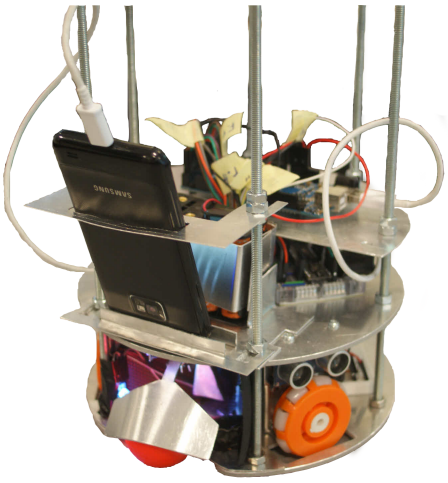
CHALMERS

KANDIDATARBETESRAPPORT



# Robotfotboll

*Slutrapport*


Joakim Berg
Mike Demant
Britta Källman
Michael Nordström

Forskargrupp
Institutionen för signaler och system
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, 2014


SSYX02-14-23

## Abstract

The basis of this project was to create a robot playing soccer from existing robot soccer rulesets. The goal was for the robot to autonomously identify the ball, carry it towards the goal and score. This was achieved by processing visual data through object detection and having the robot perform its given task accordingly. Two seperate areas were developed parallelly, the construction of the robot and the programming of the object detection software. The robot was built from scratch using commercially existing circuit boards and engines. The object detection software was based on the software library OpenCV and is used on an Android unit. An Arduino was used as controller for the hardware components and communicated with the Android unit through a USB-connection to complete the autonomous system.

At the end of the project the robot was able to locate the ball and catch it, carry it towards the goal and finally score according to the set goals.

## Sammanfattning

Detta projekt behandlar utvecklingen av en fotbollsspelande robot, vilken är designad utifrån existerande regelramverk för robotfotboll. Målet var att roboten autonomt skulle finna bollen, ta den till och skjuta den i mål. Detta gjordes genom objektidentifiering på roboten som behandlar visuell data och utifrån den tog egna beslut för att utföra uppgiften. Först skedde två utvecklingar parallellt, konstruktionen av roboten och programmering av objektidentifiering. Konstruktionen av roboten skedde från grunden med kommersiellt existerande kretskort och motorer. Objektidentifiering programmerades baserat på mjukvarubiblioteket OpenCV och utfördes på en Android-enhet. En Arduino användes som kontrollenhet för resterande hårdvara och kommunicerade med Android-enheten via USB-uppkoppling för att färdigställa det autonoma systemet.

Vid projektets slut kunde roboten enligt uppsatta mål lokalisera boll, fånga denna sedan lokalisera och skjuta bollen i mål.

# Innehåll

# 1

# INLEDNING

## 1.1 Bakgrund

Robotar i samhället utvecklas ständigt. De förnyas, förändras och återskapas i olika former och skepnader. Detta ger såklart en påverkan på miljön, vilket nödvändigtvis inte är negativt. Att tillverka nytt kräver en viss resursåtgång, men är i många fall nödvändigt och kan bidra till något positivt i slutändan.

Robotar är idag viktiga verktyg gällande bland annat att utföra arbeten som kan vara farliga för människan. Exempelvis i ett gjuteri riskeras att få direktkontakt med farliga lösningsmedel och substanser som krävs för processen. Detta kan undvikas med hjälp av robotar eftersom de i många fall kan ersätta detta arbete.

Objektidentifiering är ett annat viktigt användningsområde inom tekniken. Att robotar eller någon maskin kan identifiera olika objekt används idag inom bland annat sopsortering. Ur miljösynpunkt är detta betydelsefullt eftersom olika material i komponenter enklare tas till vara och på så vis kan återanvändas eller återvinnas i så stor grad som möjligt. Detta medför i sin tur att nytt material ej behöver tillverkas samt att en mindre mängd substanser behövs lagras på en soptipp i brist på bättre sätt att ta hand om avfallet.

Autonoma robotar har fortfarande mycket utvecklingspotential och det är en utveckling som bör fortgå, då de kan vara till stor hjälp inom många områden. Dessa skulle kunna ersätta människor i monotona sysslor eller miljöer som är riskfyllda för människan att arbeta i. Dessutom minskar risken för fel på grund av mänskliga faktorer. Den fortsatta utvecklingen kräver nya idéer och perspektiv, vilket medför att nytt intresse måste skapas.

För att fler människor skulle intressera sig för robotar anordnades år 1996 en fotbollsturnering av FIRA, *Federation of International Robot-soccer Association*, för första gången. FIRA Cup är en turnering för robotar och har sedan starten vuxit och blivit världsomfattande. Målet är att engagera yngre personer inom vetenskap och teknik genom fotbollen. Detta är viktigt för att hela tiden få ny stimulans så att utvecklingen fortsätter framåt.

## 1.2 Syfte

Detta kandidatarbete skapades för att få ökat intresse kring teknik. Projektet har som mål att bygga en robot som kan spela fotboll, där syftet är att lära sig att kontrollera och styra roboten genom att utveckla enkel objektidentifiering. Roboten ska byggas med kommersiellt existerande komponenter för att underlätta för den som vill kopiera eller vidareutveckla resultatet och på så vis fortsätta FIRAs mål att engagera människor i teknik.

Projektet utförs utav fyra studenter under en termin och omfattade 15 högskolepoäng. Till förfogande finns en budget 5000 kronor att användas på valfria kostnader samt ytterligare 2000 kronor för metall-, plast- och trämaterial.

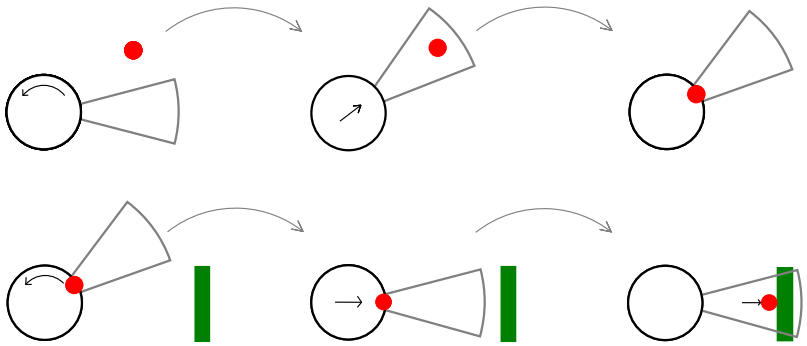
# 2

# PROBLEM

Målet med projektet var att skapa en robot som kan spela fotboll, där FIRA:s regelramverk [1] användes som grund för problemdefinitionen. FIRA arrangerar flera olika turneringar med separata regelböcker, där projektets begränsade tidsram och erfarenhet i gruppen gjorde att den kategori som kallas RoboSot [2] är mest passande. I RoboSot ingår robotar som är upp till 0.35x0.35 meter (bredd och djup) stora, obegränsat höga och antingen autonoma eller semiautonoma genom att den visuella analysen, som identifierar bland annat bollen, sköts av en värddator. Robotar möts i matcher en mot en, där den enda interaktionen mellan människa och robot är att roboten startas. För att robotarna ska klara detta måste de vara försedda med ett program som instruerar vad roboten ska utföra. Dessa program har genomgående ett generellt huvuddrag (se Figur 2.1) - de följer en grundläggande operationslista [3] [4] [5]:

1. Finna bollen.
2. Färdas till och ta kontroll över bollen.
3. Finna målet.
4. Få bollen in i målet.

Denna listan användes som definition av vad spela fotboll innebär och var målet för vad roboten skulle uppnå.



**Figur 2.1:** Robotens funktion förenklad. Första raden visar robotens som söker bollen. Andra raden visar roboten som funnit boll och ska skjuta den i mål. Den gråa cirkelsektorn representerar kamerans synfält.

# 3

# PROBLEMBEGRÄNSNING

Begränsning utav problemet gjordes utefter RoboSot reglerna, inspiration från tidigare tävlingsdeltagare [3] [4] [5] samt andra existerande robotar [6] [7] [8]. Det ansågs vara fördelaktigt då FIRA hållt tävlingar sedan år 1996 [1] med många olika robotkonstruktioner samt att det passade bättre med projektets tidsschema. Begränsningarna infördes på robotens styrning och rörelsekaraktär, samt på planen och bollen.

## 3.1 Robotens styrning

Roboten behöver styrkommandon för att veta vad den ska göra härnäst. Detta görs i FIRA:s tävlingar genom att roboten är datorstyrd, där en separat dator fjärrstyr roboten, eller autonom, att roboten är självständig.

I fallet datorstyrning monteras en kamera ovanför spelplanen vilken är uppkopplad till en dator. Denna kamera ger ett fågelperspektiv över spelplanen och kan via färgkoder på objekt identifiera positioner av robotar, boll och mål. Datorn behandlar den visuella datan och sänder information och kommandon till robotarna i spel. Det andra alternativet är en kamera monterad på roboten som i realtid läser av kamerans synfält och skickar informationen till roboten via en mikrokontroller, även denna monterad på roboten. En mikrokontroller är en liten dator som kan programmeras samt ta emot och skicka ut signaler till komponenter kopplade till det. Identifikation av objekt sker även i det andra alternativet via färgidentifiering.

Datorstyrning är det bästa tillvägagångssättet för ett större projekt som involverar lag med flertalet robotar som ska samarbeta i en strategisk realtidsmiljö. Detta är till följd av den ökade mängden av information som ska behandlas då det finns fler robotar att beakta, vilket kräver starkare processorer. Tidsbegränsningar gör alternativet mindre utförbart då det blir ett bredare projekt som täcker flera områden istället för ett projekt fokuserat på roboten. Att bygga en fullständigt autonom robot är det lämpligaste projektet, därmed valdes alternativet kamera på robot.

Styrningen löses med en Android-enhet och en Arduino. Android-enheten är en mobiltelefon med Android som operativsystem, till vilket mycket information om programmering finns tillgänglig, av modell Samsung Galaxy S2. Den fyller de nödvändiga kriterierna:

- Har en kamera.
- Har en kraftfull processor för projektets ändamål.
- Finns tillgänglig, belastar därmed inte budgeten.

Enhetens kamera nyttjas av roboten för att ta in visuell information som den skickar vidare till Arduinon, som är ett mikrokontrollerkort. Arduino plattformen används ofta i hobbybyggen och dess populäritet gör att det finns mycket information tillgänglig för att programmera denna.

## 3.2 Robotens rörelsekaraktär

I FIRA:s tävlingar finns det två använda metoder för rörelse hos robotarna: hjul och ben. Efter studie av de två metoderna valdes att roboten skulle röra sig med hjälp av hjul till följd av komplexiteten i att bygga och styra artificiella ben. Detta minskar svårigheten att skapa ett fungerande system som gör att roboten kan röra sig och tillåter således att större arbete läggs på programmering av robotens autonoma styrning. Studierna baserades på videor från FIRA:s tävlingar [3] [4] [5].

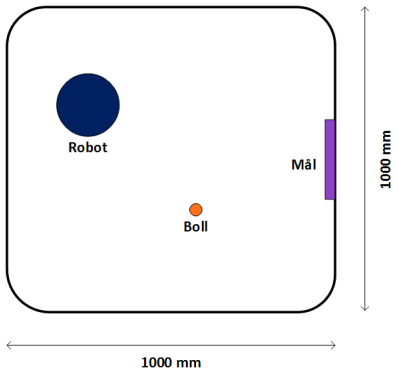
## 3.3 Bollens egenskaper

Det viktigaste kravet på bollen var att den var monokrom med en tydlig färg då roboten identifierar objektet med hjälp av denna. Bollen ska vara både mindre och lättare än roboten samt standardiserad för att underlätta tillgänglighet. Golfboll valdes då de:

- är små,
- väger mycket mindre än roboten,
- har låg friktionskoefficient,
- är standardiserade med varierande färger.

## 3.4 Planens egenskaper

Spelplanen (se Figur 3.1) behöver endast ett mål på grund av att det bara är en robot och därmed inte finns något syfte med två mål. Till spelplanen behöves även en sarg för att förhindra att bollen rullar iväg. Sargen måste vara i en färg som inte påverkar robotens förmåga att identifiera viktiga objekt på planen samt att den behöver rundade hörn för att bollen ej ska fastna i ett oåtkomligt läge för roboten som resultat av kollisionssensorerna. Målet identifieras via en specifik igenkännlig färg som ej är samma som bollen.




**Figur 3.1:** Spelplanen

## 3.5 Krav på robot

Med projektets ramar bestämda behövdes målet klargöras, varför en kravspecifikation skapades (se Bilaga A). Här följer en lista med huvudkraven:

- Identifiera boll efter färg.
- Färdas till bollen.
- Transportera bollen.
- Skjuta iväg bollen.
- Åka rakt fram, med max 15 graders avvikelse.
- Rotera önskat antal grader, med max 22 procent avvikelse.
- Hålla projektet inom budgetens ramar.

I kravspecifikationen finns även önskemål om mer precist utförande av vissa funktioner (som att rotera önskat antal grader med hög precision) och att roboten ska kunna hålla en hög hastighet. För att roboten ska anses uppfylla sin funktion måste alla krav vara uppfyllda.

# 4

# TEORI

Detta kapitel behandlar viktiga teoriområden, som berörs i samband med projektet, för att få en bättre förståelse för beslutsfattande och genomförande inom projektet.

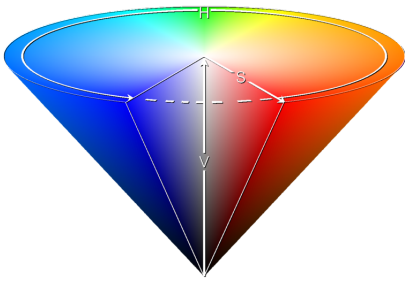
## 4.1 Objektidentifiering

Det finns ett antal olika metoder för identifiering av objekt i en datormiljö, exempelvis kantidentifiering och blob detection [9]. Det första alternativet behandlar identifiering via former medan det andra urskiljer objekt med olika karaktär som ljusstyrka eller färg. Projektet nyttjar det senare alternativet. Att identifiera och följa ett objekt baserat på dess färg kräver att färger skilda från objektets effektivt kan filtreras bort. I projektet utförs detta arbete i Android-enheten.

Den standardiserade färgmodellen RGB (Röd, Grön, Blå) är av additiv karaktär där de tre grundfärgerna adderas i olika styrka för att representera ett brett spektrum av färger [10]. Korrekt filtrering av mycket specifika kulörer blir svår med denna modell då resultantkulören beror på alla tre parametrar samt styrkan på ljuset som belyser objektet. Ett system där endast en parameter definierar kulören är att föredra då endast den parametern behövs ställas in för att finna objektet med specifik färg.

### 4.1.1 NMI-systemet

Ett väl använt alternativt färgsystem är NMI-systemet (engelska HSV) vars parametrar Nyans, Mättnad och Intensitet är en representation av RGB-modellen i cylindriska koordinater (se Figur 4.1) [12]. Fördelen med detta systemet är att enbart nyansparametern bestämmer kulören vilket således gör att en specifik färg enkelt kan passera ett filter genom en övre och undre gräns på nyansvärdet (så kallad thresholding). Mättnad




**Figur 4.1:** Visuell representation av NMI-systemet [11].

och intensitet filtreras på liknande sätt för att fokusera filtret för att överensstämma med objektets specifika färg. En komplikation med NMI-systemet är att olika applikationer sällan använder samma implementering vilket medför att gränsvärdena för de tre parametrarna varierar. Det finns således ingen universal standard för vilka färger som representeras av vilka värden.

### 4.1.2 Thresholding

Thresholding är en teknik som används för att segmentera en bild [13]. Inom objektidentifiering används thresholding på binära bilder, där pixlarna enbart antar värdet noll eller ett, för att separera bilden i två segment: förgrunden (objektet som ska identifieras) samt bakgrunden (resterande bild). Förgrunden representeras av vit färg medan bakgrunden blir svart vilket möjliggör lokalisering av objektet. För att få önskat objekt i förgrunden appliceras ett filter på NMI-representationen av en bild, liknande ett bandpassfilter, som endast släpper igenom nyans-, mättnads- samt intensitetsvärden mellan två givna gränser. När den filtrerade bilden omvandlas till binär är det de genomsläppta värdena som placeras i förgrunden.

## 4.2 OpenCV

OpenCV (Open Source Computer Vision Library) är ett mjukvarubibliotek med öppen källkod vars huvudområde är datorseende och maskininlärning [14]. OpenCV konstruerades för att skapa en gemensam infrastruktur för applikationer som bygger på datorseende och är BSD-licensierad vilket möjliggör att organisationer och företag enkelt kan nyttja och även modifiera biblioteket. Att vara BSD-licensierad innebär att mjukvaran får kopieras eller modifieras och därefter säljas.

Biblioteket har mer än 2500 optimerade algoritmer, som inkluderar en omfattande uppsättning av de främsta både klassiska och moderna algoritmerna inom datorseende och maskininlärning.

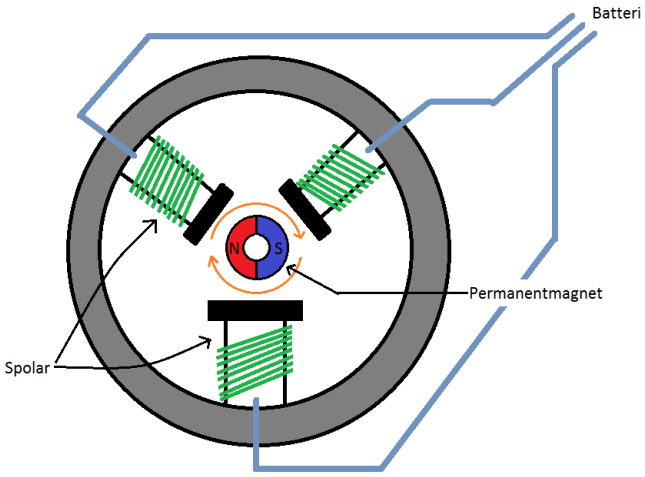
Typiska användningsområden av OpenCV:

- Identifiera föremål
- Ansiktsigenkänning
- Extrahera 3D-modeller av föremål
- Mobila robotar

OpenCV har C++, C, Python, Java och MATLAB-gränssnitt och stöd för Windows, Linux, Android och Mac OS [14].

## 4.3 Arduino

Arduinon är ett mikrokontrollerkort med öppen kretsddesign [15]. Ett mikrokontrollerkort är en dator i en liten integrerad krets innehållandes en processor, ett minneskort och




**Figur 4.2:** Borstlös motor med tre spolar jämt placerade runt statorn.

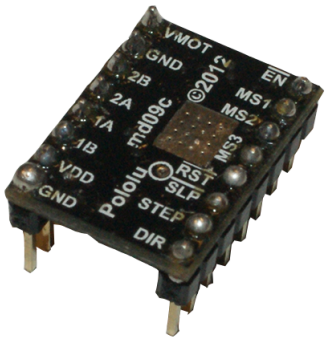
programmerbara in- och utportar. Arduinons mjukvara sköter kommunikationen mellan de elektriska komponenterna så som motorer och sensorer.

## 4.4 Motorer och drivkort

Likströmsmotorer brukar kategoriseras i två typer, borstad motor och borstlös motor [16]. Generellt för båda typerna är att rotation uppnås genom skillnader i magnetisk polaritet hos rotorn och statorn. Rotorn är den roterande delen i en motor vars mittpunkt består av den axel som används för att överföra den genererade kraften. Statorn är den statiska delen som omsluter rotorn. Beroende på typen av motor konstrueras dessa på olika vis.

### 4.4.1 Borstlös motor

I en borstlös motor konstrueras rotorn som en magnet medan statorn har spolar utplacerade runt om (se Figur 4.2) [16] [17]. Rotationskaraktären för en borstlös motor är att ett helt varvs rotation är uppdelat i ett fixt antal steg med specifik stegvinkel. Stegning uppnås genom att spolarna aktiveras i sekvensiell ordning vilket resulterar i att rotorn hela tiden attraheras till nästa spole. Behovet av att kontrollera aktivering av spolarna i sekvensiell ordning innebär att extra kretslogik i form av ett drivkort (se Figur 4.3) samt ett kontrollsystem behövs. Drivkorten tar emot steg- och rotationspulser från kontrollsystemet och konverterar dem till elektriska pulser som strömför rätt spole vid rätt tidpunkt. En stegpuls krävs för varje steg-förflyttning av motoraxeln. Ytterligare funktionalitet hos vissa drivkort är möjligheten att påtvinga mindre stegvinkel än vad motorn är konstruerad för. Detta uppnås genom att flera spolar strömförs samtidigt med olika strömstyrkor, vilket medföljer till olika styrkor på de magnetiska fälten som attraherar rotorn.




**Figur 4.3:** Drivkort

### 4.4.2 Stegmotor och servomotor

En stegmotor bygger på tekniken av en borstlös motor [16]. Antal steg per hel rotation är vanligtvis 200 steg med stegvinkel på 1.8 grader.

Servomotorer är ej begränsade till motortyp [16]. Beroende på prisklass av servomotorn används antingen borstad eller borstlös typ. Detta möjliggörs då en extra positionssensor är installerad inuti som mäter motoraxelns vinkelposition. Detta medför en ständig kunskap om hur mycket axeln har roterat från utgångsläget.

## 4.5 Ultraljudssensor

En ultraljudssensor upptäcker och beräknar avståndet till föremål inom dess räckvidd genom att sända ut ljudvågor och mäta tiden det tar för ekot att komma tillbaka till sensorn [18] [19]. Då mätningen sker i luften är det viktigt att det inte finns några störningar så som ventiler eller objekt med höga temperaturer som virvlar runt luften.

## 4.6 Fotoresistor

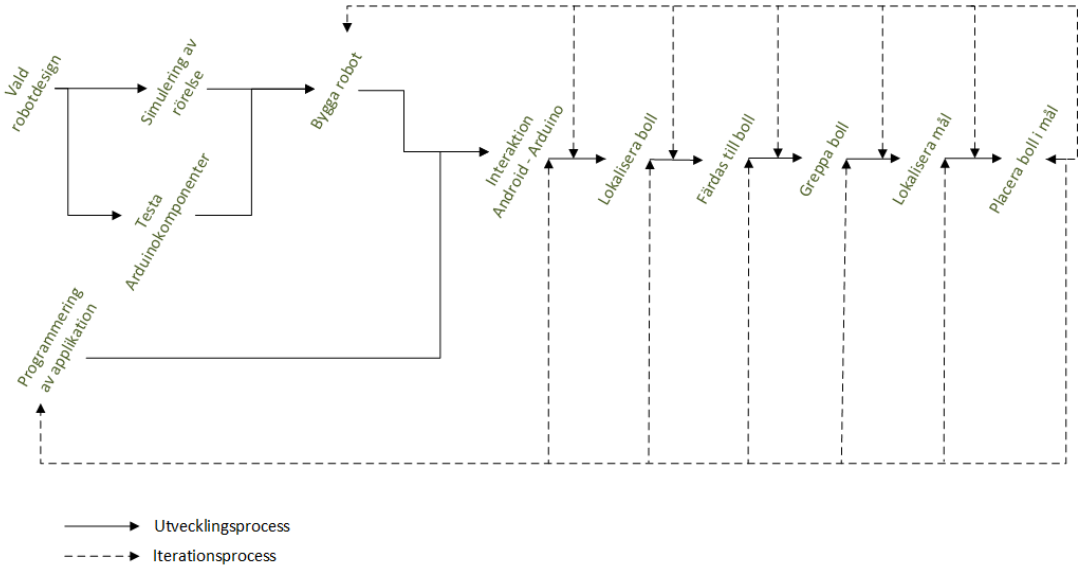
En fotoresistor är ljuskänslig vilket innebär att minskad ljusintensitet framför resistorn resulterar i ökat motstånd [20]. På detta sätt kan det identifieras om något föremål finns framför komponenten genom att det skyler sensorn så att mindre ljus når fram till den.

# 5
# METOD OCH UTFÖRANDE

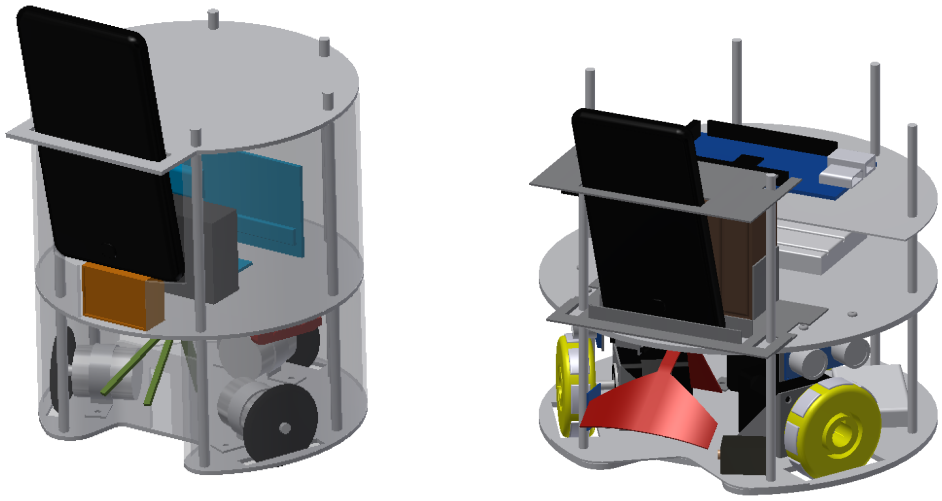
Detta kapitel behandlar hur robotens hård- och mjukvara utvecklades enligt den iterativa processen illustrerad i Figur 5.1. Problemet delades upp i två mindre bitar, robotskelettet och dess design (inkluderat val av komponenter) samt programmering utav Android-enhet och Arduino. Dessa delar fördes sedan samman och utvecklades i sin helhet allt eftersom problem uppstod.

Den iterativa processen karaktäriseras utav att små framsteg görs, testas och omvärderas varpå resultatet används för att både förbättra tidigare och framtida steg. Denna typ av utveckling är vanlig inom mjukvaruutveckling där arbete och tid som krävs för att utföra tester med olika parametrar är mindre. Metoden fungerade väl för projektet då stor del av arbetet som utfördes var programmering, samt att det gav gruppen möjlighet att erhålla ny kunskap som användes för att förbättra tidigare resultat.

Projektet inleddes med att göra en budgetkalkyl för att få en uppskattning om vilken summa som beräknades att användas. För inköp av komponenter beräknades en kostnad på 2825 kronor, vilket ligger inom ramarna för projektet med god marginal. Att ungefär 60 procent utav den totala tillgängliga budgeten beräknades användas berodde på att gruppen ansåg det viktigt att projektet inte skulle påverkas negativt av oförutsedda kostnader. De komponenter som beställdes beräknades uppfylla alla krav (se Bilaga A) som ställts på roboten, samt förhoppningsvis även önskemålen.




**Figur 5.1:** Flöde över projektets arbetsgång.



**(a)** Initiella idédesignen. **(b)** Nuvarande designen på prototypen.

**Figur 5.2:** Designens utveckling med projektets gång.

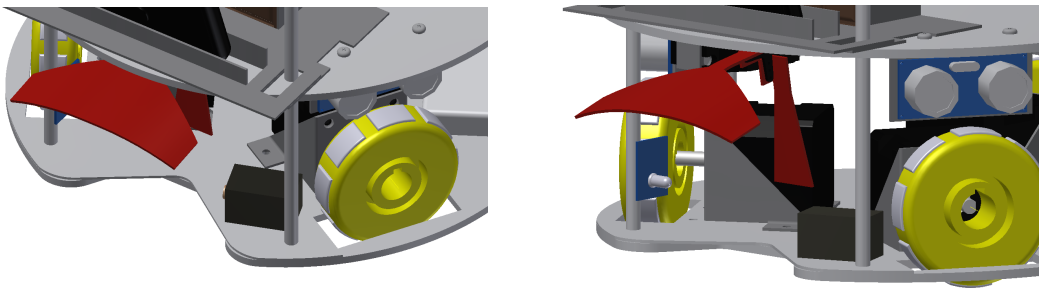
## 5.1 Robotens design

Val av hjul gjordes för att få så stor rörlighet som möjligt. Beroende på antalet hjul och deras placering var det naturligt hur robotens form skulle designas för att passa bra med dem samt få en så stor yta som möjligt för montering av komponenter, utan att robotens omkrets blir onödigt stor. De olika komponenterna monteras för att få en viktmässigt balanserad robot, att de som behövs kommas åt mellan användning blir lätt tillgängliga samt att kopplingssladdarna ska gå enklast möjliga väg. Figur 5.2a visar idédesignen som skapades i den initiella fasen av projektet, medan Figur 5.2b representerar CAD-modell (Computer-Aided Design) av robotens nuvarande design .

### 5.1.1 Bollhantering

I bottenplattan finns en inbuktning (se Figur 5.3a) där bollen kan hållas, med syftet att få bollen att rulla in även om den kommer snett mot roboten. Inbuktningen bör ha rätt bredd, vinkel och djup för att fylla sin funktion på bästa sätt. Efter tester visade det sig lämpligt att ha en inbuktning som är smal längst in för att bollen naturligt ska hamna där, samtidigt som det var viktigt att behålla bredden längst ut, för att få så stor infallsvinkel som möjligt.

För att roboten ska kunna göra mål behövde en skjutanordning designas. Bollen ska skjutas rakt och så hårt som möjligt samtidigt som anordningen ska passa med robotens övriga design. En ”sked” (se Figur 5.3b) som styrs av en motor och fälls ner ovanpå och låser bollen konstruerades. Till det skapades även en ”paddel” som skjuter iväg bollen. Denna konstruktion av skjutanordningen uppfyller både ivägskjutningsfunktionen samt



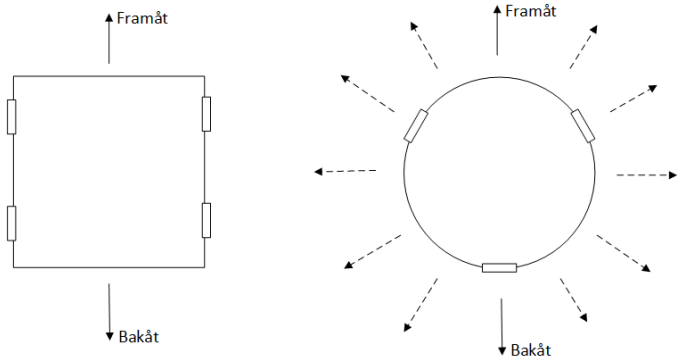
**(a)** Inbuktning **(b)** Sked samt paddel

**Figur 5.3:** Detaljerade CAD-modeller.

att den håller fast bollen när den fångats, med hjälp av endast en motor. Gällande skeden var det viktigt att tänka på längd, bredd och hur den var formad. En form som liknar golfbollen så mycket som möjligt, men som samtidigt hjälper till att fösa in bollen i inbuktningen vid infångning var det som ansågs mest effektivt. Skeden gjordes om flertalet gånger för att hitta en form som uppfyller funktionskraven.

### 5.1.2 Rörelse

Hjul placerade parallellt med körriktningen ger en snabbare robot som kräver mer kraft för att kunna svänga på stället än om hjulen istället placeras i ett cirkulärt mönster. I det senare fallet fås en robot med lägre hastighet men fler accelerationsriktningar (se Figur 5.4) samt en bättre kontroll över rotationen kring egen axel, dock kräver det att omnihjul används. Ett omnihjuls omkrets består av små hjul som roterar vinkelrätt mot hjulets rotationsriktning (se Figur 5.5). Detta motverkar att hjulet släpas då roboten inte färdas i hjulets riktning. Det cirkulära mönstret utgörs av tre eller fyra hjul för optimal styrning. Tabellen nedan (se Tabell 5.1) visar på fördelar och nackdelar med tre respektive fyra hjul enligt De Witte [21].




**Figur 5.4:** Riktningar som roboten kan accelereras i för hjul parallellt med körriktning respektive hjul placerade i cirkulärt mönster.

**Figur 5.5:** Omnihjul

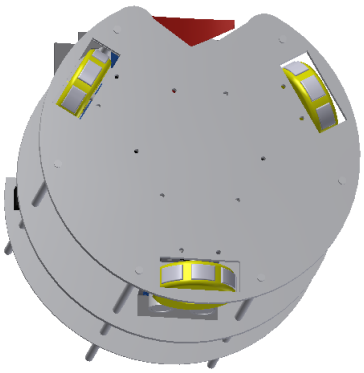
| Tre hjul | |
|---|---|
| Fördelar | Nackdelar |
| - Mindre komplex att implementera och kontrollera<br>- Billigare<br>- Alltid kontakt med marken | - Sämre grepp<br>- Mindre kraftfull |
| Fyra hjul | |
| Fördelar | Nackdelar |
| - Mer plats för bollhanteringsannordning<br>- Bättre grepp kan åstadkommas<br>- En fjärde motor medför att mer kraft kan tillföras | - Dyrare<br>- Svårare att implementera och kontrollera<br>- Mindre plats tillgänglig generellt<br>- Kontakt mellan fyra hjul och markytan ej garanterad |

**Tabell 5.1:** Fördelar och nackdelar med tre respektive fyra hjul [21].

Tre omnihjul i cirkulärt mönster (se Figur 5.6) valdes på grund av den förenklade styrningen gentemot fyra hjuls bättre grepp (se Tabell 5.1). Konstruktionen testades även i en simulering för att se att funktionen uppfylls.

### 5.1.3 Drivande komponenter

Inför val av motorer som driver hjulen och skjutanordningen gjordes enklare beräkningar på vilket moment respektive varvtal som krävdes. Beräkningsfallen förenklades genom att försumma rullmotstånd och friktionskoefficienter samt att total överföring från motor till mark antogs. Detta gjordes eftersom endast en grov uppskattning behövdes som utgångspunkt. Motorer valdes sedan med högre moment och varvtal än vad som framkom från beräkningarna för att de även ska ta upp de förluster som uppstår på grund av bland

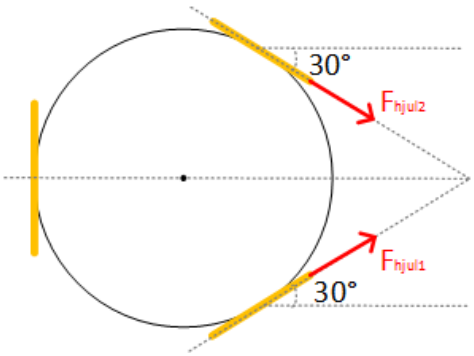


**Figur 5.6:** Konstruktion sett underifrån.

annat friktion.

### Stegmotorer

Motorerna som driver hjulen kräver ett visst moment för att klara av att accelerera från stillastående till topphastighet på given accelerationssträcka enligt önskemål (se Bilaga A) på prestanda. Med definitioner nedan beräknades önskad acceleration och moment med hjälp av formeler 5.1.6, 6.1.1 (Newtons andra lag) och 1.3.8 från Grahn & Janssons lärobok Statik och Dynamik [22].




**Figur 5.7:** Friläggning av krafterna från de drivande hjulen.

Definition av värden:
Vinkelskillnad mellan kraft och rakt framåt (se Figur 5.7): $\varphi = 30$ grader
Uppskattad massa på roboten: $m_{robot} = 2$ kilogram
Hjulens radie: $R_{hjul} = 0.025$ meter

Enligt kravspecifikation:
Minsta tillåtna topphastighet: $v_{topp} = 0.5\,\text{meter/sekund}$
Accelerationssträcka till topphastighet: $s_{acc} = 0.25\,\text{meter}$

Som drivande motorer valdes stegmotorer på grund av dess pålitlighet och jämna prestanda i förhållande till andra motorer. Dessa är viktiga faktorer för att minimera risken att de driver ojämnt och därmed att roboten förflyttar sig snett. Önskad acceleration, $a_{robot}$, beräknas utifrån formel 5.1.6 Grahn & Janssons [22].

$$\Rightarrow a_{robot} = \frac{v_{topp}^2}{2s_{acc}} = \frac{0.5^2}{2 \cdot 0.25} = 0.5\,\text{meter/sekund}^2 \tag{5.1}$$

Beräkning av moment:

Kraften, $F_{hjul}$, som krävs för att driva ett hjul beräknas med hjälp av Newtons andra lag.

$$2 \cdot F_{hjul} \cdot \cos(\varphi) = m_{robot} \cdot a_{robot} \tag{5.2}$$

Där vänsterledet är den totala kraften som riktas framåt. Kraften skapas med hjälp av två hjul (därav tvåan) som har en vinkel på 30 grader från körriktning (se Figur 5.7).

$$\Rightarrow F_{hjul} = \frac{m_{robot} \cdot a_{robot}}{2 \cdot \cos(\varphi)} = \frac{2 \cdot 0.5}{2 \cdot \cos(30)} \approx 0.577\,\text{Newton} \tag{5.3}$$

Momentet beräknas enligt formel 1.3.8 Grahn & Janssons [22].

$$M_{hjul} = F_{hjul} \cdot R_{hjul} = 0.577 \cdot 0.025 \approx 0.015\,\text{Newtonmeter} \tag{5.4}$$

Där $M_{hjul}$ är momentet som krävs på respektive hjul för att klara av att driva roboten enligt kraven.

Med beräkningar enligt ovan (se Ekvation (5.4)) fås de grundläggande kraven på motorn. En bipolär stegmotor, SY35ST36-1004A, valdes. För att styra motorerna behövs tillhörande drivkort och A4988 valdes då detta kort har funktionaliteten att påtvinga mindre stegvinkel genom olika konfigurationer av höga eller låga signaler som skickas till inportarna MS1, MS2 och MS3 (se Tabell 5.2).

| MS1 | MS2 | MS3 | Stegvinkel |
|---|---|---|---|
| Låg | Låg | Låg | Helsteg |
| Hög | Låg | Låg | Halvsteg |
| Låg | Hög | Låg | Fjärdedelssteg |
| Hög | Hög | Låg | Åttondelssteg |
| Hög | Hög | Hög | Sextondelssteg |

**Tabell 5.2:** Tabell över konfigurationerna av MS-portarna.

### 5.1.4 Servomotor

För att driva skjutanordningen så att bollen skjuts iväg med tillräckligt hög hastighet krävs ett visst varvtal, vilket beräknades och användes som referens vid val av motor. En positionsservo valdes då den styrs efter hur långt den vrids istället för hur många steg. Med detta är det lättare att reglera skjutanordningen för att ge så bra fart som möjligt utan att slå i andra komponenter på roboten. Sambandet mellan servomotorns varvtal och bollens hastighet är enligt Ekvation (5.5).

$$\underbrace{x\,[\mathrm{s}/60^\circ]}_{\text{motorns hastighet}} \Leftrightarrow \underbrace{\frac{60}{6x}\,[\mathrm{RPM}]}_{\text{motorns varvtal}} \Leftrightarrow \underbrace{\frac{2\pi h}{6x}\,[\mathrm{m/s}]}_{\text{bollens hastighet}} \tag{5.5}$$

Där $h$ är hävarmen, det vill säga avståndet mellan motorns axel och anslagspunkten på bollen.

Servomotorn GS-D9257 (för datablad se Bilaga C) med rotationshastighet $x = 0.07\,\text{sekunder}/60\,\text{grader}$ ansågs lämplig och valdes. Bollen skjuts iväg av servomotorn enligt Ekvation (5.5) med en hastighet:

$$0.07\,\text{sekunder}/60\,\text{grader} \Rightarrow 0.67\,\text{meter/sekund}$$

med $h$ som 45 millimeter, vid en spänning på sex volt.

### 5.1.5 Sensorer, fotoresistor och LED

Roboten använder sig av tre stycken avståndssensorer för att mäta avstånd till väggar och undvika krock. De två typerna av sensorer som undersöks är infraröda sensorer samt ultraljudssensorer. Infraröda sensorer är generellt billigare och verkar på kortare avstånd medan ultraljudssensorer verkar inom ett större spann och kan avgöra mer exakta avstånd. Som avståndsmätare valdes därför ultraljudssensorer, HC-SR04, på grund av deras mer exakta mätning.

En fotoresistor användes för att identifiera bollen. Dess funktion var bristfällig på grund av att ljuset föll in från flera håll vilket gjorde att den inte kunde identifiera om bollen befann sig framför eller inte. Problemet löstes genom att mörkläggande väggar runt inbuktningen monterades samt att en diod monterades på motsatt sida av inbuktningen för att fotoresistorn tydligt ska märka av bollens närvaro. Fotoresistorn av okänd modell erhölls från examinator. Diodkretsen som användes var Keyes KY-009 3-Colour LED.

### 5.1.6 Arduino

Kommunikationen mellan kamera, sensorer och motorer sker med hjälp av en Arduino. Arduino Mega ADK valdes som mikrokontroller på grund av dess antal digitala portar samt att den har ett USB-uttag för att koppla samman med en Android-enhet. Den är även kompatibel med Androids *Accessory Development Kit (ADK)* vilket är en referensimplementation på hur accessoarer, externa enheter som erbjuder funktionalitet, byggs

för Android [23]. Den har 54 digitala in- och utportar vilket är bland det mesta som kan fås med en Arduino.

Arduinon är komponenten som kräver högst spänning, rekommenderad spänning är enligt datablad [15] 7-12 volt och därför används en spänningskälla på cirka nio volt. För att se hur Arduinon är kopplad samman med resterande komponenter se Bilaga D.

## 5.2 Simulering av rörelse

För att få ökad förståelse om de köregenskaper som hjul placerade i ett cirkulärt mönster på roboten bidrar till utfördes med en konstruktion baserad på tre hjul. Simuleringens huvudfokus var att undersöka de kraftvektorer som motormomentet ger upphov till och hur roboten accelereras utav dem. Behandlingen utav detta gjordes i Matlab, en programvara som används i huvudsak för numeriska beräkningar. Simuleringen programmerades så att alla motorer kunde ge olika effekter vilket gjorde att fler manövrar, som att svänga, var möjliga.

### 5.2.1 Beräkningar i simuleringen

Simuleringen baserades på beräkningar inom mekaniken där formler är tagna från Grahn & Janssons [22]. Överföringen av momentet, $M_{motor}$, från motor till den accelererande kraften, $F_{hjul}$, mellan hjulet och underlaget beräknades med formel 1.2.4 enligt:

$$M_{motor} = F_{hjul} \cdot R_{hjul} \tag{5.6}$$

där $R_{hjul}$ är hjulets radie. Kraften $F_{hjul}$ ger upphov till robotens acceleration, $a_{robot}$, enligt Newtons andra lag, formel 6.1.1:

$$F_{hjul} = m_{robot} \cdot a_{robot} \tag{5.7}$$

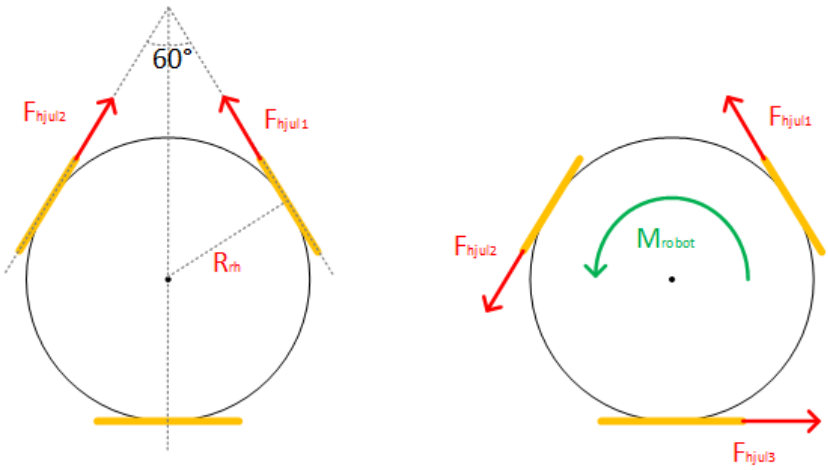
där $m_{robot}$ är robotens massa. Ovanstående beräkningar (se Ekvation (5.6)-(5.7)) ger inverkan av ett hjul på roboten. För att beräkna effekten av alla hjulen måste systemet som de utgör studeras. Roboten frilades för att finna de nödvändiga ekvationerna. I friläggningen togs hänsyn till två fall: acceleration framåt och rotation moturs. Detta är för att roboten ska kunna lokalisera bollen och färdas till den. Bild på friläggningen visas i Figur 5.8.

När ett hjul snurrar påverkas det utav en kraft som kallas rullmotstånd. Den definieras enligt Hibbler [24] som den kraften som agerar i motsatt riktning till färdriktningen och beror av robotens tyngd och underlagets förmåga att låta hjulen sjunka in i det. $C$ är den dimensionslösa rullmotståndskoefficienten och $N_{robot}$ är normalkraften som roboten påverkas av, vilket ger rullmotståndskraften $F_{rullm}$:

$$F_{rullm} = C \cdot N_{robot} \tag{5.8}$$

Robotens acceleration framåt ges av Ekvation (5.7), (5.8) och friläggningen (Figur 5.8):

$$(F_{hjul,1} + F_{hjul.2}) \cdot cos(30) - F_{rullm} = m_{robot} \cdot a_{robot} \tag{5.9}$$




**Figur 5.8:** Friläggning av robot. Till vänster accelererande krafter framåt, till höger roterande krafter.

Robotens moment, $M_{robot}$, kring dess mittpunkt ges av $R_{rh}$ som är avståndet mellan robotens mitt och hjulet, (5.6), (5.8) och friläggningen (Figur 5.8):

$$M_{robot} = (F_{hjul,1} + F_{hjul,2} + F_{hjul,3} - F_{rullm}) \cdot R_{rh} \tag{5.10}$$

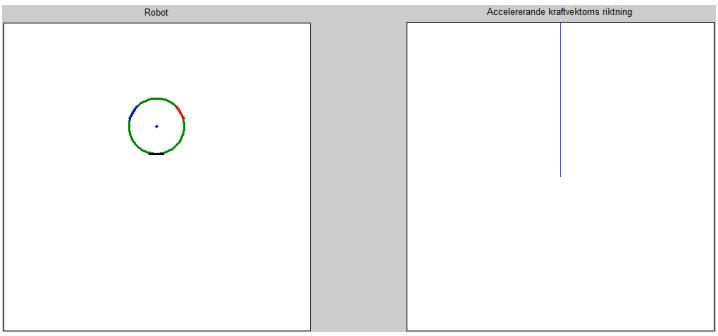
Överföringen från moment till en rotationsacceleration beräknas med formel 9.1.3 som beskriver rörelsemängsmomentet, $L$, med hjälp av yttröghetsmomentet, $I$, samt vinkelhastighet, $\dot{\omega}$:

$$L = \sum M = I \cdot \dot{\omega} \tag{5.11}$$

Ekvation (5.9) och (5.11) ger förändringar i hastighet och rotationshastighet. Matlab beräknar iterativt fram hastigheterna beroende på dessa förändringar som leder till förflyttning till ny position. Robotens rörelse ritas därefter upp i ett fönster i Matlab där simuleringen visualiseras (se Figur 5.9).

### 5.2.2 Begränsningar i simuleringen

Robotens simulerade rörelser begränsades till de två mest vanliga rörelsefallen: rotation och rörelse framåt (som även kan beskriva rörelse bakåt), med eventuell svängande rörelse. För att svänga under färd kan två metoder nyttjas: rotation av det bakre hjulet eller genom att ge olika effekt från motorn till de två accelererande hjulen. Rotation av det bakre hjulet är en enkel metod där högre rotationshastighet kan uppnås och detta utan att det påverkar hastigheten framåt markant. Om roboten skulle färdas i hög hastighet skulle denna metod kunna leda till att roboten välter om den försöker svänga för snabbt. Att svänga med hjälp av osymmetriskt moment från de två drivande motorerna ger motsatta för- och nackdelar: ökad stabilitet men nedsatt rörlighet. Den rörelse som begränsats är således färd diagonalt bakåt genom drivning av bakre hulet i samband med




**Figur 5.9:** Simulering i Matlab. Till vänster visas robotens position samt hur hjulen drivs genom färgkodning (blått för moturs, rött för medurs och svart för ingen drivning). Till visas höger accelerationskraftens riktning (ej storlek).

ett av de andra hjulen, med eventuell tillagd svängande rörelse. Rörelsen diagonalt bakåt är dock av samma typ som den framåt, så det som inte testas är att kunna kombinera dessa till en rörelse.

Inverkan av omnihjulens form på friktionen mellan hjul och underlag samt annan inverkan är inte känd varför beräkningar gjordes med ett ordinärt cirkulärt hjul som modell för omnihjulet.

### 5.2.3 Slutsatser från simulering

Simuleringen bekräftade att placeringen av hjulen i det cirkulära mönstret medför god kontroll över robotens rotation. Detta är önskvärt för att underlätta sökandet av föremål som exempelvis bollen. Uniform drivning av alla hjulen leder till en rotation utan förflyttning där rotationshastigheten bestäms av stegfrekvensen (hur ofta motoraxeln stegar framåt) i motorerna.

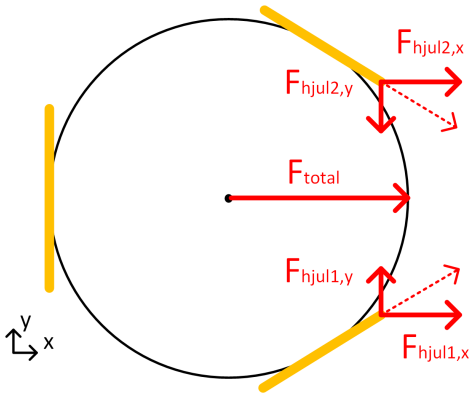
Simuleringen bekräftade även att en konstruktion med tre hjul uppfyller kraven. Med denna konstruktion åstadkommes förflyttning framåt genom att två hjul drivs med lika stort moment i motsatt riktning vilket resulterar att de accelererande krafternas sneda riktning tar ut varandra och resultatet är en acceleration riktad framåt (se Figur 5.10.

För rotation under färd fanns två metoder: drivning av bakre hjulet eller olika effekt från motorn till de två accelererande hjulen. Båda metoderna fungerade väl i simuleringen men drivning av bakre hjulet gav större accelererande kraftkomposant, varför den valdes.

Simuleringen visade inga nackdelar i konstruktionen varför den anses tillfredställa behoven: rörlighet och funktionalitet, samt tillåta fortsatt utveckling.

## 5.3 Tillverkning av robotskelett

Robotskelettet, det vill säga hela roboten förutom hjul och elektriska delar, tillverkades huvudsakligen i aluminium men fästelement i plast förekommer. Robotens två vånings-




**Figur 5.10:** Drivningen av roboten framåt, detta fallet i x-led. Krafterna i y-led från motorerna kancellerar varandra och roboten accelereras framåt av den summerade x-komposanten.

plan (aluminiumplattor) tillverkades med hjälp av vattenskärning medan majoriteten av de resterande aluminiumdelarna klipptes, sågades och bockades till önskad form. De fästelement som gjordes i plast var delarna mellan hjulen och dess motoraxlar. Komponenten skrevs ut i en 3D-skrivare och fick därför rätt form och dimensioner med endast lite arbete.

## 5.4 Programmering av applikation

Inför utvecklandet av Android-applikationen för objektigenkänning formulerades funktionskrav och lösningsmetoder studerades. De iterativa utvecklingsstegen var följande:

1. Fungerande kamera via applikation.
2. Kameravy konverterad från RGB- till NMI-systemet.
3. Färgfiltrering möjliggjord.
4. Växla mellan RGB- och NMI-vy i applikationen möjliggjort.
5. Möjliggjort funktion för att lokalt spara samt ladda färgkoder för filtrerad boll samt mål.
6. Grundläggande kommunikation med USB-enhet möjliggjord.
7. Protokoll (innehållande regler för hur kommunikationen mellan flera parter utförs) för USB-kommunikation med Arduino implementerat.

## 5.5 Programmering av robot

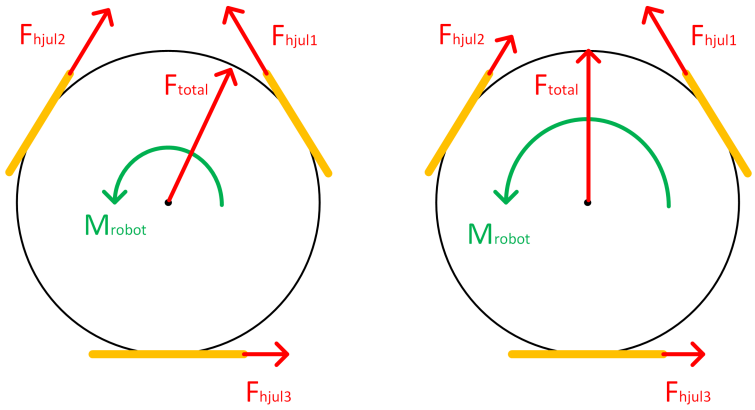
Mindre komponenter programmerades och testades enskilt under de iterativa utvecklingsstegen innan slutgiltig implementering. Processen var följande:

1. Fungerande servomotor.
2. Fungerande fotoresistor.
3. USB-kommunikation med Android-enhet upprättad.
4. Fungerande ultraljudssensor.
5. Protokoll för kommunikation med Android-enhet implementerat.
6. Fungerande stegmotorer.
7. Önskad rörelsefunktionalitet programmerad.
8. Finjustering av komponenter.

För att uppnå önskad rörelsefunktionalitet implementerades två rörelsemönster och två operationsförlopp enligt följande:

- Rörelsemönster
  - Köra fram/bak. Rakt eller svängande.
  - Rotera höger/vänster.
- Operationsförlopp
  - Hämta bollen.
  - Skjut bollen.

**Köra fram/bak:**
Det grundläggande rörelsemönstret för färd rakt fram implementeras genom att de främre hjulen driver med samma rotationshastighet. Den svängande rörelsen uppnås genom att det bakre hjulet agerar liknande ett roder och driver med en rotationshastighet fyra gånger lägre än de främre hjulen. Det hjul motsvarande vilken svängriktining som vill uppnås drivs med halva hastigheten (Se Algoritm 1) för att förhindra att den resulterande kraftkomposanten blir snett förskjuten och roboten åker snett (se Figur 5.11).




**Figur 5.11:** Rörelsemoment fram med svängande rörelse åt vänster. Vänstra figuren representerar scenariot där de främre hjulen roterar med samma rotationshastighet, kraftkomposanten är förskjuten åt höger. Högra figuren representerar scenariot där vänstra hjulet kör med halva rotationshastigheten, kraftkomposanten är förskjuten lite åt vänster.

Backfunktionaliteten är implementerad på samma vis som färd framåt med omvänd rotationsriktning på hjulen.

**moveForward(int hastighet, boolean Svänga, boolean Svänga vänster)**;
**if** *Svänga* **then**
  **if** *Svänga vänster* **then**
    Ställa in rotationsriktning på hjulen;
    Rotationshastighet för högra hjulet = hastighet;
    Rotationshastighet för vänstra hjulet = hastighet/2;
  **else** `// Svänga höger`
    Ställa in rotationsriktning på hjulen;
    Rotationshastighet för högra hjulet = hastighet/2;
    Rotationshastighet för vänstra hjulet = hastighet;
  **end**
  Rotationshastighet för bakre hjulet = hastighet/4;
  Stega alla tre motorerna;
**else** `// Köra rakt fram`
  Ställa in rotationsriktning på hjulen;
  Rotationshastighet för högra hjulet = hastighet;
  Rotationshastighet för vänstra hjulet = hastighet;
  Stega de främre motorerna;
**end**

**Algoritm 1:** Pseudokod för rörelsemönstret köra fram/bak med variation.

**Rotera höger/vänster:**
Rotationen implementerades genom att sätta samma rotationshastighet och rotationsriktning till alla hjulen. Rotationsriktningen förändras beroende på om önskad rotation är höger eller vänster.

**Hämta bollen:**
Implementeringen av bollhämtning upprättades genom att i en sekvens först åka rakt fram tills fotoresistorns avlästa resistans blir mindre än dess gränsvärde. Därefter skickas styrsignaler till servon att den ska greppa bollen. Om resistansen på fotoresistorn ej underskrider gränsvärdet efter två sekunder kommer roboten att backa en sekund. Fotoresistorns resistans läses av i en avbrottsrutin som körs tio gånger i sekunden (se Algoritm 2). En avbrottsrutin avbryter ordinarie exekveringsförlopp och initierar en ny sekvens av instruktioner. När denna nya sekvens är avslutad återupptas ordinarie exekveringsförlopp där den först avbröts.

**Hämta boll**;

boolean har_boll = false;
resistans = Läs av fotoresistorn;
**while** *Tid kört framåt < tidskonstant och har_boll == false* **do**
  Kör rakt fram;
  **if** *resistans < Gränsvärde av fotoresistorn* **then**
    har_boll = true;
  **end**
**end**

**if** *Har ej bollen* **then**
  Backa en sekund;
**else**
  Skicka låssignal till servo;
  Börja leta mål;
**end**

**Algoritm 2:** Pseudokod för operationsförlopp hämta boll.

**Skjuta bollen:**
Avgörandet när roboten ska skjuta bollen beräknas i applikationen. Arduinon skickar en signal till servomotorn att skjuta, samt skickar data till Android-enheten att söka efter boll.

## 5.6 Kommunikation

Kommunikationen mellan Android-enhet och Arduino implementerades med hjälp av ett protokoll som nyttjar operationskoder (hädanefter refererade till som OP-koder). De båda enheterna skickar och tar emot OP-koder i en kontinuerlig ström av datapaket bestående av två bytes där den första byten innehåller operationen som ska utföras

| OP-kod | Android | Arduino | Funktion |
|---|---|---|---|
| 0x01 | Mottager | Skickar | Sök efter boll. |
| 0x02 | Mottager | Skickar | Sök efter mål. |
| 0x03 | Skickar | Mottager | Objekt ej inom synfält. |
| 0x04 | Skickar | Mottager | Objekt centrerat inom synfältet. |
| 0x05 | Skickar | Mottager | Objekt direkt till vänster om centrum av synfältet. |
| 0x06 | Skickar | Mottager | Objekt direkt till höger om centrum av synfältet. |
| 0x07 | Skickar | Mottager | Objekt i vänstra utkanten av synfältet. |
| 0x08 | Skickar | Mottager | Objekt i högra utkanten av synfältet. |
| 0x09 | Skickar | Mottager | Hämta boll. |
| 0x0A | Skickar | Mottager | Skjut boll. |
| 0x0B | Skickar | Mottager | Backa. |

**Tabell 5.3:** Tabell över OP-koder samt deras funktioner.

| Hastighetskod | Funktion |
|---|---|
| 0x01 | Helsteg. |
| 0x02 | Halvsteg. |
| 0x04 | Fjärdedelssteg. |
| 0x08 | Åttondelssteg. |
| 0x16 | Sextondelssteg. |

**Tabell 5.4:** Tabell över hastighetskoder samt deras funktion. Samtliga koder skickas från Android-enhet och mottages av Arduino.

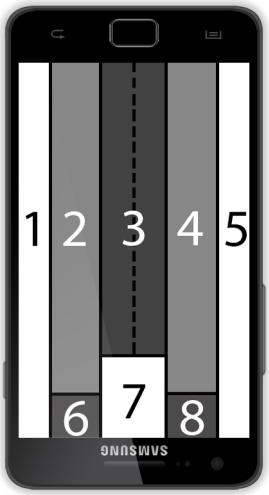
av roboten (se Tabell 5.3) och den andra byten innehåller hastigheten på de drivande motorerna (se Tabell 5.4).

Implementeringen av kommunikationen upprättades via en USB-anslutning enligt Böhmer [25]. Android-enhetens uppgift är att kontinuerligt förse Arduinon med det filtrerade objektets position medan Arduinons uppgift är att ge information till applikationen om vilket objekt som ska lokaliseras samt exekvera de mottagna OP-koderna.

### 5.6.1 Arduino till applikation

Arduinons första uppgift är att kommunicera med applikationen om vilket objekt som ska lokaliseras. Detta bestäms beroende på vilket utslag fotoresistorn ger.

- Hög resistans innebär att roboten ej har bollen och därmed ska boll sökas. Här skickas OP-koden 0x00 till applikationen.




**Figur 5.12:** Fördelning av segmenten. Streckad linje representerar referenslinjen. Telefonen är upp och ner på grund av att detta är den position den har på roboten.

- Låg resistans innebär att roboten har bollen och därmed ska mål sökas. Här skickas OP-koden 0x01 till applikationen.

Vad som innebär låg eller hög resistans bestäms av ett gränsvärde som kalibreras vid första starten av roboten då den roterar ett varv och avläser omgivningens ljusstyrka.

Arduinons andra uppgift är att exekvera de OP-koder som skickas från telefonen. Detta upprättades genom användandet av de rörelsemönster och operationsförlopp som definierades enligt ovan (se 5.5 Programmering av robot).

### 5.6.2 Applikation till Arduino

Kommunikationen mellan applikation och Arduino upprättades genom att kameravyn delades in i åtta olika segment samt en referenslinje centrerad på skärmen som används för att avgöra om roboten är riktad mot målet (se Figur 5.12). När det filtrerade objektets mittpunkt befinner sig inom ett visst segment skickas motsvarande OP-kod samt vilken hastighet roboten ska köra i via USB-anslutningen till Arduinon.

Figur 5.12 samt nedanstående punktlista förklarar vilka OP-koder samt hastighetskoder som skickas inom vilka segment.

- Segment 1:

  Objekt i vänstra utkanten av synfältet. Roboten ska köra fram med svängande rörelse vänster. OP-kod 0x07 och hastighetskod 0x02 halvsteg skickas oberoende av objektet

- Segment 2:

Objekt direkt till vänster om centrum av synfältet. Roboten ska köra fram med svängande rörelse vänster. OP-kod 0x05 och hastighetskod 0x04 fjärdedelssteg skickas oberoend av objekt.

- Segment 3:

  Objekt centrerat inom synfältet. Roboten ska köra rakt fram. OP-kod 0x04 och hastighetskod 0x02 halvsteg skickas oberoende av objekt.

- Segment 4: Objekt direkt till höger om centrum av synfältet. Roboten ska köra fram med svängande rörelse höger. OP-kod 0x06 och hastighetskod 0x04 fjärdedelssteg skickas oberoende av objekt.

- Segment 5: Objekt i högra utkanten av synfältet. Roboten ska köra fram med svängande rörelse höger. OP-kod 0x08 och hastighetskod 0x02 halvsteg skickas oberoende av objektet

- Segment 6 och 8: Bollen är snett framför. Roboten ska backa en sekund. OP-kod 0x0B och hastighetskod 0x04 fjärdedelssteg skickas oberoende av objekt.

- Segment 7: Bollen är nära framför roboten. Roboten ska köra fram och låsa skjutanordningen. OP-kod 0x09 och hastighetskod 0x04 fjärdedelssteg enbart när bollen är det sökta objektet.

- Referenslinje:

  När någon del av denna linje befinner sig inom konturen av det filtrerade målet samt att arean på konturen uppnår en viss storlek skickas OP-koden 0x0A för att skjuta bollen.

- Objektet ej är inom synfältet (Mittpunkten är ej i något segment): Objekt ej i synfältet. Roboten ska rotera och söka efter objektet. OP-kod 0x03 och hastighetskod 0x02 halvsteg skickas oberoende av objekt.

Vid tillfällen då flera objekt med liknande nyans befinner sig inom kameravyn kommer endast det objekt med störst area att filtreras för minskat brus vid nyansskillnader. Detta uppnås genom att vid filtreringen spara konturerna för alla objekt med samma filtreringsfärg i en lista. Denna listan itereras igenom för att urskilja vilken kontur som har störts area (se Algoritm 3).

**Iterering av största area**;

```
Kontur contour ;                          // Den kontur som visas på skärmen
for alla konturer i listan do
    if contour ej initierad then
        contour = nästa kontur;
    end
    if Area av contour < Area av nästa kontur then
        contour = nästa kontur;
    end
end
```

**Algoritm 3:** Pseudokod över itereringen av största kontur

## 5.7 Sammankoppling

Implementeringen av samverkan mellan applikationen och Arduinon upprättades genom att sammankoppla respektive OP-kod skickad från applikationen med ett rörelsemönster eller operationsförlopp (se Algoritm 4).

```
Sammankopplingen;
//Variabeln "senaste_sida" representerar den sida där bollen senast befann sig på;
switch OP-koden do
    case 0x03                                   // Objektet ej inom synfältet
        if senaste_sida är vänster then
            Rotera vänster;
        else
            Rotera höger;
        end
    end
    case 0x04                              // Objekt centrerat inom synfältet
        Kör rakt fram;
    end
    case 0x05   // Objekt direkt till vänster om centrum av synfältet.
        Kör fram med svängande rörelse åt vänster;
    end
    case 0x06     // Objekt direkt till höger om centrum av synfältet.
        Kör fram med svängande rörelse åt höger;
    end
    case 0x07                 // Objekt i vänstra utkanten av synfältet.
        Rotera vänster;
        senaste_sida = vänster;
    end
    case 0x08                   // Objekt i högra utkanten av synfältet.
        Rotera höger;
        senaste_sida = höger;
    end
    case 0x09                                                  // Hämta boll
        Operationsförlopp hämta boll;
    end
    case 0x0A                                                  // Skjut boll
        Operationsförlopp skjut boll;
    end
    case 0x0B                                                       // Backa
        Backa en sekund;
    end
endsw
```

**Algoritm 4:** Sammankoppling av OP-kod till respektive rörelsemoment/operationsförlopp

# 6

# RESULTAT

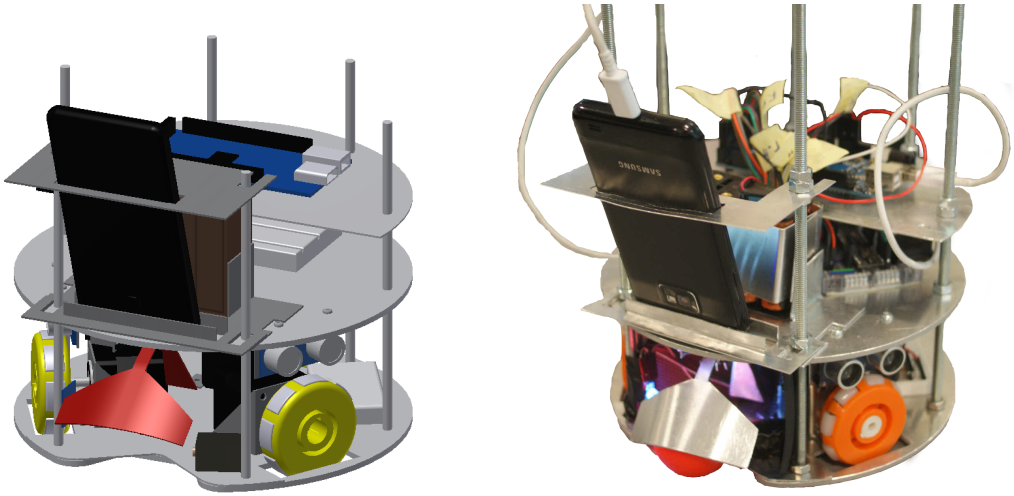
Kapitlet behandlar resultatet av robotens design och applikationens samt robotens funktion. Därefter redovisas verifiering av kraven samt projektets budget.

## 6.1 Robotens design

Skelettet till roboten är uppbyggt i två våningar med metallplattor av cirkulär form. Dessa hålls samman med sex stycken metallstänger fördelade längs plattornas kanter (se Figur 6.1a-6.1b). De tre omnihjulen styrs av motorer fastmonterade på bottenplattans ovansida (se Figur 6.2). Resultat från rörelsesimuleringen (se Avsnitt 5.2.3 Slutsatser från simulering) visar att denna konstruktion uppfyllde kraven på rörlighet.

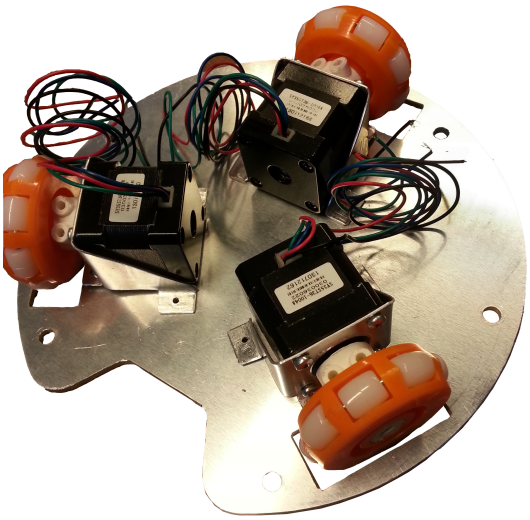
När roboten detekterar bollen och ”fångar in” den i inbuktningen observeras detta av en fotoresistor monterad på en av inbuktningens sidor (se detalj ett i Figur 6.3). För att fotoresistorns funktion ska vara pålitlig är en diod (se detalj två i Figur 6.3) monterad mitt emot för att ljuset den registrerar ska ha markant skillnad när bollen är i inbuktningen respektive inte är det. ”Skeden” (se detalj tre i Figur 6.3) styrs av servomotorn och låser bollen. På servon är även ”paddeln” som skjuter iväg bollen monterad (med cirka 90 graders vinkel mellan den och skeden).

En applikation för Android-enheten skapades för att hantera objektidentifiering med hjälp av OpenCV. Android-enheten monteras framtill på roboten för att enkelt kunna

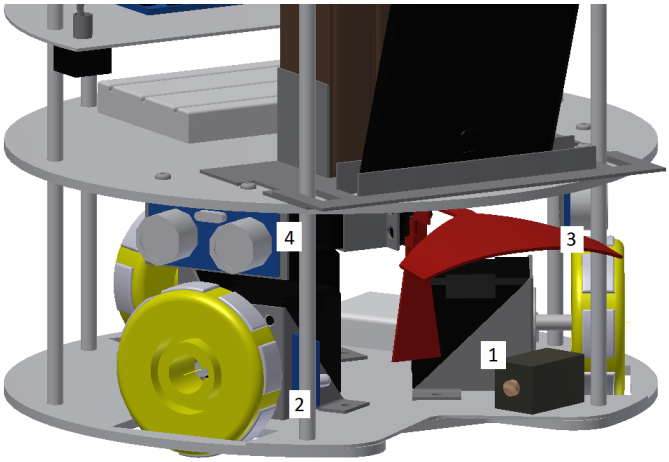


**(a)** CAD-modell på designen. **(b)** Bild på färdig robot.

**Figur 6.1:** Robotens utveckling under projektet.



**Figur 6.2:** Hjul och stegmotorer monterade på bottenplattan

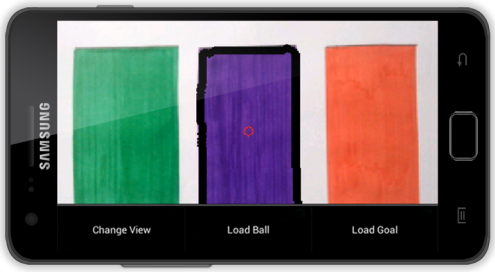



**Figur 6.3:** Nedre delen av konstruktionen. 1: Fotoresistor (svart-beige). 2: Diodkrets (blå). 3: Sked och paddel (röd). 4: Ultraljudssensor (blå-grå), för avståndsbedömning.

lokalisera bollen. Runtom roboten sitter tre stycken sensorer (detalj fyra i Figur 6.3) som ska mäta avstånd och se till att roboten inte kolliderar med något. Deras funktion har ej programmerats och implementerats ännu. På robotens översta plan finns en Arduino som står för kommunikationen mellan Android-enheten och resterande komponenter.

### 6.1.1 Designproblem

Hjulen monterades på motoraxlarna med fästelement däremellan. Dessa fästelement svällde efter tillverkning i 3D-skrivaren och var därmed tvungna att filas till för att




**Figur 6.4:** Huvudaktiviteten med meny där lila färg blivit filtrerad. Rutorna är från vänster till höger: grön, lila, orange.

passa. Detta medförde ojämnheter som ledde till att roboten fick sneda hjul. På grund av detta skapas en ojämn kraftöverföring från hjul till underlag.

Robotens motorer är ej kraftiga nog att accelerera den i vissa fall. Detta beror på två saker: robotens vikt samt omnihjulens utseende. Roboten väger cirka 2.2 kilogram vilket har gjort att den, beroende på underlag, får svårt att accelerera från stillastående på grund av för hög friktion. Omnihjulen som investerats i har för stora småhjul längs omkretsen samt att dessa även sitter för glest mellan varandra. Detta medför att hjulen kan balansera i två olika höjdlägen; läge ett då hjulet står rakt på ett vitt mindre hjul och läge två då det står mellan två vita småhjul. Följden av detta är att när roboten tar sig framåt märks denna höjdskillnad och roboten studsar fram. Om det bakre hjulet står i läge två vid framåtkörning, det vill säga när bakre hjulet släpas med, så blir friktionen för hög för att de mindre hjulen ska sättas i rullning och roboten får även i detta fall svårt att accelerera.

## 6.2 Design och funktion av applikationen

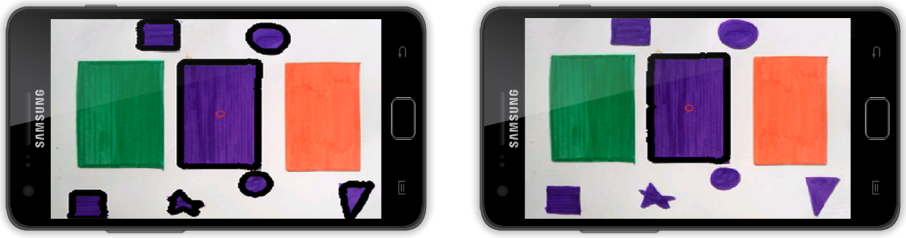
Applikationen består av två aktiviteter: huvudaktiviteten samt filtreringsaktiviteten. Nedan beskrivs grafiken samt den underliggande logiken för dessa aktiviteter.

### 6.2.1 Huvudaktivitet

Huvudaktiviteten består av en kameravy som använder sig av RGB-modellen för att ge en tydlig bild av det identifierade objektet. Konturen av objektet färgas svart för att ge en visuell indikation med hjälp av den binära filtreringen som skapas i filtreringsaktiviteten (se Figur 6.4). Röd cirkel i bilden representerar konturens mittpunkt. Vid förekomst av flera objekt av samma färg på skärmen identifieras enbart den kontur med störst area (se Figur 6.5a samt 6.5b).

Menyn för den huvudaktiviteten består av tre valmöjligheter:

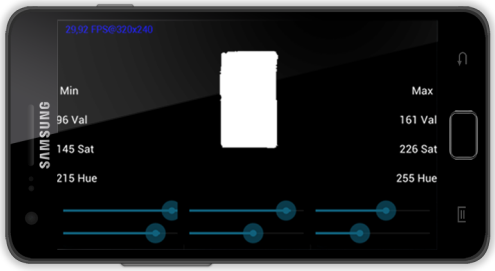
1. Change View



**(a)** Filtrering av lila färg där alla konturer visas på skärmen

**(b)** Filtrering av lila färg där enbart största konturen visas på skärmen

**Figur 6.5:** Visar skillnaden när applikationen filtrerar bort de mindre figurerna.




**Figur 6.6:** Filtreringsaktivitet där lila färg blivit filtrerad. Kameravyn visar en binär representation av bilden.

- Detta menyval sparar undan huvudaktivitetens tillstånd innan den stoppas. Därefter startas filtreringsaktiviteten.

2. Load Ball
   - Detta menyval laddar information från Android-enhetens minne med data angående bollens specifika filtreringsvärde.
3. Load Goal
   - Detta menyval laddar information från Android-enhetens minne med data angående målets specifika filtreringsvärde.

### 6.2.2 Filtreringsaktivitet

Filtreringsaktiviteten hanterar inställningarna för vilka färger som ska filtreras. Den består av en kameravy samt skjutreglage med tillhörande informationsrutor (se Figur 6.6).

Skjutreglagens funktion är att ställa in de max- respektive mingränser för nyans, mättnad och intensitet som används vid filtreringen av datan. De nedersta tre ställer in mingränser medan de övre tre ställer in maxgränserna. Deras värden skrivs dessutom till respektive informationsruta. Kameravyn fyller två funktioner:


**Figur 6.7:** Filtreringsaktiviteten där lila färg blivit filtrerad med meny och är redo att sparas.

1. Bearbetning av data (den bild som fångas upp av kameran i RGB-format):
    - Konverterar datan från RGB-modellen till NMI-modellen och därefter till binär form beroende på de konfigurerade gränsvärdena av skjutreglagen (se Figur 6.6).
2. Visuell representation av det filtrerade objektet:
    - Den funktion som används när en ny färg ska filtreras. Kameravyn visar den binära bilden som vid förändring av skjutreglagens värden ger en tydlig visualisering av vilken färg som filtrerats (se Figur 6.6).

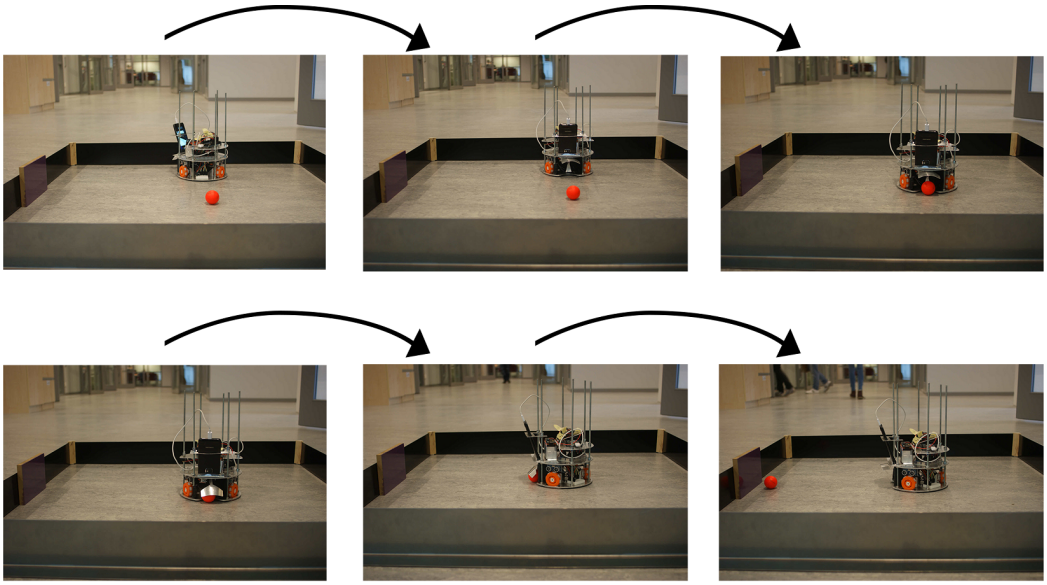
Menyn för filtreringsaktiviteten består av två valmöjligheter (se Figur 6.7):

1. Save Ball
    - Detta menyval tar de nuvarande filtreringsgränserna och sparar dem i Android-enhetens minne för bollen.
2. Save Goal
    - Detta menyval tar de nuvarande filtreringsgränserna och sparar dem i Android-enhetens minne för målet.

Dessa menyvalen kombinerat med de från huvudaktiviteten tillför möjligheten att spara och ladda information mellan sessionerna. Applikationen kan avslutas och därefter fortsättas vid senare tillfälle utan att behöva konfigurera om färgfiltreringen.

## 6.3 Robotens funktion

Vid start roterar roboten till dess att bollen identifieras inom kamerans synfält varpå rörelse mot den inleds (se Figur 6.8). Är bollen ej centrerad när den försvinner från kameravyns underkant backar roboten för att tillåta en mer precis uppfångst. Vid centrerad boll kör roboten fram tills dess att fotoresistorn registrerar en förändring i ljusintensitet varpå skjutanordningen fälls ned. Registreras ingen förändring hos fotoresistorn på två



**Figur 6.8:** Robotens funktion vid körning. Övre raden visar sökande och uppfångade av boll. Andra raden visar sökande efter, samt skott i mål.

sekunder har bollen missats och en backmanöver inleds för att återfå kontroll över dess position.

Med bollen greppad roterar roboten till dess att kameran identifierar målet varpå rörelse mot det inleds. Roboten rör sig framåt till dess att mittlinjen på kameravyn faller inom målarean samt att arean är stor nog för att möjliggöra ett skott i mål varpå bollen skjuts. Efter skott börjar processen på nytt med lokalisering av boll. För blockschema över hur signaler skickas se Bilaga B.

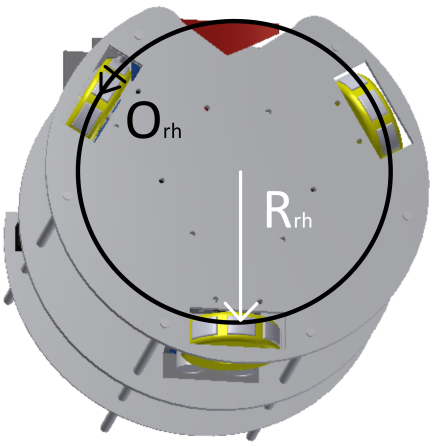
## 6.4 Uppfyllnad av krav

En testkörning på maxfart längs en rak linje på en meter användes för att verifiera både kravet på robotens förmåga att köra rakt fram samt önskemålet om robotens hastighet. Slutfört test visade att avvikelsen mellan robotens start- och slutpunkt var mindre än en grad vilket är väl inom godkända gränser. Körtiden mättes till 6.5 sekunder vilket ger hastigheten:

$$v = \frac{s}{t} = \frac{1}{6.5} \approx 0.154 \text{ meter/sekund}$$

Roboten lyckades således inte nå upp till den önskade hastigheten på 0.5 meter/sekund.

Kravet på rotationsvinkel testades genom att rotera roboten 360 grader. Hjulen färdas under denna sträcka $O_{rh} = 2 \cdot R_{rh} \cdot \pi = 2 \cdot 0.08165 \cdot \pi$ meter (se Figur 6.9 för definition av beteckningar). De valda motorerna kräver 200 steg ($steg_{360}$) för en full rotation av hjulen vars radie, $R_{hjul} = 0.0255$ meter, som då rullar sträckan $O_{hjul} = 2 \cdot R_{hjul} \cdot \pi = 2 \cdot 0.0255 \cdot \pi$




**Figur 6.9:** Måtten för den omkrets ($O_{rh}$) hjulen färdas då roboten roterar 360° samt avståndet ($R_{rh} = 0.08165$ meter) från robotens center till hjulets mitt.

meter. Antal steg som motorerna tar för att rotera roboten ett fullt varv ges därför av ekvationen:

$$\Rightarrow n = \frac{O_{rh}}{O_{hjul}} \cdot steg_{360} \approx \frac{0.163 \cdot \pi}{0.051 \cdot \pi} \cdot 200 \approx 639 \text{ steg}$$

Då roboten roterar med halvstegshastighet krävs det att motorerna tar emot 2 stegkommando för att ta ett steg. Detta gör att $2 \cdot n = 1278$ steg skickas för att få ett varvs rotation på roboten. Vid praktiska tester med 1278 steg roterade roboten 360 grader med avvikelse på plus/minus tio grader under tio försök vilket uppfyller både önskemålet samt kravet.

Önskemålet att bollen ska kunna skjutas rakt fram med en avvikelse på maximalt tio grader per meter uppfylls. Tester visade att den maximalt avvek fem grader. Dock skjuts bollen ej med någon hög hastighet vilket medför att ett lutande golv skulle påverka avvikelsen ganska drastiskt.

Resterande krav verifierades och uppfylldes i samband med att roboten utförde sin förväntade uppgift att lokalisera och fånga boll samt skjuta den i mål.

## 6.5 Budget

Nedan visas kostnader för inköpta komponenter (se Tabell 6.1). Projektets avgränsningar var maximalt 5000 kronor. Konstruktionsmaterial var exkluderat denna summa, därav avsaknaden av material som till exempel Aluminium och plast i tabellen nedan.

Den beräknade kostnaden för komponenterna var 2825 kronor vilket stämmer väl med den slutgiltiga kostnaden på cirka 3100 kronor. Anledningen till att den faktiska

| **Produkt** | **Antal** [st] | **Kostnad** [Total kostnad i kr] |
|---|---|---|
| Stegmotor | 3 | 535,03 |
| Drivkort | 5 | 527,45 |
| Positionsservo | 1 | 196,55 |
| Ultraljudssensor | 4 | 199,6 |
| Arduino Mega ADK | 1 | 578,61 |
| Fotoresistor | 2 | 0 |
| Diodkrets | 1 | 49,9 |
| Omnihjul | 3 | 159,65 |
| Batterier | 2 (Pack) | 258 |
| Kopplingskomponenter | 5 (Pack) | 264,5 |
| Kopplingsplatta | 2 | 102,87 |
| Batterihållare | 2 | 99,7 |
| Golfbollar | 1 (Pack) | 127 |
| **Totalt** | | **3098,86** |

**Tabell 6.1:** Budgetkalkyl

kostnaden blev högre än den beräknade beror på två saker: förändringar i inköpslistan samt att några komponenter var defekta. Förändringarna i inköpslistan grundade sig på att projektgruppen fick nya kunskaper längre in i projektet och därmed insåg att vissa antaganden som gjordes i den inledande fasen inte var optimala. Därmed skedde en del förändringar: vissa planerade komponenter behövdes inte längre medan det var andra komponenter som inte var med i planeringen som behövdes införskaffas.

# 7

# DISKUSSION

Roboten uppfyller alla krav och har de önskade funktionerna. Den kan hitta, hantera och skjuta bollen i mål som önskat, även om den inte alltid gör detta på snabbast möjliga sätt. Således finns en del problem och ett antal områden som skulle kunna förbättras.

## 7.1 Robotens design

Robotens design i sin helhet, exempelvis form och storlek, skapades utefter vilka komponenter som valdes och hur de på enklaste sätt kunde monteras. De funktioner eller komponenter som skapade de största problemen är följande: infångning av boll, robotens rörelse, motorerna samt robotens vikt. Nedan diskuteras samt ges förbättringsförslag på dessa.

### 7.1.1 Infångning av boll

Initiellt skulle bollen identifieras med hjälp av en ultraljudssensor av samma modell som avståndssensorerna baktill och på sidorna (se Avsnitt 5.1.5 Sensorer, fotoresistor och LED). Denna skulle placeras framtill och därmed med enkelhet kunna identifiera när bollen var fångad. Den var dock för stor för att monteras där på grund av övriga komponenter som redan använde all ledig yta. Därmed valdes att använda en fotoresistor eftersom den var mycket mindre till storleken.

Omgivningens ljus påverkar filtreringen av boll samt mål. På grund av detta är det viktigt att filtreringen av färgerna kontrolleras när roboten byter miljö. Detta beror på att kameran i Android-enheten uppfattar färgen annorlunda om det är ljusare respektive mörkare i omgivningen. Detta har hittills inte uppfattats som något problem under användandet av roboten, då ljuset oftast är konstant, men skulle kunna påverka resultatet. Ett sätt för att undkomma detta vore att alltid köra roboten på samma ställe alternativt göra egen anordning med lampor eller liknande för att minska fluktuation i ljusstyrka.

### 7.1.2 Rörelse

Robotens rörelse påverkades av hur hjulen var monterade. Fästelementet mellan hjulet och motoraxeln tillverkades i plast med hjälp av en 3D-skrivare, vilket i slutändan resulterade i att hjulen blev skeva. Detta påverkade robotens rörelse och kunde ha hindrats genom att tillverka fästelementen på annat vis, till exempel genom att tillverka dem i trä eller metall.

Simuleringen av robotens rörelse visar att de nuvarande motorerna ska vara tillräckligt kraftfulla, dock tog simuleringen inte hänsyn till vilka extra krafter som krävs

(framförallt vid stillastående) på grund av omnihjulens utformande enligt ovan, vilket leder till att de motorer vi har nu inte är tillräckligt kraftfulla. I simuleringen används även en rullmotståndskoefficient som antas vara 0.35, vilket är högt i jämförelse med Gillespies exempel på olika rullmotstånd [26], där exempelvis 0.3 är koefficienten för bildäck på sand. Det använda värdet stämmer inte men då den är högre än i verkligheten så påverkas inte resultatet negativt på grund av det.

### 7.1.3 Motorer

Stegmotorerna är jämna och fyller i det avseendet sin funktion. Däremot uppfylls ej önskemålet på topphastighet. För att kunna nå önskvärd hastighet krävs starkare motorer. Med detta skulle det bli lättare för roboten att hinna ifatt bollen om den hamnar i rullning och det skulle leda till ett mer intensivt fotbollsspelande. Genom att använda en positionsservo som kan komma upp i högre hastighet än den vi har nu skulle ett intensivare spelande uppnås, detta på grund av att bollen skjuts iväg hårdare. Även servomotorn uppfyller sina huvudsakliga funktioner och kan dessutom lätthanterligt ställas in efter vart den ska stanna eller byta riktning.

### 7.1.4 Vikt

Redan tidigt i projektet upptäcktes att roboten var väldigt känslig för vikt samt hur vikten var fördelad över roboten. Den drev lätt ojämnt så fort tyngdpunkten var förflyttad från mitten, vilket ledde till att roboten rörde sig snett då den skulle åka rakt fram. Det gjorde att det blev viktigt att montera fast komponenterna jämt för att undvika problemet. Robotens totala vikt har stor betydelse då det påverkar friktionen mot underlaget och därmed hur pass snabbt roboten kan köra. Med de nuvarande omnihjulen är roboten på gränsen till för tung, den kör bra på vissa underlag och är känslig så fort underlaget har något högre friktionskoefficient. Anledningen till dess tyngd är främst de olika våningarna (bottenplatta och mittplatta) som är tillverkade i onödigt tjockt material (4 millimeter). Anledningen till tjockeleken på dessa plattor är för att säkerställa prototypens hållbarhet även om roboten skulle råka ut för en krock eller liknande.

## 7.2 Konstruktion mjukvara

Programmeringen av Arduino var enkel då exempelkod fanns att tillgå. Problemet var att eftersom Arduinon skickar och tar emot signaler till och från alla hårdvarukomponenter på roboten betydde en försenad leverans eller icke funktionabel komponent även en försening i programmeringen, detta var främst märkbart under projektets gång då ett drivkort till stegmotorerna var defekt och ett nytt behövde beställas.

Tillgängliga komponenter testades enskilt, oftast med inte mer än ett par rader kod. Exempelkod för alla möjliga sorters komponenter fanns i flera fall tillgängligt att hämta vid beställning av komponenten (exempelvis RGB-dioden som används i projektet).

# 8

# SLUTSATS

Projektets mål var att skapa en robot som kan spela fotboll, vilket uppnåddes. Roboten klarar av att utföra operationslistan:

1. Finna bollen.
2. Färdas till och ta kontroll över bollen.
3. Finna målet.
4. Få bollen in i målet.

Den gör detta genom objektidentifieringsmjukvaran som programmerats i Android-telefonen placerad på roboten. Mjukvaran är baserad på mjukvarubiblioteket OpenCV och använder Android-enhetens kamera med vilken den urskiljer boll och mål från omgivningen genom deras specifika färg. Android-enheten skickar information om var objektet befinner sig i förhållande till roboten till en Arduino, en mikroprocessor, som styr robotens motorer och övriga komponenter.

Innan utvecklingen av roboten påbörjades formulerades krav och önskemål för robotens egenskaper. Alla kraven uppfylldes, däribland de mycket viktiga att roboten skulle kunna färdas rakt och rotera till given vinkel, vilket gav roboten behövd prestanda. Även majoriteten av önskemålen uppfylldes, men inte de gällande önskad acceleration och hastighet. Motorerna som beräknades vara kraftiga nog levererar tillräckligt med kraft för att roboten ska uppnå målet, men för önskemålen skulle starkare motorer samt eventuellt bättre hjul behövas.

Robotens konstruktion består av komponenter som alla är kommersiellt tillgängliga eller tillverkade från aluminiumplåt, förutom plastkomponenterna som sitter mellan motor och hjul. De är tillverkade i en 3D-skrivare och således mindre tillgängliga för allmänheten. Det anses ändå att roboten är konstruerad med kommersiellt existerande delar då plastkomponenterna kan ersättas med metall- eller trädetaljer.

Projektet avslutas med uppnådda mål och förbättringsmöjligheter, samt en stor möjlighet att vidareutveckla roboten och utöka till fler robotar.

# 9

# VIDAREUTVECKLING

Resultatet blev väldigt lyckat men det finns fortfarande mycket utrymme för utveckling. Nedan belyses några av de viktigaste punkterna.

## 9.1 Utöka robotens synfält

Robotens synfält begränsas utav Android-enhetens kameras synfält, så om det skulle utökas ger det roboten mer och bättre data att ta beslut utifrån. Förbättring utav kamerans synfält kan fås genom två metoder:

- Sätta en vidvinkellins på kameran.
- Göra Android-enhetens position på roboten förändringsbar. Om enheten ges förmågan att flyttas upp och ner samt att rotera skulle den kunna lokalisera föremål utan att hela roboten behöver rotera.

Ytterligare förbättringar skulle kunna fås genom exempelvis: fler kameror på roboten eller att föremål som ska identifieras skickar ut en IR-signal som roboten kan detektera. IR-signal referar till en infraröd signal som kan detekteras av IR-mottagare som kan monteras på roboten.

## 9.2 Förbättra robotens bollhanteringsförmåga

Robotens enkla skjutanordning valdes för att minska arbetsbördan i dess utveckling. I RoboSot:s regelbok får enbart 30 procent utav bollen döljas från motståndarna när en robot har kontroll över den, något som den utvecklade roboten inte tar hänsyn till. Arbete skulle således kunna utföras för att göra roboten mer enligt FIRA:s regler.

Även robotens skjutanordning skulle kunna förbättras så att bollen skjuts rakare och snabbare. Utrymmet mellan robotens två främre motorer skulle kunna användas för att placera en anordning baserad på lufttryck eller fjäderkraft som kan ge större kraftöverföring till bollen, vilket skulle resultera i att bollen färdas snabbare och rakare genom att vara mindre känslig för ojämnheter i underlaget.

## 9.3 Flera robotar

En fotbollsmatch kräver två lag som spelar mot varandra, där vardera lag kan bestå av en eller flera robotar. För att kunna göra detta krävs att fler robotar byggs och programmeras. En match mellan två robotar kan utföras enbart genom att en till robot konstrueras då båda robotarna kan använda samma boll och mål.

En annan utvecklingsmöjlighet är att fokusera på samarbete istället för konkurrensen. Två eller fler robotar skulle kunna programmeras att samarbeta genom att passa och genomföra strategier av både offensiv och defensiv karaktär. Det svåra i denna utveckling blir kommunikationen. Om en större mängd robotar planeras att användas kan kraftigare processorer behövas för att bearbeta den ökade informationsmängden. Detta skulle kunna göras genom att:

- Använda en ytterligare Arduino. Denna kan stå för att hantera en del av informationsmängden, exempelvis kommunikationen.
- Byta ut Arduinon mot ett kraftigare mikrokontrollerkort än Arduinon. Detta skulle exempelvis kunna vara en Raspberry Pi [27].
- Överföra en del av informationsbehandlingen till Android-enheten. Detta skulle kunna medföra en minskad uppdateringshastighet utav informationen som fås från enhetens kamera då processorkraft tas från den.
- Effektivisera kod. På Arduinon och Android-enheten finns kod som beskriver vad roboten ska göra. Kan denna kod effektiviseras skulle det kanske räcka.

# SPECIELLA TACK

UNDER PROJEKTET har gruppen fått stor hjälp från några personer som ett extra tack sträcks ut till.

**Stefan Bergquist**
Projektets handledare som bistått med expertis och konstruktiv kritik på såväl robotens utveckling som rapporten. Extra stort tack för den tid du lagt på att läsa genom utkast på rapporter, du har varit en stor hjälp.

**Jonas Fredriksson**
Projektets examinator som bistått med hjälp när Stefan varit otillgänglig, samt givit konstruktiv kritik vid behov och på delrapporter.

**Göran Stigler, Jan Bragee och Reine Nohlborg**
Personalen i Prototyplabbet för all hjälp och handledning under konstruerandet och byggandet av robotskelettet.

**Erik Karlsson**
Mer känd som ”LED-mannen”, geniet som kläckte förslaget att tillföra en LED-lampa i inbuktningen för att förbättra fotoresistorns funktion.

# KÄLLFÖRTECKNING


[1] Federation of International Robot-soccer Association, (2014) FIRA, http://www.fira.net/main (23-02-2014).

[2] J. H. Kim, D. J. Stoiner, S. Choi, N. S. Kuppuswamy, (2006) FIRA RoboSot Competition, http://www.fira.net/contents/data/RoboSot.pdf (30-01-2014).

[3] EuroSon99, (2010) Robot Football Final - Portugal VS Holland, [YouTube] http://www.youtube.com/watch?v=QSuje3iy2pA (23-01-2014).

[4] Chris I-B, (2013) FIRA 2013 Penalty Kick, [YouTube] http://www.youtube.com/watch?v=_keH_SMHiH0 (23-01-2014).

[5] Robotclub Malaysia, (2013) HongQin - Robot for FIRA Cup 2013 (HD), [YouTube] http://www.youtube.com/watch?v=5mMD-35Hdx4 (23-01-2014).

[6] V. Vikas, (2013) OpenCV Android Robot with Arduino Mega ADK, [YouTube] https://www.youtube.com/watch?v=HHpo-wXAqlo (23-01-2014).

[7] S. Haar, (2012) Arduino and Android Powered Object Tracking Robot, [YouTube] https://www.youtube.com/watch?v=mw_gOjdyYI (23-01-2014).

[8] G. Samad, (2012) Way overkill line follower robot using Android smartphone video, [YouTube] https://www.youtube.com/watch?v=4POKLgpPwU8 (23-01-2014).

[9] G. Bradski, A. Kaehler, (2004) Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly.

[10] C. Poynton, (2003) Digital Video and HDTV: Algorithms and Interfaces, Elsevier.

[11] A. Sosio, (2006) HSV cone, http://commons.wikimedia.org/wiki/File:HSV_cone.png [Åtkomst 02-03-2014].

[12] T. Berk, A. Kaufman, L. Brownston, (1982) A human factors study of color notation systems for computer graphics. *C*ommunications of the ACM. Vol. 25, nr 8 sida 547–550.

[13] R. C. Gonzalez, (2008) Digital Image Processing. Upplaga 3, Pearson.

[14] OpenCV Developer Team, (2014) OpenCV, http://opencv.org/about.html (2014-03-10).

[15] Arduino, (2014) Arduino ADK, http://arduino.cc/en/Main/ArduinoBoardADK#.UxWr8oVbN8h (14-02-2014).

[16] A. Hughes, B. Drury, (2013) Electric Motors and Drives. Upplaga 4, Elsevier.

[17] I. Advanced Micros Systems, (2010) Stepper Motor System Basics, http://www.stepcontrol.com/pdf/step101.pdf, (09-02-2014).

[18] OEM Automatic AB, (2014) Allmän information ultraljudsgivare, http://www.oemautomatic.se/Produkter/Sensor/Ultraljudsgivare/Allman_information/Allman_information_ultraljudsgivare/541032-531495.html, (20-02-2014).

[19] Metric Industrial AB, (2014) Ultraljudsgivare, http://www.metric.se/produkter/sensorteknologi/industrisensorer/ultraljudsgivare/ (20-02-2014).

[20] I. Poole, (2014) Light dependent resistor, photo resistor, or photocell, http://www.radio-electronics.com/info/data/resistor/ldr/light_dependent_resistor.php (20-02-2014).

[21] J. De Witte, (2010) Mechatronic Design of a Soccer Robot for the Small-Size League of RoboCup, Master's thesis, Vrije Universiteit Brussel.

[22] R. Grahn, P. A. Jansson, (2007) Mekanik - statik och dynamik. Upplaga 2:5, Studentlitteratur.

[23] Android Development Team, (2012) Accessory Development Kit, http://developer.android.com/tools/adk/index.html (17-02-2014).

[24] R. C. Hibbler, (2007) Engineering Mechanics: Statics & Dynamics. Upplaga 11, Pearson.

[25] M. Böhmer, (2012) Baginning Android ADK with Arduino, Apress.

[26] T. D. Gillespie, (1992) Fundamentals of Vehicle Dynamics, SAE International.

[27] J. Adams, (2014) Raspberry PI Compute Module, http://www.raspberrypi.org/raspberry-pi-compute-module-new-product/ (10-05-2014).

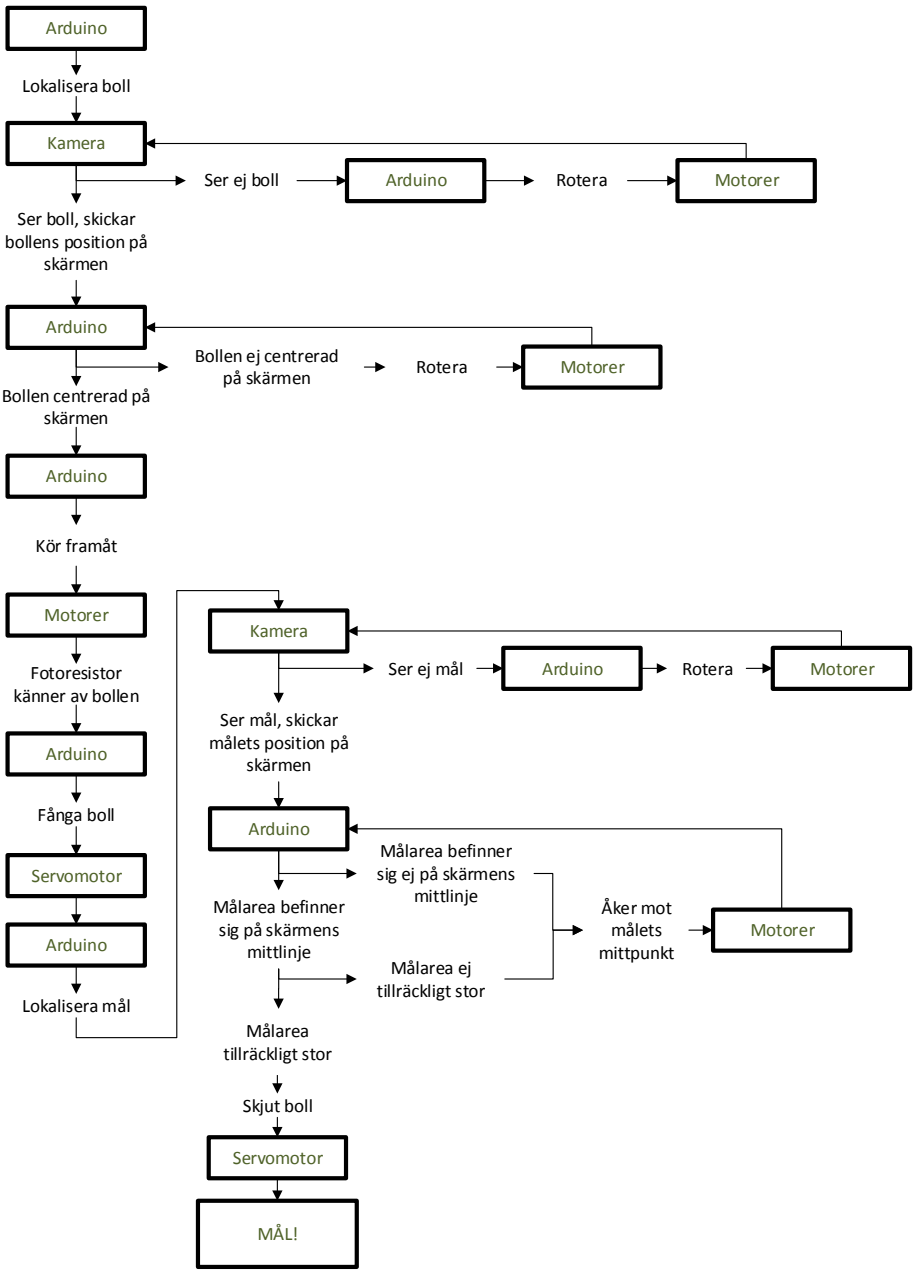
# Kravspecifikation

| Huvudfunktion | Krav/ Önskemål | Viktning | Mätbarhet | Enhet | Målvärde | Status | Resultat |
|---|---|---|---|---|---|---|---|
| Hitta givet objekt och föra till given plats | K | | | | | Godkänt | |
| **Stödjande funktioner** | | | | | | | |
| Åka rakt fram | K | | Mätning av avvikelse mellan start och slutpunkt | grader/meter | 15 | Godkänt | 0.050 |
| Åka rakt fram | Ö | 4 | Mätning av avvikelse mellan start och slutpunkt | grader/meter | 5 | Godkänt | 0.050 |
| Uppnå hastighet framåt | Ö | 5 | Mätning av hastighet | meter/sekunder | 0.5 | Ej godkänt | 0.154 |
| Accelerationssträcka till den önskade hast. | Ö | 4 | Mätning av sträckan | meter | 0.25 | Ej godkänt | - |
| Rotera visst antal grader | K | | Mätning av avvikelse mellan start och slutposition | grader fel/90 grader | 20 | Godkänt | < 5 |
| Rotera visst antal grader | Ö | 4 | Mätning av avvikelse mellan start och slutposition | grader fel/90 grader | 10 | Godkänt | < 5 |
| Lokalisera golfboll efter färg | K | | Tester | ja/nej | Ja | Godkänt | Ja |
| Åka fram till golfboll | K | | Tester | ja/nej | Ja | Godkänt | Ja |
| Transportera golfboll under rörelse | K | | Tester | ja/nej | Ja | Godkänt | Ja |
| Skjuta iväg golfbollen | K | | Tester | ja/nej | Ja | Godkänt | Ja |
| Skjuta golfbollen rakt fram | Ö | 4 | Mätning av avvikelse mellan start och slutpunkt | grader/meter | 10 | Godkänt | < 10 |
| **Tilläggsfunktion** | | | | | | | |
| Lokalisera mål efter färg | Ö | 3 | Tester | ja/nej | Ja | Godkänt | Ja |
| Skjuta bollen i mål | Ö | 3 | Tester | ja/nej | Ja | Godkänt | Ja |
| **Kostnad** | | | | | | | |
| Maximal kostnad för projekt från prototyplabbet | K | | Kostnadsberäkning | kronor | 2000 | Godkänt | - |
| Maximal kostnad för projekt utöver det | K | | Kostnadsberäkning | kronor | 5000 | Godkänt | ≈ 3100 |

1= minst viktigast
5 = viktigast

# B

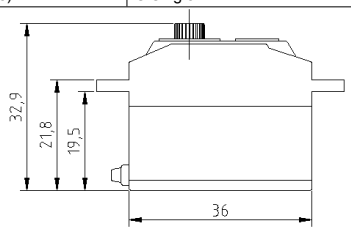
# BLOCKSCHEMA ÖVER SIGNALER

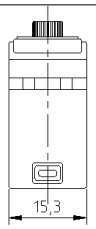


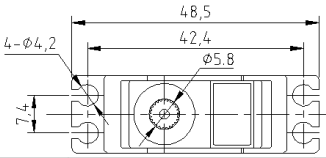
# C

# DATABLAD FÖR POSITIONSSERVO

| **GS-D9257**(1520 μ s) | | |
|---|---|---|
| Control System | Positive PWM control 1500 usec Neutral | |
| Operation Voltage Range | 4.8V ~ 6.0V | |
| Operation Temperature Range | -20C°~ +60C° | |
| Test Voltage: | At 4.8V | At 6.0V |
| Standing Torque | 3.8kg.cm | 4.2kg.cm |
| Speed | 0.08 sec / 60 deg at no load | 0.07 sec / 60 deg at no load |
| Idle Current | 180mA at stopped | 240mA at stopped |
| Running Current | 1400mA at no load | 1600mA at no load |
| Dead band width | 3 usec | 3 usec |
| Circle | 15000 times | |
| Operation Travel | 60° ± 10° | |
| Direction | Re-clock wise/ Pulse Travel 1500 to 1900 usec | |
| Drive type | FET drive | |
| Motor Type | Coreless motor | |
| Potentiometer Type | Indirect drive | |
| Amplifier Type | Digital Control | |
| Dimensions | 36.0*15.3*21.8 mm | |
| Weight: | 25g without splined horn | |
| Ball Bearing | Double ball bearing | |
| Gear Material | POM | |
| Case Material | Nylon plus fiber | |
| Connector Wire Length | 300mm | |
| Connector Wire Gauge | 26AWG heavy duty（JR Universal） | |
| Wire Info (Brown) | Negative | |
| Wire Info (Red) | Positive | |
| Wire Info (Orange) | S Single | |



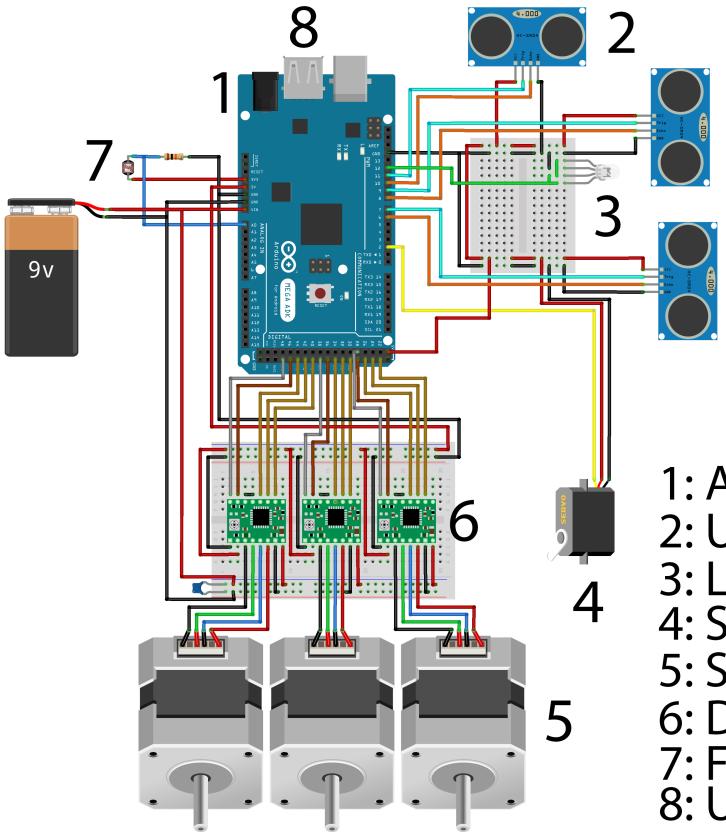

规格
尺寸：长 36mm
宽 15.3mm
高 21.8mm
重量：25g

| GOTECK RC INC | GS-D9257(**1520 μ s**) | |
|---|---|---|

Web:www.goteckrc.com updated:2010-09-01

# D

# KOPPLINGSSCHEMA




1: Arduino Mega ADK
2: Ultraljudssensor
3: LED-diod
4: Servomotor
5: Stegmotorer
6: Drivkort
7: Fotoresistor
8: USB-anslutning