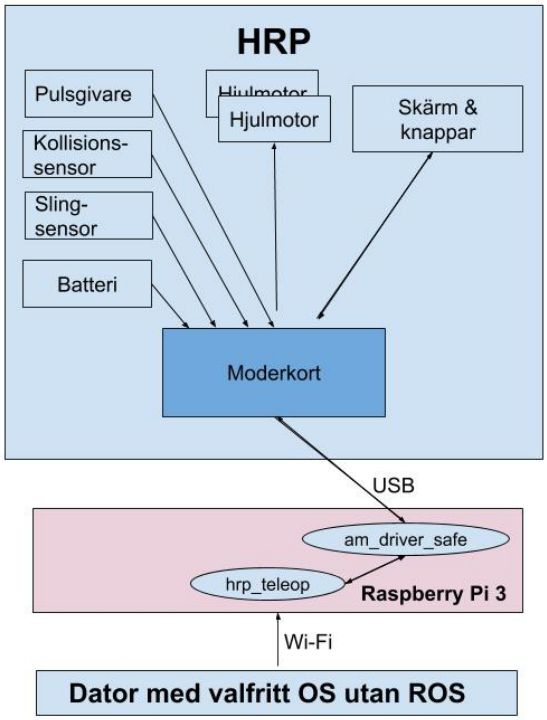
3.2.2 Uppsättning av framdrivning med HRP

För att styra HRP med manuell kontroll användes först en Linux-dator som kommunicerade till plattformen via USB. Styrningen skedde genom tangentbordskontroll via gränssnittsnoden *hrp_teleop* som kommunicerar till noden *am_driver_safe* med USB-kabeln deklarerad som serieport. Den manuella, icke-trådlösa styrningen verifierade att HRP fungerade som väntat med ROS-Kinetic och att de tillhörande noderna fungerade likadant i simuleringen som i praktiken, *se figur* 3.3.



Figur 3.3: Första styrningstestet av HRP med tangentbordskontroll och kommunikation via USB som serieport.

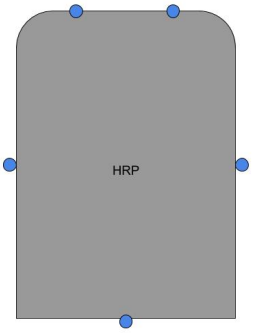
För att göra styrningen av HRP trådlös och kommunikationen ej längre begränsad till endast linux-datorer med en ROS installation via USB-kabel användes Raspberry Pi 3 (RPi3) *se figur* 3.4. Skälet till att RPi3 valdes som styrkort var på grund av att RPi3 har en ARMv7l arkitektur och därmed är kompatibelt med ROS Kinetic tillsammans operativsystemet Linux 16.04. För att kunna använda RPi3 ihop med HRP och ROS-Kinetic införskaffades ett SanDisk Ultra Micro-SD-kort. På SD-kortet installerades sedan en iso-bild av operativsystemet Ubuntu Mate 16.04, specialformat för RPi3, med hjälp av programvaran Etcher. Efter konfigurering av trådlös kommunikation via Secure Shell (SSH), IP-adresser och trådlös internetanslutning installerades sedan ROS Kinetic (Full Desktop Version) på RPi3. Nödvändiga paket från Husqvarna Research importerades. Vid kompillering av paketen användes kommandot *catkin_make -j1* för att inte överbelasta RPi3s RAM-minne som annars inte klarar av kompilering av stora ROS-paket. Mikrodatorn är inte dimensionerad för att köra bildanalysprogram men klarar av att få kommandon via ROS.



Figur 3.4: Andra styrningstestet av HRP med tangentborskontroll och trådlös kommunikation via RPi3.

3.3 Hinderundvikning

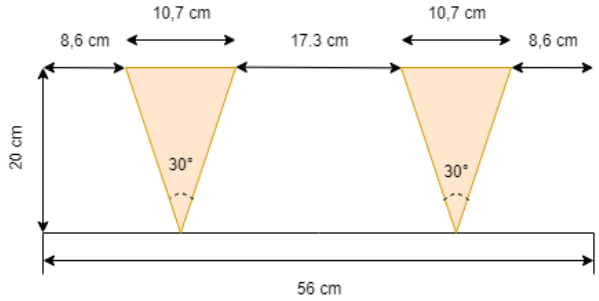
För att upptäcka stora objekt som människor och bilar i prototypens närhet införskaffades ultraljudssensorer av modellen HC-SR04 av tillverkaren ITead Studio. Räckvidden är 2-500 cm och ljudsignalens spridningsvinkel är 30° enligt sensorns produktspecifikation, *se* [16]. För att säkerhetsställa att sensorerna täcker en tillräckligt stor yta runtom prototypen genomfördes geometriberäkningar i Matlab. Fem ultraljudssensorer uppskattades vara tillräckligt för att upptäcka stora föremål. Placeringen av sensorerna på prototypen bestämdes till 2 stycken fram, 1 på vardera sida och 1 på robotens baksida, *se figur* 3.5.



Figur 3.5: Slutgiltig sensorplacering på prototypen.

Sensorerna placerades för att täcka alla tilltänkta färdriktningar. För att uppfylla kraven på överhäng ansågs sensorplaceringen tillräcklig och ingen uppåtpekande sensor installerades. Detta på grund av att ljudvågorna som sprids ur sensorna antogs ha tillräcklig spridning uppåt.

Se figur 3.6 för frontsensorernas täckningsyta på 20 cm avstånd som är det kortaste avståndet prototypen ska befinna sig från hinder. Kravspecifikationen kräver att hinder med bredden 15 cm ska kunna detekteras men 17.282 cm anses vara tillräckligt då exakt ett sådant scenario är extremt osannolikt. För att denna situation ska uppstå måste föremålet dels hamna precis emellan sensorerna och dels inte upptäckas tidigare vilket är ytterst osannolikt då sensorernas avsökningsbredd ökar med avståndet.

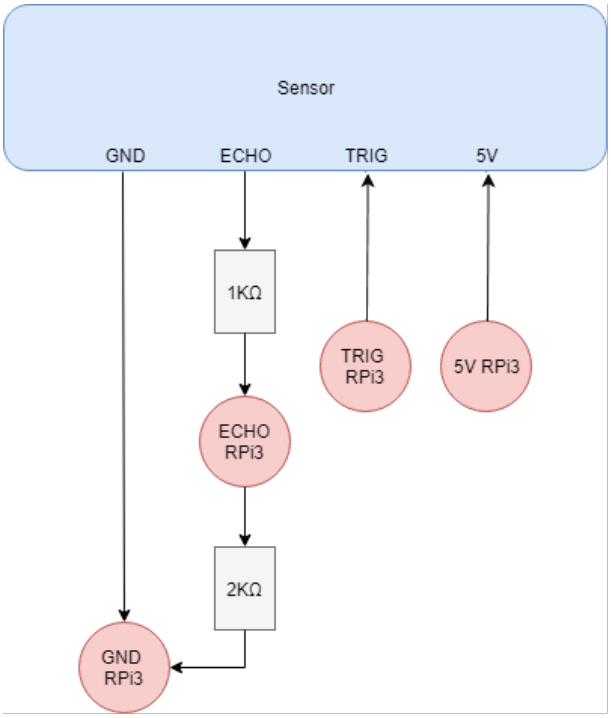


Figur 3.6: Sensorernas avtäckta yta framåt vid ett avstånd på 20 cm.

3.3.1 Koppling av sensorer

Strömförsörjning, beräkningskraft och avståndsmätningen med sensorerna möjliggjordes med hjälp av en RPi3. HC-SR04 har två inportar, en för strömförsörjning (5 V) och en för signalen av mätstart (TRIG). Det finns även två utportar, en till jord (GND) och en som mäter pulstid (ECHO). Sensorn strömförsörjs av RPi3 med 5 V. TRIG får en signal av RPi3 när ultraljudssensorn skall skicka ut en ljudsignal. ECHO skickar ut 5 V till RPi3 samtidigt som TRIG aktiveras tills retur av ljudsignal. Resistorer användes för att ändra spänningen i kretsen mellan ECHO och RPi3 eftersom sensorerna skickar signaler på 5 V men RPi3s GPIO-pinnar är designade

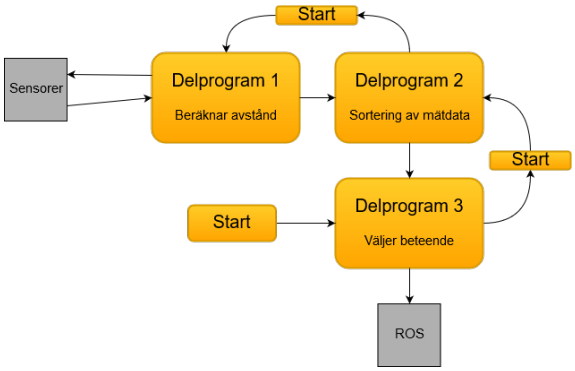
för 3,3 V. RPi3 räknar därefter ut avståndet beroende på ECHO-signalens varaktighet. Flödesschema över kopplingen syns i *figur* 3.7. Alla fem sensorer kopplades på samma sätt. Koppling skedde *enligt* [17].



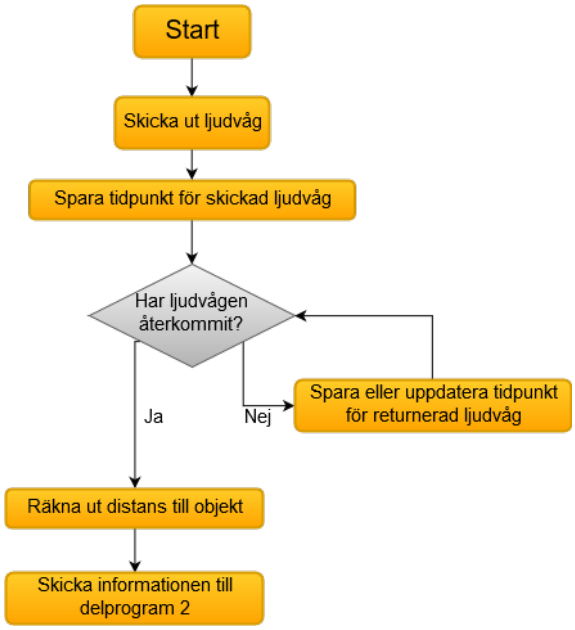
Figur 3.7: Flödesschema över kopplingen av sensorer med RPi3. GND är jord och 5 V fås av RPi3. RPi3 skickar en signal till TRIG som skickar ut ljudsignalen. ECHO skickar ut 5 V till RPi3 tills att ljudsignalen reflekteras tillbaka.

3.3.2 Programmering av sensorer och beteende

Vid programmering av sensorerna användes Python eftersom det är kompatibelt med ROS och RPi3. Koden består av tre delprogram *se figur* 3.8. Delprogram 1 utför avståndsmätningar med hjälp av sensorerna, mätningarna skickas sedan vidare till nästa delprogram, *se figur* 3.9. Mätningen avbryts om ljudsignalen inte reflekterats tillbaka inom 0,029 s.

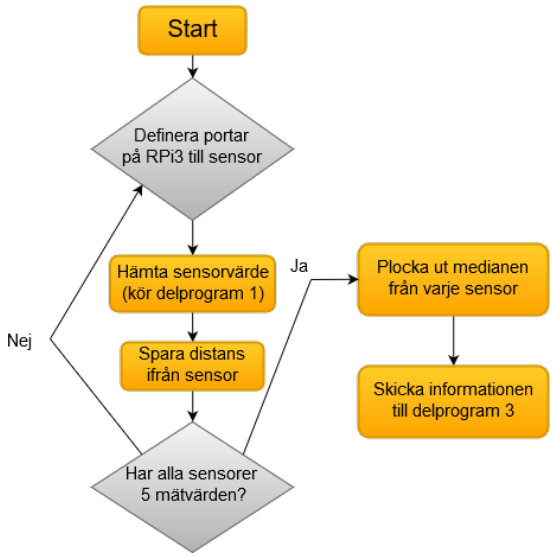


Figur 3.8: Flödesschema över hur de olika delprogrammen kommunicerar mellan varandra och använd hårdvara.

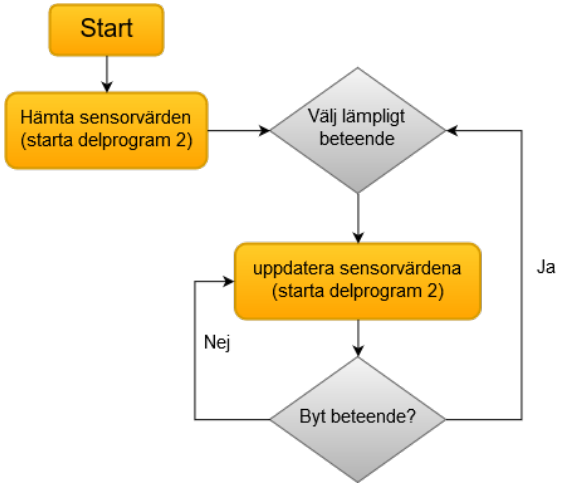


Figur 3.9: Flödesschema över delprogram 1 som utför avståndsmätningar med hjälp av sensorer.

Delprogram 2 kör fem mätningar på varje sensor med hjälp av delprogram 1. Varje sensor aktiveras separat för undvika interferens. En median av de fem mätningarna används för att eliminera mätfel och skickas vidare till delprogram 3, *se figur* 3.10. Delprogram 3 använder mätvärdena för att välja vilket körmönster som används och meddelar därefter ett rörelsesätt till ROS, *se figur* 3.11.



Figur 3.10: Flödesschema över delprogram 2 som beräknar medianen av mätresultatet.



Figur 3.11: Flödesschema med delprogram 3 som bestämmer körbeteende.

Kravspecifikationen skiljer på rörliga och fasta objekt men det är inget som har implementerats i prototypen. De två avstånden från kravspecifikationen har däremot implementerats i programmeringen som gränserna för de två beteendena: slumpmässig körning och kollisionsundvikning.

Slumpmässigt körning innebär att prototypen åker framåt tills den kommer nära ett objekt (<50 cm). Då roterar prototypen en slumpvis vald vinkel mellan 30° och 180° åt höger eller vänster beroende på vilken frontsensor som är närmast hindret.

Kollisionsundvikningen aktiveras om prototypen misslyckats med att upptäcka hinder i tid och kört för nära (<20 cm). Beteendets ändamål är att om möjligt försöka undvika kollision för att därefter återgå till slumpmässig körning. Om prototypen

har kört in i ett för trångt utrymme kommer den backa tills den kommer ut ur området. Prototypen larmar och stängs av om den fastnat i ett trångt utrymme.

3.4 Smutsdetektering

För att kunna detektera grus fästes en webbkamera på prototypen. Kameran som användes var en Logitech HD PRO Webcam C920 med full HD. Smutsdetekteringrogrammet valdes att skapas och simuleras i Matlab eftersom programmet har stöd för skapandet av ROS-Nodes. Matlab tar in videoflödet från webbkameran och tar stillbilder med frekvensen en bild per sekund som sedan konverteras till svartvitt. Med hjälp av Cannymetoden hittas och markeras kanter på gruskornen i bilden. $Tröskelvärde_{hög}$ sattes till 0,35 och $Tröskelvärde_{låg}$ ansattes enligt *avsnitt* 2.4 steg 2. Alla konturer som är ihopsittande fylls sedan i med vit färg, *se figur* 3.12.

Figur 3.12: Bild på indata från webbkamera som kört genom smutsdetektionsprogrammet.

Smutsdetektionsprogrammet mäter hur mycket vitt, alltså mängden registrerade objekt, som det finns i bilden. Om det finns mindre än 99,5 % svart yta registrerar programmet golvet som smutsigt. Efter varje analyserad bild skriver smutsdekteteringsprogrammet ut om ytan är ren eller smutsig. Programmet kan även deklarera i vilken kvadrant av bilden som smutsen befinner sig i.

3.5 Global navigation: Uppbyggnad av simulerade neurala nätverk

Bildigenkänning och ett neuralt nätverk valdes framför andra navigationslösningar, som exempelvis WiFi-trangulering och GPS då det ger en möjlighet att använda den kända omgivningen. Genom att träna nätverket på specifika objekt som finns i lokalen kan prototypen få kändedom om sin position. Ett convolutionnätverk som kan lära sig att känna igen karaktäristiska drag i tagna bilder skapades. Nätverket valdes att skapas och simuleras i Matlab eftersom programmet har stöd för deep learning och skapandet av ROS-Nodes.

Convolutionnätverket är uppbyggt av lager som struktureras enligt *tabell* 3.6. För att läsa in bilder i Matlab krävs ett så kallat image input-lager där storleken och bildfärgen läses in. Nätverket innehåller tre convolutionlager med olika antal filter, 8, 16 respektive 32 stycken med storleken $3 \cdot 3$.

Tabell 3.6: Uppbyggnaden av det neurala nätetsverket som används för att känna igen karaktäristiska drag av objekt i bilder.

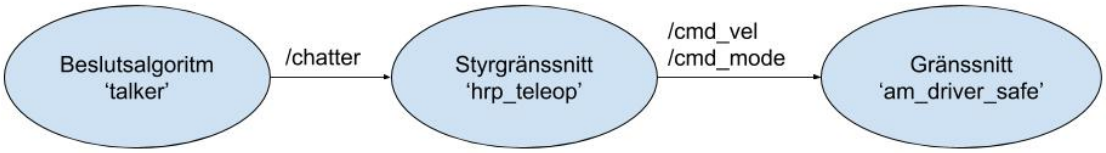
Lager	**Funktion**
1. Image input	Läser in bild storlek och färg
2. Convolution (8 filter)	Beräknar skalärprodukten av vikterna och indatan
3. Normalization	Normaliserar datan
4. RelU	Sätter alla värden mindre än 0 till 0
5. Maxpooling	Reducerar dimensionerna på matrisen
6. Convoltion (16 filter)	Beräknar skalärprodukten av vikterna och indatan
7. Normalization	Normaliserar datan
8. RelU	Sätter alla värden mindre än 0 till 0
9. Maxpooling	Reducerar dimensionerna på matrisen
10. Convoltion (32 filter)	Beräknar skalärprodukten av vikterna och indatan
11. Normalization	Normaliserar datan
12. RelU	Sätter alla värden mindre än 0 till 0
13. Maxpooling	Reducerar dimensionerna på matrisen
14. Fully connected	Kopplar samma neuronerna i de olika lagren
15. Softmax	Normaliserar utdatan
16. Klassificering	Klassificerar med hjälp av sannolikhet

Den stokastiga gradienten används vid träningen av det neurala nätverket. Vikterna kalibreras för att gradienten hela tiden skall minska. Minskningen är eftersträvansvärd då det innebär att felet reduceras. Nätverket tränades med en uppsättning av kända bilder. För att bespara tid användes ett färdigt bildbiliotek föreställande olika trafikskyltar för att träna det neurala nätverket. Träningsuppsättningen bestod av 99 bilder på hastighetsskyltar, 137 bilder på huvudledsskyltar och 106 bilder som innehöll andra objekt, *se figur* 3.13 för exempelbilder. Bilderna var sorterade och etiketterade enligt hastighetsskylt, huvudled och övrigt. Det tränade nätverket gavs sedan okända testbilder som indata och returnerade en klassificeringsetikett, alltså vad nätverket ser för motiv på bilden. Nätverket återkopplar även med vilken sannolikhet som nätverkets klassifikation är korrekt.

Figur 3.13: Exempel över bilder som användes för att träna convolutionnätverket.

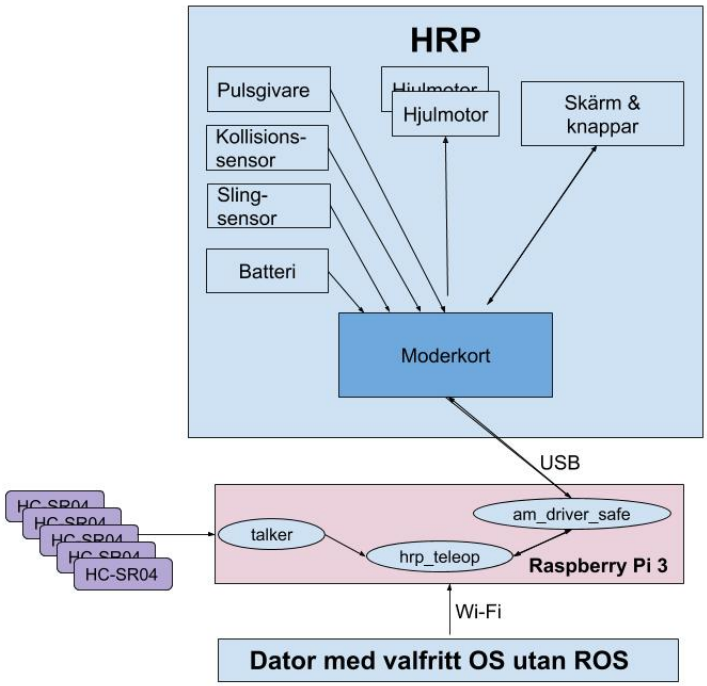
3.6 Sammansättning av system

För att styra prototypen gjordes hinderundvikningens beslutsalgorim i delprogram 3 om till en ROS-Node, talker, som i sin tur publicerar information i ett ROS-Topic, chatter, *se figur* 3.14. Koden i noden *hrp_teleop* modifierades för att ta emot talker som styrsignal istället för tangentbordstryckningar.



Figur 3.14: Illustrering av de aktiva noderna i ROS vid autonom styrning av prototypen med RPi3.

Styrsystemet sattes samman enligt *figur* 3.15. Smutsdetekteringen arbetar separat och påverkar ej beslutsarlgoritmen eller prototypens styrsystem. *Se figur* 3.16 för färdig prototyp. Dellösning IV implementerades ej på den färdiga prototypen.



Figur 3.15: Illustrering av det sammankopplade styrsystemet bestående av samverkan mellan dellösning I och dellösning II.

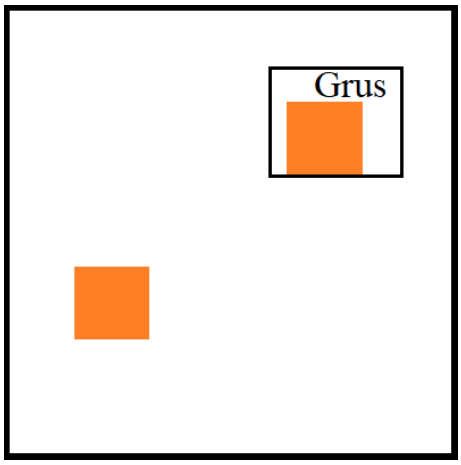
Figur 3.16: Bild på färdig prototyp med implementerat styrsystem och smutsdetektering. Notera att dellösningarna I-III monterats på prototypen men att dellösning III arbetar separat och ej påverkar styrsystemet.

3.7 Validering av system

Ett flertal valideringstester togs fram för att validera de uppsatta kraven enligt *avsnitt* 3.1 för dellösningarna I-III på den färdiga prototypen. Testerna har olika svårighetsgrader och baseras på hur många krav och önskemål som skall valideras per test. Den globala navigeringen verifierades separat.

3.7.1 Valideringstest: Enkel

Valideringstest *Enkel*, *se figur* 3.17, testar om prototypen klarar de mest grundläggande kraven för dellösningarna I-III. Testet prövar om prototypen kan åka slumpmässigt inom ett område på $2,2 \cdot 3,2$ meter som avgränsas med pappersskivor. I ett av hörnen finns ett område på ca 1 m^2 med gruskorn som är mellan $2 - 5$ mm^3 stora. Mängden grus är 1 dl, utspritt slumpmässigt. Två stycken orangea soptunnor som täcker en area på $0,55 \cdot 0,3$ m användes som fasta hinder. Den ena soptunnan var placerad på ett rent område och den andra inom ett område med grus. Testet utfördes med full ljusstyrka i lokalen samt i dunkel belysning.



Figur 3.17: Valideringtest *Enkel* testar de mest grundläggande kraven för dellösningarna I-III. Det finns två hinder i form av soptunnor markerade som orangea fyrkanter i figuren.

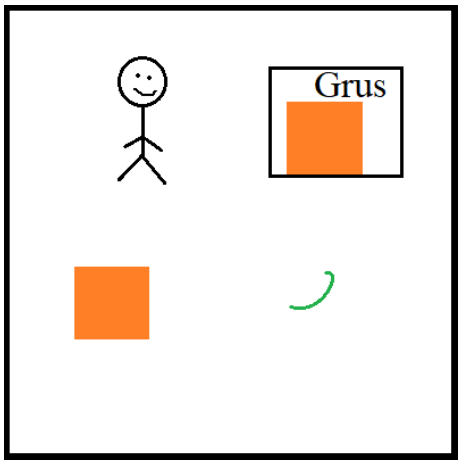
Valideringstest *Enkel* testar följande krav och önskemål:

- Kunna åka framåt och bakåt.
- Kunna rotera.
- Undvika kollision för fasta objekt med höjd över 2 cm och bredd på 15 cm.
- Trådlös framdrivning.
- Detektera grus med en kornstorlek på minst $2mm^3$ under rörelse.
- Detektera grus på 0,5 m avstånd.
- Fungera i varierande ljusstyrka.
- Inte registrera skuggor som smuts.

3.7.2 Valideringstest: Svår

Valideringtest *Svår*, *se figur* 3.18, skiljer sig från valideringstest *Enkel*. Det finns en $0,5$ m sladd liggandes på området som är fritt från grus. I testet *Svår* ska även en person röra sig i testområdet utan att bli påkörd av prototypen. Krav och önskemål som testas utöver valideringstest *Enkel* är:

- Undvika rörliga objekt med en höjd över 2 cm och en bredd på minst 15 cm.
- Inte detektera sladdar som smuts.

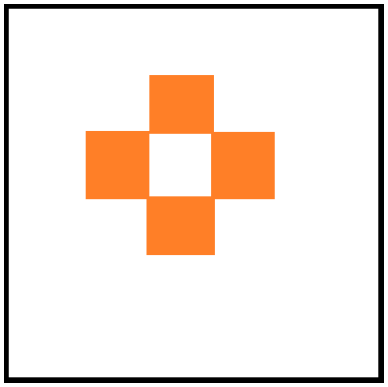


Figur 3.18: Valideringtest *Svår* testar om den färdiga prototypen uppfyller ytterligare krav. I valideringtest *Svår* finns två hinder i form av soptunnor markerade som orangea fyrkanter i figuren. Sladden i testet är markerad som en grön linje. En person rör sig i testområdet.

3.7.3 Valideringstest: Instängd

Valideringstest *Instängd* testar om prototypen kan hantera att bli instängd, *se figur* 3.19. Protypen placeras i mitten av fyra stora hinder. Valideringstest *Instängd* testar följande krav:

- Larma och stängas av om prototypen fastnat.



Figur 3.19: I valideringtest *Instängd* placeras prototypen i mitten av fyra orangea soptunnor som blockerar alla sensorer. Soptunnorna är markerade som orangea fyrkanter.

3.7.4 Testbildsvalidering: Neurala nätverk

Valideringstestet av det tränade neutrala nätverk genomfördes på en dator skild från prototypen. I testet laddas okända bilder in till det neutrala nätverket föreställande skyltar och andra objekt. Nätverket analyserar och klassificerar samtliga testbilder,

se figur 3.20 för exempel på bild som användes för att validera nätverket. Sannolikheten för att bilden var korrekt klassificerad beräknades och det neurala nätverkets identifieringsprecision kontrollerades. För att visa på möjlig implementering i labbet testades nätverket även på utskrivna skyltar, *se figur* 3.21.

Figur 3.20: Exempel på en testbild som användes för att validera nätverket. Bilden innehåller skyltar och många övriga objekt. Nätverket har tränats för att korrekt identifiera skyltarna.

Figur 3.21: Exempel på testbild som användes för att validera möjligheten att enkelt placera ut skyltar för prototypens igenkänning av miljön i industrilokalen.

3.7.5 Simuleringtester: Körbeteende

För att verifiera att delprogrammen fungerade korrekt innan implementering genomfördes simuleringstester på körningsmönstret kollisionsundvikning. Sensorerna monterades på prototypen och testet genomfördes stationärt. Hinder placerades i olika formationer framför sensorerna för att verifiera att koden skickar ut rätt körbeteende i rätt situation. Sensorerna testades genom att placera objekt vinkelrätt framför sensorerna på ett 20 cm avstånd.

3.7.6 Övriga tester

Krav och önskemål som testades separat från presenterade valideringstester:

- Nödstopp.
- Inte detektera bromsspår som smuts.
- Signalera batterinivå.
- Detektera buntband som smuts.

Under arbetets gång implementerades egenskaper i prototypen som ansågs nödvändiga trots att egenskaperna inte fanns med i den ursprungliga kravspecifikationen. Testerna genomfördes på liknade vis som valideringstesterna *Enkel* och *Svår*. Övriga egenskaper som testades var:

- Prototypen roterar ej i trånga utrymmen.
- Prototypen backar ut ur trånga utrymmen.

Krav som valdes att ej testas eller implementeras:

- Larma vid < 20 % batteri.
- Ej åka ut ur verkstaden.
- Undvika rörliga objekt med ett överhäng på 2-50 cm och en bredd på minst 15 cm.
- Undvika fasta objekt med ett överhäng på 2-50 cm och en bredd på minst 15 cm.

4

Resultat

Bedömningen av resultatet består av observationer över huruvida dellösningarna klarar valideringstesterna med avseende på kraven i *avsnitt* 3.7. Dellösningarna utvärderas separat för att se om respektive krav uppfylls.

4.1 Framdrivning

Implementering av Husqvarna Research Platform (HRP) resulterade i uppfyllda krav och önskemål enligt *tabell* 4.1.

Tabell 4.1: Resultat av framdrivningens krav och önskemål.

Krav/Önskemål	**Uppfyllt Mål**
Larma vid lågt batteri	Ja
Kapacitet för att täcka av hela ytan	Ja
Trådlös framdrivning	Ja
Nödstopp	Ja
Åka hem vid lågt batteri	Ej implementerat
Kunna åka framåt & bakåt	Ja
Kunna rotera	Ja
Kunna signalera batterinivå	Ja
Vikt <100 kg	Ja
Bredd, Höjd <1 m	Ja
Längd <2 m	Ja

Prototypen klarar av att trådlöst styras och framdrivas och har 9 st olika rörelsesätt. Dessa olika rörelsesätt listas upp i *tabell* 4.2. Hastigheten i både linjär- och rotationsled kan bestämmas och avläsas. Prototypen kan styras av antingen manuell trådlös kontroll eller av hinderundvikningssystemets beslutsalgoritm. I valideringstesterna fungerade alla styrkommandon från beslutsalgoritmen och därmed fungerar prototypens dellösnining för framdrivning mycket väl. Då prototypen kommunicerar och strömförsörjs trådlöst minskar risken för prototypen att fastna och individer i industrilokalen behöver ej ta hänsyn till sladdar.

Tabell 4.2: Tabell av prototypens olika rörelsesätt.

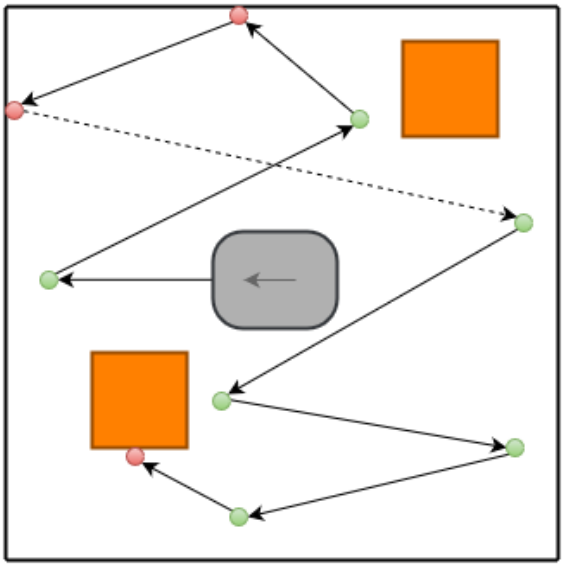
Rörelsesätt	**Beskrivning**
Framåt	Framdrivning av båda hjul
Bakåt	Bakdrivning av båda hjul
Stanna	Ingen drivning av något hjul
Höger-rotation	Bakdrivning av höger hjul Framdrivning av vänster hjul
Vänster-rotation	Bakdrivning av vänster hjul Framdrivning av höger hjul
Höger framrotation runt ett hjul	Framdrivning av vänster hjul Ingen drivning av höger hjul
Vänster framrotation runt ett hjul	Framdrivning av höger hjul Ingen drivning av vänster hjul
Höger bakrotation runt ett hjul	Bakdrivning av vänster hjul Ingen drivning av höger hjul
Vänster bakrotation runt ett hjul	Bakdrivning av höger hjul Ingen drivning av vänster hjul

4.2 Hinderundvikning

Resultatet av simuleringen av kollisionsundvikningen presenteras i *tabell* 4.3. När hinderundvikningen prövades i valideringstesterna krockade ibland prototypen med väggarna *se figur* 4.1. De olika krav och önskemål rörande hinderundvikningen och hurvida målen uppfylldes listas i *tabell* 4.4.

Tabell 4.3: Resulterande beteende vid olika typfall i beteendet kollisionsundvikning

Täckta sensorer	**Resulterat beteende**	**Önskat beteende**
Front & Bakom	Stanna & varna	Ja
Front, höger & vänster	Backa	Ja
Front & vänster	Backa åt höger	Ja
Front & höger	Backa åt vänster	Ja
Vänster, höger & bakom	Stanna och varna	Ja
Vänster & höger	Backa	Ja
Vänster & bakom	Fram åt höger	Ja
Höger & bakom	Fram åt vänster	Ja
Front	Backa	Ja
Bakom	Kör fram & varna	Ja



Figur 4.1: En testkörning av prototypens hinderundvikning i valideringstest *Enkel*. En grön cirkel visar då prototypen upptäckte ett hinder och en röd cirkel visar då den körde in i ett hinder. Pilarna visar körvägen hos prototypen och de är ritade i en heldragen eller streckad linje.

Tabell 4.4: Resultat av valideringtesterna med avseende på hinderundvikningen.

Krav/Önskemål	**Uppfyllt Mål**
Undvika kollision med fasta objekt men en höjd över 2 cm och en bredd på minst 15 cm Stanna >20 cm innan kollision	Nej
Undvika fasta objekt med ett överhäng på 2-50 cm och en bredd på minst 15 cm Stanna >20 cm innan kollision.	Ej testat
Undvika rörliga objekt med en höjd över 2 cm och en bredd på minst 15 cm Stanna >50 cm innan kollision.	Ja
Undvika rörliga objekt med ett överhäng på 2-50 cm och en bredd på minst 15 cm Stanna >50 cm innan kollision	Ej testat
Dubbla sensorsystem	Finns men ej integrerat
Larma och avstängning om prototypen fastnat.	Ja

4.3 Smutsdetektering

Smutsdetektionsprogrammet klarar med mycket god förmånga att särskilja grus från övriga objekt och att registrera rena ytor som rent. Sladdar, skuggor eller väggar registreras inte som grus. I båda valideringstesterna gav smutsdetektionsprogrammet

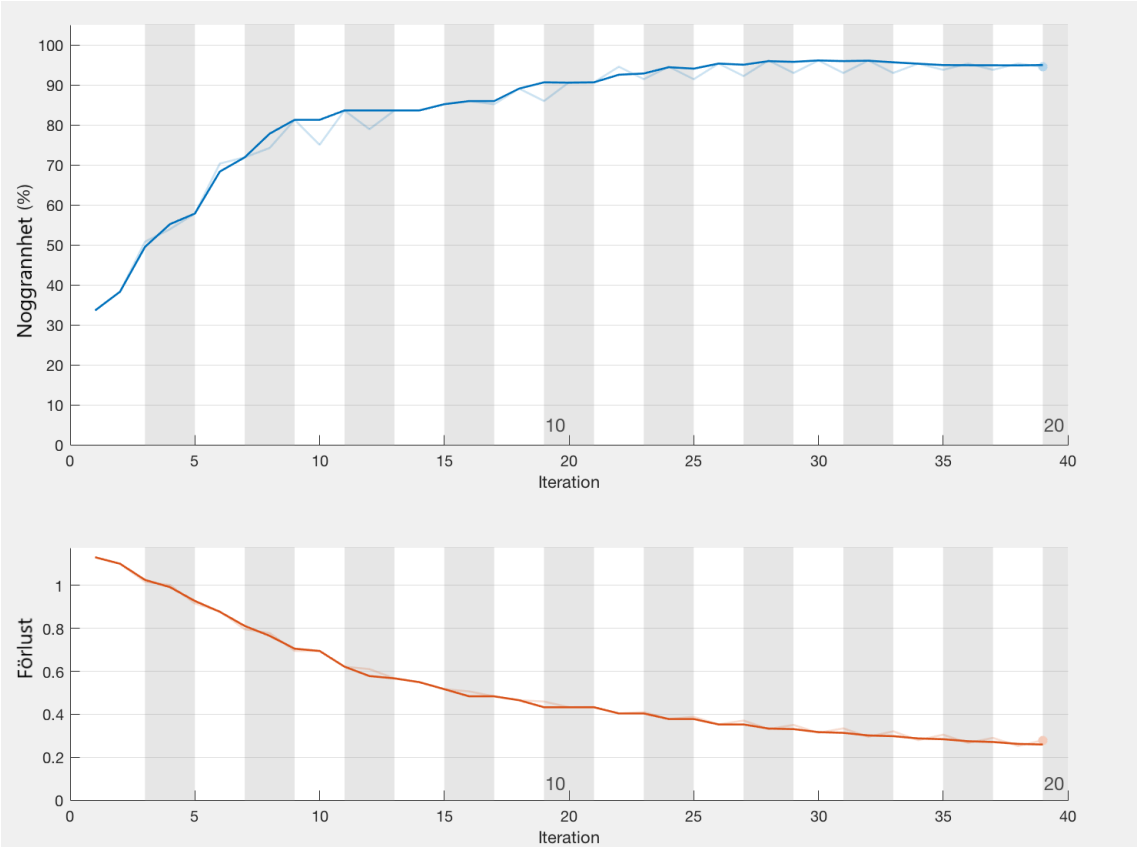
ingen felklassificering. De olika krav och önskemål rörande smutsdetekteringen och hurvida målen uppfylldes listas i *tabell* 4.5.

Tabell 4.5: Resultat av valideringstesterna med avseende på smutsdetektion.

Krav/önskemål	**Uppfyllt mål**
Detektera grus med en kornstorlek på minst 2 mm^3	Ja
Detektera grus under rörelse	Ja
Detektera buntband som smuts	Ja
Detektera vatten	Ej implementerat
Ej detektera bromspår som smuts	Ja
Ej detektera sladdar som smuts	Ja
Ej detektera skuggor som smuts	Ja
Detektera grus på 0,5 m avstånd	Ja
Fungera i varierande ljusstyrka	Ja

4.4 Global navigering

Se figur 4.2 för nätverkets träningsprocess och graf över hur nätverket blir bättre på att klassificera bilder efter varje träningsbild. Det neurala nätverket ger en exakthet på 95 %, det vill säga nätverket klassificerar bilderna rätt med 95 % säkerhet. Vid validering med testbilderna gav nätverket 100 % rätt. Nätverkets klassificering av skyltar var alltså lika exakt för alla typer av testbilder, inkluderat de bilder föreställande de utskrivna skyltarna. Dock klarar inte nätverket att hitta två olika skyltar i samma bild. För att se resultat av önskemål med avseende på global navigering *se tabell* 4.6.



Figur 4.2: Bild över träningsprocessen som illusterar hur nätverket blir bättre på att klassificera efter varje träningbild.

Tabell 4.6: Resultat av önskemålen med avseende på global navigation.

Krav/önskemål	**Uppfyllt mål**
Navigera till specifika områden	Ej testat eftersom ej implementerat i styrsystemet men prototypen kan lära sig känna igen omgivningen.
Ej åka ur verkstaden	Ej testat eftersom det ej är implementerat, dock kan prototypen tränas för att känna igen porten. Både när den är stängd och öppen.
Möjlighet att utnyttja insamlad data för att effektivisera städningen.	Ej implementerat i prototypen men det neurala nätverket hade kunnat tränas för samverkan med smutsdetekteringen.

5

Diskussion

Här diskuteras resultaten med avseende på den uppsatta kravspecifikation. Här diskuteras också de metodval som gjordes under arbetesgång och varför de besluten togs.

5.1 Framtagning av krav och önskemål

Framtagningen av kraven var en mycket tidig och påskyndad del under prototyputvecklingen. Många krav ansattes före fördjupningen inom varje delproblem. Därmed ändrades åsikterna och visionera kring prototypens krav under arbetets gång. Med tiden framgick det att vissa krav var redundanta medans andra egenskaper förbisetts och därmed saknade kravsättning. Dessa egenskaper testades separat från de ursprungliga valideringtesterna som presenterat i *avsnittet* 3.7.6. På grund av tidsbrist implementerades och validerades inte alla krav och önskemål i prototypen men möjligheten finns.

5.2 Framdrivningen

Framdrivningen med Husqvarna Research Platform (HRP) uppfyller samtliga satta krav och även en del önskemål. Att framdrivningen lyckades bli trådlös innebär att prototypen är autonomt självständig och inte behöver ta hänsyn till sladdar under drift. En ej trådlös framdrivning hade kunnat få prototypen att fastna under drift, välta verktyg och störa personalen. Autonoma robotar är robotar som agerar självständigt genom att lösa problem och uppgifter utan mänsklig kontroll. Att personalen då inte behöver ta hänsyn till sladdar eller aktivt övervaka prototypen kan då anses vara ett grundkriterium för att göra en autonom städrobotsprototyp.

5.2.1 Metod kring val av styrkort

Vid valet av mikrodator för styrsystemet så tilldelades först en Beagle Bone Black Industrial (BBB) av Resource for Vehicle Research at Chalmers. Detta märke av mikrodator var också rekommenderat utav Husqvarna, dock i en äldre utgåva. På

Beagle Bones hemsida finns det endast installationsguider för operativsystemet Debian 8 eller 9, varav endast Debian 8 var kompatibel med ROS Kinetic (HRPs mjukvara). Dessvärre var ROS Kinetic med Debian inte kompatibelt ihop med Armhf-arkitekturen på BBB vilket tyvärr upptäcktes sent på grund av dåligt uppdaterade forum på ROS egen hemsida. BBB byttes sedan ut till en Rasberry Pi3 (RPi3) som hade en tydlig och specialformad guide för installation av Ubuntu Mate 16.04 som är kompatibelt med ROS Kinetic och RPi3s Arm7l-arkitektur. De olika testade kombinationerna av mikrodator och operativsystem presenteras i *figur* 5.1.

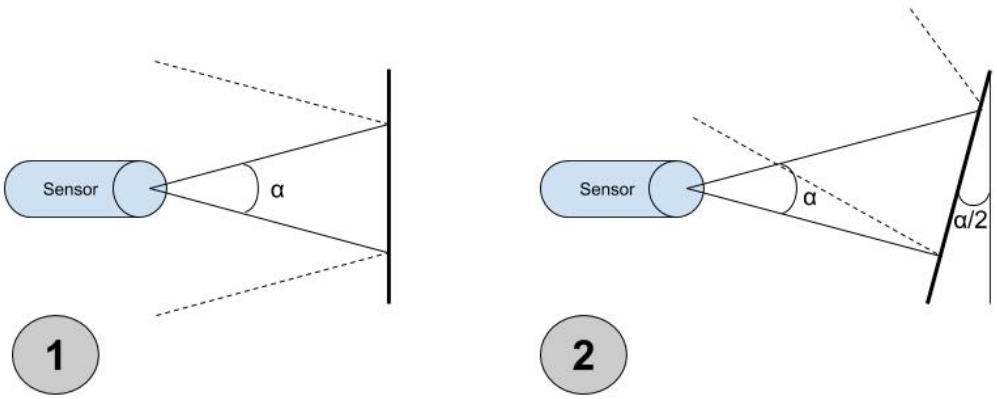
Test	Microdator	Linux OS Version	ROS Version	ARM Version	Kompatibilitet
1	BB Black	Debian 9	Kinetic	Armhf	Nej
2	BB Black	Debian 8	Kinetic	Armhf	Nej
3	Raspberry Pi 3	Rasbian 8	Kinetic	Arm7l	Möjligtvis
4	Raspberry Pi 3	Ubuntu Mate 16.04	Kinetic	Arm7l	Ja

Figur 5.1: Genomförda kompatibilitetsförsök av mikrodatorer och operativsystem.

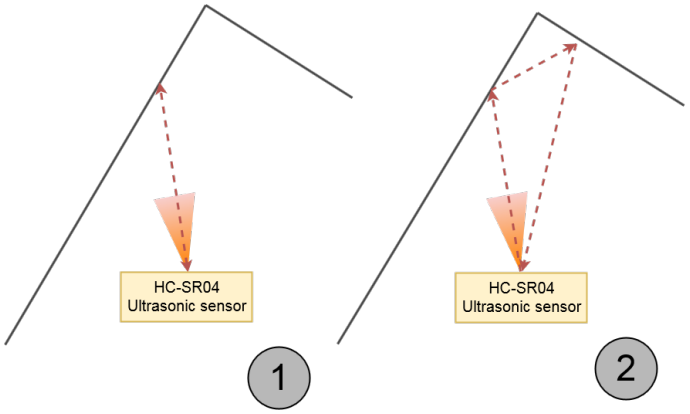
Här hade tid kunnat sparas om noggrannare efterforskning kring mikrodatorers olika arkitekturer och ROS Kinetics kompatibilitet gjorts. Detta var dock ett oväntat problem eftersom tidigare erfarenhet av mikrodatorsarkitekturer och dess betydande roll saknades. Ytterligare förvirring uppstod eftersom ROS-installationen gick problemfritt på både Linux-datorn och Windows med Ubuntu gjorde dessutom att arkitekturproblem verkade vara obefintligt.

5.3 Hinderundvikning

Hinderundvikningen klarade ej av att upptäcka väggar om infallsvinkeln till hindren är över 15°. Detta beror på den maximala infallsvinkeln, det vill säga halva sensorns spridningsvikel, som vid större vinklar inte klarar att fångas upp av sensorn, *se figur* 5.2. Beslutsalgoritmen hos prototypen får då felaktiga mätvärden och kör därmed rakt in i hindren, *se figur* 5.3. Däremot har sensorerna lättare för att upptäcka objekt som människor då ljudvågorna reflekteras på hindret med större spridning eftersom ytan är mer ojämn. Tidigare erfarenhet fanns av HC-SR04 sensorerna där de fungerat väl och problematiken kring infallsvinkeln inte upplevts. Problemet med infallsvinkeln kunde ha undvikits om en noggrannare litteraturstudie hade utförts.



Figur 5.2: Överblick över den maximala infallsvinkeln, α, som är halva spridningsvinkeln. Fall 1 visar en vinkelrät infallsvinkel som reflekteras tillbaka till sensorn. Fall 2 visar en infallsvinkel som är över halva spridningsvinkeln och därmed reflekteras bort från sensorn.



Figur 5.3: Ultraljudet kan vid en sned infallsvinkel studsa vidare i rummet och därmed ge ett fel mätvärde som visar i fall 2, istället för den korrekta distansen som visas i fall 1.

Koden fungerade mycket väl under simulerade tillstånd men blev i valideringstestet *Enkel* och *Svår* svårverifierad på grund av inkonsekventa mätningar orsakade av överskridna infallsvinklar och glapp i sladdar. Ett väggföljningsbeteende har tagits fram men för att implementera det i prototypen krävs vidare tester för att kalibrera programmet med önskat resultat. Tanken är att när prototypen följt väggen längs med hörnen i en viss tid beräknas sedan den totala vinkeln som prototypen har roterat. Om det misstänks att prototypen sitter fast i en slinga larmar den och stänger av sig. Om detta inte är fallet så ska prototypen svänga 90° ifrån väggen och byta körbeteende.

Antagandet om att sensorernas placering skulle kunna hantera överhäng undersöktes inte i valideringstesterna. Frontsensorerna ansågs initialt klara av att upptäcka hinder med prototypens höjd men ingen hänsyn hade tagits till reflektionsmönstret.

Därför stämmer inte påståendet i verkligheten och en ny lösning behövs. Mer avancerade sensorer hade kunnat användas för hinderundvikningen. Dock hade detta inneburit ökade krav av processorkraft för att kunna tolka all sensordata. Därmed ansågs ett mer avancerat val av sensorer överdimensionerat. HRPs kollisionssensor hade, om den hade implementerats i beslutsalgoritmen, kunnat minska stöten vid eventuell krock.

5.4 Smutsdetektion

Smutsdetekteringen uppfyller de satta kraven, *se avsnitt* 4.0.2. Programmet är ett väldigt robust system och gav inga felkalssificeringar i valideringstesterna. Men felresultat kan självfallet uppstå. Felresultat kan uppkomma då smutsdensiteten är mycket låg eftersom programmet inte klarar av att registrera ensamma partiklar som smuts på grund av den inställda känsligheten. Hade enskilda partiklar identifierats som grus hade detta inneburit att även skuggor hade kunnat registreras som smuts vilket inte är önskvärt. Som önskemål fanns det även att upptäcka vatten men under projektets gång valdes det att inte fokusera på detta eftersom det inte var ett högt prioriterat önskemål.

Smutsdetekteringen implementerades inte i dellösning I och II på grund av RPi3s begränsade RAM-minne och tidsbrist. Det är tveksamt om RPi3 beräkningskraft hade klarat av bildanalys. Dock om en trådlös kamera använts hade smutsdetektionen kunnat köras på en separat dator och därefter kommunicerat beslut till ROS på RPi3. Alternativt hade en dator kunnat monteras på prototypen.

5.5 Global navigation

Önskemålet om att prototypen ska kunna navigera till ett specifikt område är inte uppfyllt. Men genom att sätta upp skyltar i olika områden av lokalen hade prototypen kunnat få kändedom om sin position. En zonuppdelning hade kunnat genomföras i lokalen där varje symbol representerar ett område i labbet. Detta kräver modifiering av lokalen och på grund av tidsbrist implementerades inte det neurala nätverket i prototypen utan det neurala nätverket simulerades, testades och validerades separat från dellösningarna I-III och dess valideringstester.

Att träna nätverket på andra objekt än skyltar är i nuläget möjligt om träningsdatan införskaffas för varje objekt som skall kännas igen. Nätverket behöver minst 100 unika träningsbilder per objekt för att ge ett gott resultat. För att få ett mer exakt nätverk behövs fler bilder att träna på, mer än 1 000 bilder per kategori är önskvärt. Det neurala nätverket kunde endast garantera en precision på 95 % på grund av att mängden testbilder som gavs till nätverket.

Om implementerad i prototypen hade det neurala nätverket kunna samverka med de andra dellösningarna. Prototypen skulle vid givna områden kunna registrera hur

mycket och hur ofta smuts detekteras för att sedan prioritera navigering till de områden i industrilokalen som oftast är smutsiga. Nätverket hade också kunnat tränats för att känna igen smuts och utefter det optimera navigationen och drifttiden ytterligare.

Kravet att prototypen inte skall åka ut ur fordonslabbet stora garageport testades inte eftersom det globala nätverket inte implementerades i den färdiga prototypen. Nätverket hade dock kunnat tränas för att känna igen garageporten och därmed undvika köra ut ur verkstaden.

I början av projektet fanns en tanke på att göra ett färgigenkänningsprogram i Matlab som metod för global navigering. För att få programmet applicerbart i praktiken hade en modifiering av lokalen genom målning av olika färgzoner krävts. Detta program fungerade dock inte bra vid testning eftersom att lamporna och positionen i lokalen ger stora förändringar i färg och kontrast i bilderna. Därför valdes det att inte arbeta vidare på färgigenkänningsprogrammet utan att istället fokusera på det neurala nätverket.

6

Slutsats

Efter genomförandet av de fyra dellösningarna; framdrivning, hinderundvikning, smutsdetektering och global navigering dras följande slutsatser:

- Prototypen kan framdrivas trådlöst och autonomt.
- Prototypen kan oftast undvika hinder med krockar ibland in i saker.
- Mängden sensorer bör utökas.
- Prototypens sensorer kan inte detektera objekt med infallsvinkel på över 15°.
- Prototypen detekterar grus och buntband som smuts med mycket hög precision.
- Det neurala nätverkets bildklassificering fungerar med mycket hög precision.
- Det neurala nätverket för global navigering fungerar men behöver vidare implementering i prototypen och lokalen.
- Mikrodatorn är inte dimensionerad för att köra bildanalysprogram utan detta måste köras separat.
- All mjukvara kan utvecklas separat och samverka tillsammans via ROS.
- Prototypen har möjlighet att utvecklas vidare av framtida projekt.

Prototypen tillgodoser sitt syfte trots vissa brister i utförandet.

7

Förslag till framtida projekt

Här diskuteras förslag till framtida utveckling av städprototypen med avseende på delproblemen I-IV. Här diskuteras möjliga framtida metodval för att optimera prototypen.

7.1 Framdrivning

Använd den rekommenderade mikrodatorn och operativsystemet från denna rapport som styrkort, nämligen RPi3 ihop med Ubuntu Mate 16.04. Denna fungerande kompabilitetskombination är svår att hitta på nätet och för att bespara framtida projektgruppers tid rekommenderas det att köra på det konceptet som bevisats att fungera här. Bildigenkänningsprogram rekommenderas dock inte att köras på mikrodatorn. Detta bör köras på en dator med mer processorkraft. Det är möjligt för mikrodatorn att ta in information från en redan färdigt exekverad bildanalys men den bör inte köras lokalt på kortet.

7.2 Hinderundvikning

För framtida projekt rekommenderas ett inköp av ytterligare ultraljudssensorer för att eliminera problemet med infallsvinklarna. Med fler sensorer minskar sannolikheten för att objektet med 15° infallsvinkel från alla sensorer. HRPs krocksensorer bör även integreras i beteendet för att säkerhetställa att alla hinder upptäcks och minimera risken för skada om prototypen krockar.

7.3 Smutsdetektion

I kravspecifkationen finns det ett önskemål om att prototypen ska kunna navigera med hjälp av smutsdetektionsprogrammet för att minimera städtiden. Funktionen implementerades inte men det aktuella smutsdetektionen kan upptäcka smuts och ange vart i bilden programmet upptäckte smutsen. Det skulle då vara möjligt att skapa en ROS-nod för att skicka vidare informationen till prototypens-styrsystem

som kan använda informationen för att påverka prototypens körbeteende. Genom att implementera en lampa i prototypen som lyser när smuts detekteras hade återkopplingen till användaren blivit mer lättillgänglig.

7.4 Global navigering

För att utveckla den globala navigeringen skulle det neurala nätverket kunnas tränas på objekt i rummet och ihop med HRPs pulsgivare räkna ut exakt position. En ROS-nod skulle kunna skapas innehållande det neurala nätverket i Matlab för att möjliggöra implementeringen av det globala navigationssystemet ihop med prototypens styrsystem.

För att se till att prototypen inte åker ut genom garageportarna i lokalen skulle det globala positioneringssystemet kunna tränas för att signalera om prototypen är vid garageporten. Alternativt kan en begränsningsslinga användas. Det är en kabel som triggar HRPs gränsslingessensorer och därmed vet prototypen vart gränserna går på samma sätt som en robotgräsklippare upptäcker vart den ska svänga.

Litteraturförteckning

[1] G. A. Bekey, "Overview: Autonomous robots: From biological inspiration to implementation and control," 2018-01-26, (https://mitpress.mit.edu/books/autonomous-robots, Hämtad: 2018-01-28).

[2] "Välja skurmaskin," , (https://www.bimo.se/category/valja-skurmaskin, Hämtad: 2018-01-30).

[3] "Skurmaskin," 2017-10-30, (http://www.xn--stdamiljvnligt-6hbh81a.se/skurmaskin, Hämtad: 2018-01-30).

[4] D. Forouher, "Topics," 2016-05-04, (http://wiki.ros.org/Topics), Hämtad: 2018-04-08).

[5] G. Hoang, "Documentation," 2017-03-31, (http://wiki.ros.org/, Hämtad: 2018-04-08).

[6] T. Foote, "Husqvarna research platform," 2016-05-04, (http://www.ros.org/news/2016/05/husqvarna-research-platform.html), Hämtad: 2018-04-04).

[7] Husqvarna, "Robotgräsklippare automower 430x," 2016-05-04, (https://www.husqvarna.com/se/produkter/robotgrasklippare/automower-430x/967622521/), Hämtad: 2018-04-04).

[8] "Vad är en ultraljudssensor?," 2016-10-08, (http://sv.nous-utile.info/article/vad-ar-en-ultraljudssensor, Hämtad: 2018-05-07).

[9] J. Philipsson, J.-A. Dahlström, J. Malmquist, L. Grahm, and R. Hall, "Ultraljud," , (https://www.ne.se/uppslagsverk/encyklopedi/l%C3%A5ng/ultraljud, Hämtad: 2018-05-07).

[10] "Edge detection," , (https://se.mathworks.com/help/images/edge-detection.html, Hämtad: 2018-04-18).

[11] J. Canny, "A computational approach to edge detection," , (https://pdfs.semanticscholar.org/55e6/6333402df1a75664260501522800cf3d26b9.pdf, Hämtad: 2018-05-07).

[12] P. G. Christian Balkenius, Jonas Skeppstedt, "artificiell intelligens," , (https://www.ne.se/uppslagsverk/encyklopedi/lång/artificiell-intelligens, Hämtad: 2018-04-18).

[13] R. Vendel, "Skillnaden på artificiell intelligens, machine learning och deep learning," 2017-03-30, (https://www.ucit.se/blogg/skillnad-ai-artificiell-intelligens-machine-learning-deep-learning, Hämtad: 2018-04-19).

[14] O. Enqvist, "Lecture notes in image analysis," , Hämtad: 2018-04-19).

[15] M. A. Nielsen, "Neural networks and deep learning," 2015, (http://neuralnetworksanddeeplearning.com/chap1.html, Hämtad: 2018-05-04).

[16] "Ultrasonic ranging module: Hc-sr04," 2010-11-03, (https://file.m.nu/pdf/HC-SR04.pdf, Hämtad: 2018-05-09).

[17] ModMyPi, "Ultrasonic ranging module: Hc-sr04," 2014-7-03, (https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi, Hämtad: 2018-05-12).

A

Planritning över fordonslabbet