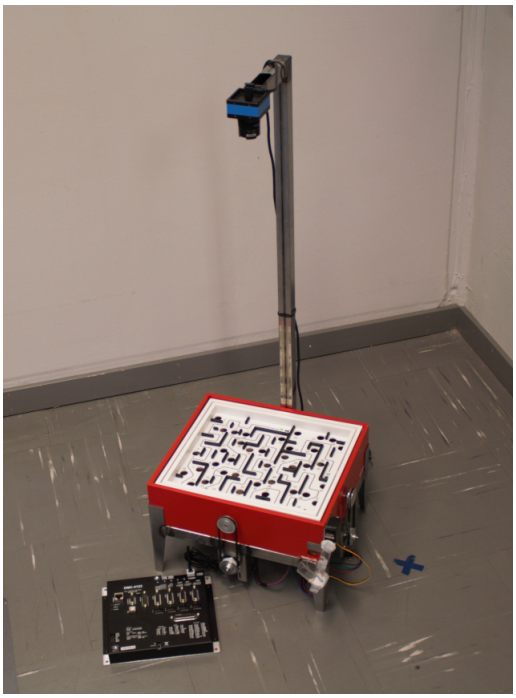
CHALMERS



# Självspelande Labyrintspel

*Kandidatarbete vid institutionen för Signaler och System*

Joel Heinerud
Karl-Johan Hagelin
Lars Kallryd
Oskar Jonson
Karl Tageman
Olof Hill

Chalmers Tekniska Högskola
Göteborg, 2014

# Tack till

För att projektet ska ha varit möjligt att genomföra har hjälp och stöd från flera personer behövts. Vi vill först och främst tacka vår handledare Bo Egardt för hans stöd och entusiasm under projektets fortgång. Tack till Institutionen för Signaler och System för tillhandahållande av material och lokaler.

Ett extra tack riktas till Compotech som bistått med finare motorer och drivsteg än budgeten tillät.

Tack också till alla övriga som varit mer eller mindre involverade i projektet.

## Abstract

This thesis covers the development of an automated marble maze. A ball is guided through the maze by tilting the maze around its two axes. To actualize this, a camera locates the ball and the path which the ball is supposed to follow. MATLAB processes the signals from the camera and generates values that can be used as process values and setpoints. MATLAB then calculates the desired tilt angles that the maze is supposed to have using a PID regulator. These values are then forwarded to a motor driver which is controlling two stepper motors connected to its corresponding axis at the maze. The project resulted in a prototype, which also was simulated in Simulink.



## Sammandrag

Denna rapport behandlar utvecklingen av ett automatiserat labyrintspel. En kula styrs genom en labyrint genom att luta labyrinten kring dess två axlar. För att lösa detta används en kamera för att lokalisera kulan och den bana som skall följas. Två motorer används för att styra labyrinten. MATLAB används för att bearbeta informationen från kameran så att är- och börvärde kan genereras. Även implementeringen av regulatorn sker i MATLAB, som sedan skickar ut önskade vinklar till drivsteget. Drivsteget skickar sedan ut styrsignaler till motorerna som ställer in labyrintspelet till önskad vinkel. Projektet resulterade i en prototyp för demonstration. Detta system simulerades även i Simulink.

# Innehåll

# 1 Inledning

Labyrintspelet är en riktig klassiker. Målet är att leda en kula genom en labyrint och samtidigt undvika de hål som spelplanen innehåller. Den riktning som kulans hastighet har, bestäms genom att vinkla hela labyrinten med hjälp av två reglage på labyrintens sidor. Detta är en utmanande uppgift för en människa och det är därför spännande att se om det går att bygga ett system som kan lösa labyrinten utan mänsklig interaktion.

*Figur 1.1: BRIO:s labyrintspel med två extra banor*

## 1.1 Bakgrund

Ett självspelande labyrintspel har många intressanta utmaningar. En av dessa uppkommer av att det endast krävs en liten vinkel för att få kulan att accelerera längs planet. Spelet är designat på detta sätt för att vara svårt. Därför utgör det en utmärkt plattform för att demonstrera vad som kan åstadkommas med hjälp av reglerteknik och mekatronik. Implementering av reglertekniska system gör att många problem kan lösas på ett snabbare och stabilare sätt än vad en människa klarar av. Regleringen av spelet kommer också att påvisa möjligheterna att skapa stabilitet i dynamiska system, något som är viktigt både inom industri och elektronikutveckling.

Ett annat ingenjörsmässigt problem som uppkommer när man automatiserar ett labyrintspel är hur kulans position ska lokaliseras. I projektet har "vision based control" använts för att lösa detta problem. Detta innebär att en kamera nyttjas för att lokalisera kulan och läsa in labyrintbanan. Denna information används sedan för att beräkna styrsignaler till de elmotorer monterade på labyrintens reglage. Samma problematik uppkommer i flera industriapplikationer, exempelvis industrirobotar med bildanalys.

## 1.2 Syfte

Syftet med projektet är att med hjälp av reglerteknik utforska möjligheterna att stabilisera och automatisera ett labyrintspel. Det skall även öka projektgruppens kunskaper inom reglerteknik, signalbehandling samt mekatronik.

## 1.3 Mål

Målet med projektet är att tillverka en fungerande prototyp som klarar av att att lösa labyrintspelet utan mänsklig interaktion. Detta skall ske helautomatiskt och med repeterbarhet. Systemet skall även kunna lösa flera olika labyrinter vilket uppnås genom att implementera ett generellt styrsystem. Ett sådant system skulle också leda till att spelet löses även om kulan läggs på en godtycklig punkt i labyrinten. Resultatet skall kunna användas i utbildningssyfte, eller för att öka intresset för reglerteknik på utställningar, mässor och liknande evenemang.

## 1.4 Avgränsningar

Projektet hade en budget på 5000 kr som inte fick överstigas. Projektet hade även en tidsbudget som uppgick till 400 timmar per gruppmedlem uppdelat på två läsperioder á 8 veckor. Allt arbete genomfördes inom denna tidsram och slutfördes under den andra läsperioden. Komplexitet i projektet har anpassas till tidsbudgeten.

Projektet avgränsades även med avseende på komponenter. Motorer, sensorer och labyrintpel köptes in och konstruerades inte av projektgruppen. Tidigt i projektet var tanken att ett helt omodifierat spel skulle automatiseras, men senare i projektet gjordes mindre justeringar.

Eftersom gruppen hade för avsikt att bygga en prototyp för enstaka demonstrationer har inte faktorer som livslängd, tillverkningsbarhet, produktionskostnad och massproducerbarhet behandlas i projektet.

# 2 Metod

För att påvisa att syfte och mål är uppfyllda konstruerades en prototyp. För att realisera denna simulerades först en modell av systemet. Denna användes för att testa och utvärdera systemet och dess reglertekniska egenskaper. Tester och utvärderingar har gjorts löpande genom hela projektet. Olika parametrar har testats parallellt och de som fungerade sparades för utvärdering.

## 2.1 Simulering och regulatordesign

Simuleringar genomfördes i Simulink. Först analyserades alla dynamiska processer i systemet noggrant och överföringsfunktioner togs fram till respektive process. Detta gjordes genom att undersöka labyrintspelet och dess ingående delar. Med de matematiska modellerna som togs fram realiserades sedan Simulinkmodellen. Från simuleringen togs viktiga parametrar ut, vilka användes vid dimensioneringen av prototypens komponenter. Dessutom användes simuleringen till att beräkna regulatorparametrar som sedan implementerades i det verkliga systemet.

## 2.2 Sensorval

Att lokalisera kulan är ett av de viktigaste delsystemen. För att göra detta krävs någon slags sensor. Efter analys av olika alternativ togs beslutet att använda en kamera som sensor. Detta eftersom den har egenskapen att kunna lokalisera både kulan och banan som kulan skall följa. Eftersom ett mål är att klara en godtycklig labyrint så är en kamera fördelaktig även i det fallet eftersom en kamera kan analysera vilken bana som helst.

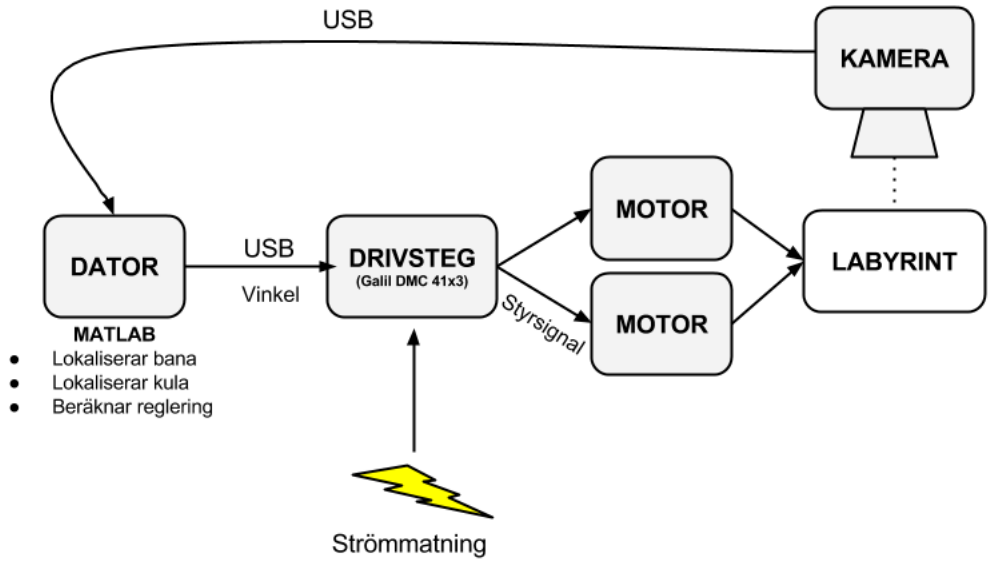
## 2.3 Testning

Många tester och experiment har utförts på prototypen, där resultatet jämförts med simuleringen. Detta för att säkerställa resultatet och för att kunna jämföra verkligheten med simuleringen. Detta är intressant för gruppen ur ett utbildningsperspektiv eftersom en analys kan utföras om hur nära verkligheten simuleringen är.

# 3 System

För att reglera labyrintspelet krävs ett antal komponenter. Kopplingen mellan dessa kan ses i Figur 3.1.

Vid uppstart av systemet begär datorn en bild av kameran för att kunna lokalisera rutten som kulan skall färdas utmed. Rutten sparas sedan som en vektor av börvärden. När rutten är fastställd börjar datorn att kontinuerligt uppdatera kulans position. Positionen jämförs sedan med den aktuella börvärdespositionen och datorn skickar sedan en styrsignal till drivsteget, som i sin tur översätter signalen till motsvarande spänningspulser som matas till motorerna. Börvärdet uppdateras i takt med att kulan färdas längs rutten.



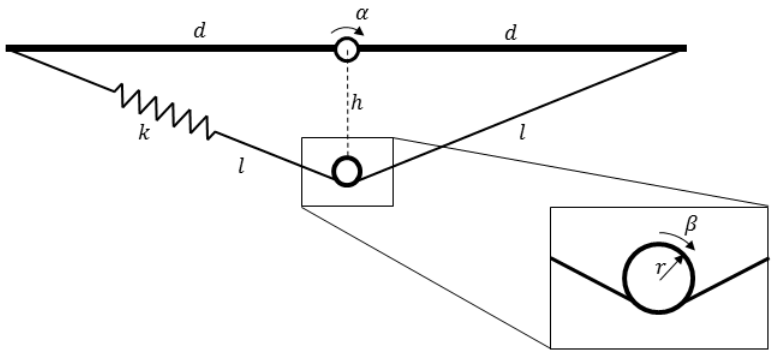

*Figur 3.1: En schematisk bild över systemet*

# 4 Matematiska modeller

För att kunna genomföra de simuleringar som beskrivs i Kapitel 6 Simulering har systemet analyserats och beskrivits matematiskt. De modeller som tagits fram beskriver tillsammans hur en ingående vinkel på vredet ger en acceleration på kulan.

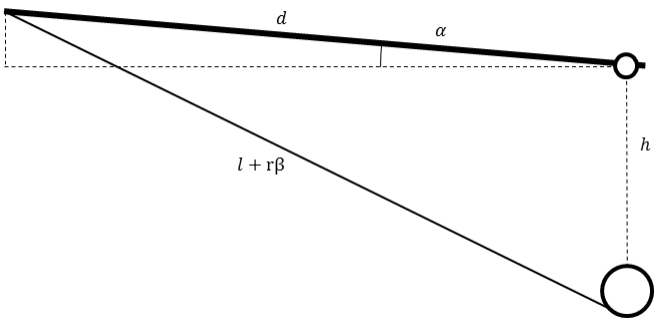
## 4.1 Brädets dynamik

Innan inköpet av labyrinten uppskattades brädets matematiska modell till en enkel utväxling mellan reglagens och brädets vinkel. När labyrinten levererades upptäcktes dock att brädets dynamik var betydligt mer komplex. Denna insikt ledde till att brädets dynamik behövde undersökas och beräknas för att kunna göra en så representativ simulering som möjligt. Lutningen på brädet styrs genom att vrida på reglage vilka har snöre lindat runt reglagens axlar. Dessa snören har sina ändar fastsatta i brädet. Detta leder till att en vinkeländring på reglagen gör att en bit snöre lindas av på ena sidan axeln och att snöret blir kortare på andra sidan. Det är denna längdskillnad som gör att brädet lutar. Dynamiken försvåras ytterligare av att det i snörena sitter fjädrar som används för att hålla dessa spända. Fjädrar skapar en fjäderdynamik i systemet. Fjäderdynamiken gör att en snabb vinkelförändring på reglaget skapar en oscillation på brädet (se Figur 4.4) innan det stabiliserar sig vid önskad vinkel. I Figur 4.1 är brädets geometri illustrerat när brädet är horisontellt.




*Figur 4.1: Figur över hur brädets geometri ser ut. Överst ser man brädet i profil och dess rotationspunkt. Nedanför i den förstorade bilden syns axeln som roteras av spelaren.*

Kraftöverföringen från reglage till bräde har en komplicerad geometri vilket försvårar beräkningarna. Därför approximerades vinklarna som små och därmed blir $\sin\alpha \approx \alpha$ och $\cos\alpha \approx 1$. Samma förenkling gjordes för $\beta$. Först beräknades ett samband mellan $\alpha$ och $\beta$ vid stationärtilstånd.




*Figur 4.2: Approximation av brädets läge vid små vinklar.*

Från Figur 4.2 fås ett samband mellan den nya längden på snöret och vinkeln $\alpha$:

$$(l + r\beta)^2 = h^2 + d^2 - 2hd\cos\left(\frac{\pi}{2} + \alpha\right) \tag{4.1}$$

Med vetskapen att $\cos\left(\frac{\pi}{2} + \alpha\right) = -\sin\alpha$ och approximationen för små vinklar kan Ekvation 4.1 skrivas om enligt

$$(l + r\beta)^2 \approx h^2 + d^2 + 2hd\alpha \tag{4.2}$$

Nu kan $h^2 + d^2$ brytas ut till följande ekvation:

$$(l + r\beta)^2 \approx \left(h^2 + d^2\right)\left(1 + \frac{2hd}{h^2 + d^2}\alpha\right) \tag{4.3}$$

Därefter tas kvadratroten av båda leden och den högra parantesen i högerledet taylorutvecklas enligt

$$l + r\beta \approx \sqrt{h^2 + d^2}\left(1 + \frac{hd}{h^2 + d^2}\alpha\right) \tag{4.4}$$

Från Figur 4.1 kan sambandet $l = \sqrt{h^2 + d^2}$ fås vilket slutligen ger ett stationärt förhållande mellan $\beta$ och $\alpha$ enligt

$$r\beta = \frac{hd}{\sqrt{h^2 + d^2}}\alpha \tag{4.5}$$

Med denna ekvation kan en linjär fjäderkraft approximeras. Kraften beror av avvikelsen från stationärläget, det vill säga vid skiljetecken i Ekvation 4.5. Denna fjäderkraft bildar då ett moment kring brädets axel vilket ger en acceleration $\ddot{\alpha}$ enligt

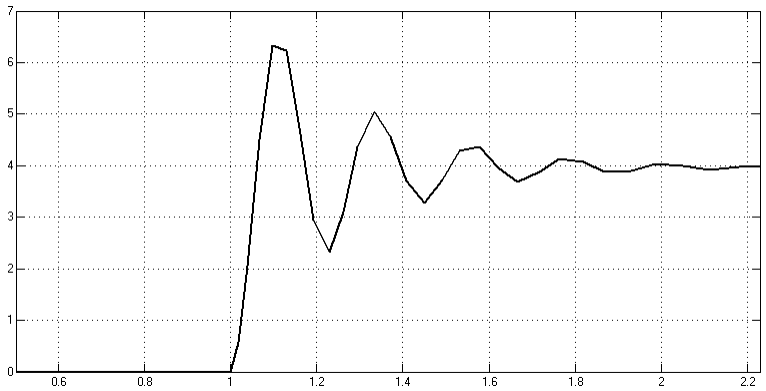
$$I_\alpha\ddot{\alpha} = kd\left(r\beta - \frac{hd\alpha}{\sqrt{h^2 + d^2}}\right) \tag{4.6}$$

Denna differentialekvation saknar dock dämpning. Med en enkel approximation kan detta läggas till i modellen som en konstant multiplicerat med $\dot{\alpha}$. Den totala differentialekvationen blir då

$$I_\alpha \ddot{\alpha} + c\dot{\alpha} + \frac{khd^2\alpha}{\sqrt{h^2+d^2}} = kdr\beta \tag{4.7}$$

Den slutgiltiga överföringsfunktionen från insignal $\beta$ till utsignal $\alpha$ fås då enligt 4.8 och ger stegsvaret som illustreras i Figur 4.3.

$$G(s) = \frac{kdr}{I_\alpha s^2 + cs + \frac{khd^2}{\sqrt{h^2+d^2}}} \tag{4.8}$$

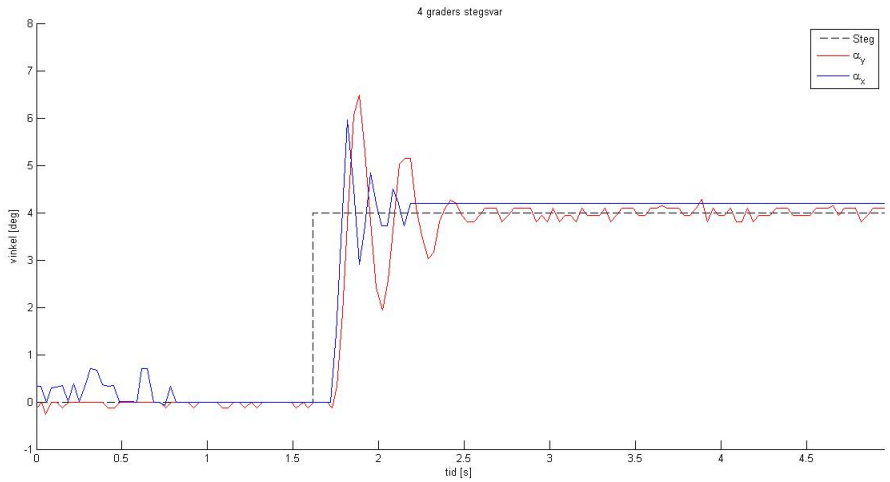



*Figur 4.3: Ett 4 graders stegsvar av brädvinkeln från simuleringen.*

### 4.1.1 Analys av bräddynamik

För att analysera bräddynamiken gjordes tester på prototypen. Testerna gick ut på att mäta upp ett stegsvar på prototypen och jämföra med ett stegsvar från simuleringen. Detta gjordes för att approximera fjäder- och dämpningskonstanten. För att kunna analysera stegsvaret för prototypen klistrades markörer, i form av gröna papperslappar på brädets hörn. Kameran användes sedan för att bestämma de gröna områdenas koordinater. Genom att analysera hur dessa koordinater förvrängs i bilden kan vinkeln på brädet sedan beräknas.

Testet gick till så att vinkeln på brädet loggades i realtid. Efter ett antal samplingar gav MATLAB ett kommando att vinkla brädet 4 grader. Testet utfördes först på y-axeln och sedan på x-axeln. Samplingarna plottades sedan ovanpå varandra och resultatet kan ses i Figur 4.4

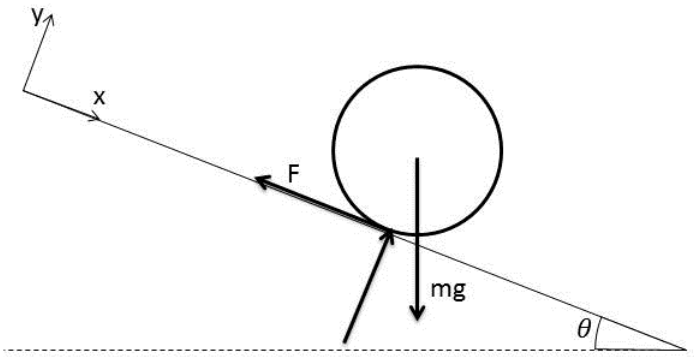



*Figur 4.4: Stegsvar av brädvinkeln från den verkliga prototypen med fyra graders lutning. $\alpha_y$ är vinkeln kring y-axeln och $\alpha_x$ kring x-axeln*

En iakttagelse från Figur 4.4 är att $\alpha_y$ har kraftigare översläng och längre insvängningstid än $\alpha_x$. Det kommer av att labyrintens fjädersystem är mindre spänd kring y-axeln och även att brädets tröghetsmoment är större kring den axeln eftersom ramen för båda axlarna är upphängd i y-axeln. I detta test syns också dödtiden som finns i systemet. Dödtiden uppskattades i testet till 0.1 sekunder. Dödtiden innefattar tiden det tar från att MATLAB skickar kommandot att ändra vinkeln på brädet, tills det att kameran uppfattar att brädet har börjat röra på sig.

## 4.2 Kulans dynamik

För att veta hur kulan påverkas av en viss vinkel på brädet behövs en beskrivande modell. När denna togs fram försummades rullmotståndet mellan kulan och brädet. Detta eftersom det ansågs ha liten effekt, samtidigt som det fanns störningar i systemet som hade större inverkan på kulan än rullmotståndet. Bland annat är brädets yta vågigt med varierande ytskrovlighet. Det finns även ojämnheter från målade linjer. En friläggning över kulan med försummat rullmotstånd genomfördes enligt Figur 4.5. Här är $N$ normalkraften som verkar på kulan och $m$ kulans massa. $g$ är tyngdaccelerationenen och $\theta$ är brädets vinkel.




*Figur 4.5: Friläggning av kulan på ett lutande plan*

Jämvikten i $y$-led blir då:

$$0 = N - mg\cos\theta \tag{4.9}$$

samt i $x$-led

$$m\ddot{x} = mg\sin\theta - F \tag{4.10}$$

Kraften $F$ i jämvikten får kulan att rotera, denna har ett samband med kulans tröghet som kan tecknas:

$$F = \frac{I_z\dot{\omega}}{r} \tag{4.11}$$

$F$ substitueras i Ekvation 4.10 med Ekvation 4.11. Då fås:

$$m\ddot{x} = mg\sin\theta - \frac{I_z\dot{\omega}}{r} \tag{4.12}$$

Här är $I_z$ tröghetsmomentet, som för en kula är:

$$I_z = \frac{2}{5}mr^2 \tag{4.13}$$

Tröghetsmomentet från Ekvation 4.13 sätts in i Ekvation 4.12. Massan kan nu förkortas och detta ger:

$$\ddot{x} = g\sin\theta - \frac{2}{5}r\dot{\omega} \tag{4.14}$$

Då $\dot{\omega}$ är vinkelaccelerationen och $r$ kulans radie blir produkten av de två kulans acceleration, $\ddot{x}$. Detta kan nyttjas till Ekvation 4.14

$$\ddot{x} = g\sin\theta - \frac{2}{5}\ddot{x} \tag{4.15}$$

Förenklas 4.15 fås:

$$\ddot{x} = \frac{5}{7}g\sin\theta \tag{4.16}$$

För små vinklar $\theta$ fås att $\sin(\theta) \approx \theta$, med detta sambandet fås en linjär approximerad modell över kulans dynamik:

$$\ddot{x} = \frac{5}{7}g\theta \tag{4.17}$$

För att kunna ta fram överföringsfunktionen Laplacetransformeras Ekvation 4.17 vilket ger

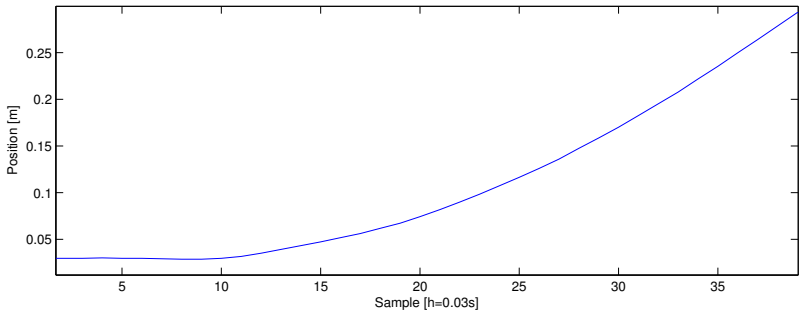
$$s^2X(s) = \frac{5}{7}g\theta(s) \tag{4.18}$$

Kulans överföringsfunktionen $G(s)$ ges av kvoten utsignal/insignal. Då Ekvation 4.18 skrivs om på detta sätt erhålls:

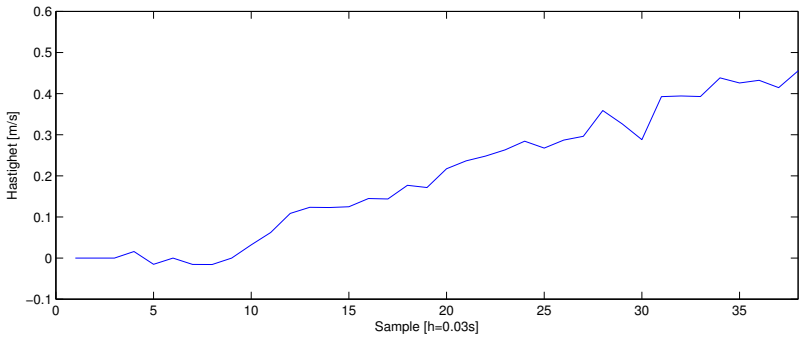
$$G(s) = \frac{X(s)}{\theta(s)} = \frac{5g}{7s^2} \tag{4.19}$$

### 4.2.1 Verifiering av kulans dynamik

Ekvation 4.17 verifierades genom ett experiment med prototypen. Ett MATLAB-script skrevs där kulans position loggades mot tiden. Resultatet visade att den uppmätta accelerationen stämde mycket väl överens med accelerationen framräknad med Ekvation 4.17. Med vinkeln 4 grader beräknades accelerationen 0.49 $m/s^2$, medan experimentet mätte upp en acceleration på 0.47 $m/s^2$. I experimentet togs accelerationen fram under ett intervall mellan sample 15 och 40. Detta eftersom det finns en viss vilofriktion som gör att kulans acceleration är lägre i starten, något som syns tydligt i Figur 4.6.




*Figur 4.6: Kulans position vid 4 graders vinkel*




*Figur 4.7: Kulans hastighet vid 4 graders vinkel*
*Grafernas x-axlar visar sample som enhet, varje sample motsvarar 0.03s*

# 5 Regulatordesign

För att lutningen av brädet ska ske korrekt vid styrning av kulan genom labyrinten krävs en regulator. Regulatorn ger även ett stort bidrag vid kompensering för systemstörningar. Vid en första implementering av regulator i modellen användes en PID-regulator. Eftersom kulans process är en ren dubbelintegrator och själv bidrar till stor integratorverkan bedömdes dock I-delen i regulatorn vara liten. Vidare tester visade också på att I-delen var försumbart liten, varefter en mer simpel PD-regulator med fördel kunde väljas utan att resultatet försämrades.

Regulatorns prestanda beror dels av dess parametrar, men även på metoden som börvärdet förändras med.

## 5.1 Parametrar

Formeln för PD-regulatorn med ett första ordningens filter fås enligt

$$F_{PD}(s) = \frac{\omega_c^2\left(1 + \frac{s}{\omega_c/\sqrt{b}}\right)}{\sqrt{b}K\left(1 + \frac{s}{\omega_c\sqrt{b}}\right)} \tag{5.1}$$

Där K är processesförstärkningen som ges av $5g/7$. Parametern $b$ beräknas enligt

$$b = \frac{1 + \sin\varphi_m}{1 - \sin\varphi_m} \tag{5.2}$$

Överkorsningsfrekvensen $\omega_c$ och fasmarginalen $\varphi_m$ ställs sedan in tillsammans med börvärdesgenereringen för att uppnå önskat beteende.

Vid simuleringens första steg användes en kontinuerlig regulator enligt Ekvation 5.1. Denna behöver sedan diskretiseras för att kunna användas i det verkliga systemet. Detta görs med hjälp av MATLAB-komandot `c2d`. Samplingstiden 0.04 sekunder användes vid diskretiseringen. Den diskretiserade regulatorn med $\omega_c = 4.5$ $rad/s$ och $\varphi_m = 65\,^\circ$ fås enligt

$$F_{PD}(z) = \frac{13.02z - 12.67}{z - 0.444} \tag{5.3}$$

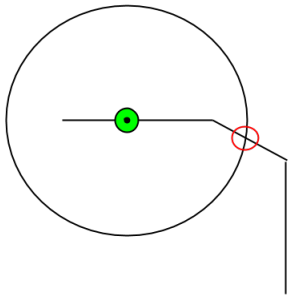
## 5.2 Börvärdesgenerering

Rutten som kulan ska följa är sparad i en vektor med koordinater. Dessa koordinater används som dynamiskt börvärde, där det finns olika metoder att förändra det. Denna börvärdesgenerering är starkt kopplad till valet av regulator och regulatorparametrar. Om börvärdesgenereringen är restriktiv, det vill säga att börvärdet håller sig nära kulan, så kommer ett litet fel att mätas upp. Har man däremot en friare börvärdesgenerering, där börvärdet är längre bort från kulan, fås ett större fel. Detta kommer att förändra hur snabb och robust regulatorn måste vara för att ge bra prestanda. Fyra metoder utformades för börvärdesgenerering.Alla metoder har gemensamt att de, när ett villkor är uppfyllt, byter börvärdet till nästa i listan. Nedan följer de olika villkoren.

1. Om kulans position är inom en viss radie ifrån nuvarande börvärde
2. Om arean mellan ärvärde, börvärde och linjen som kulan borde följa är mindre än ett visst värde samt att kulans position är inom en viss radie ifrån nuvarande börvärde
3. Om en viss tid har passerat, alltså att flytta fram börvärdet med konstant hastighet
4. Om vinkeln mellan en vektor från kulans position till en punkt på börvärdeslinjen, och en vektor av börvärden i kulans riktning är tillräckligt liten.

### 5.2.1 Börvärdesgenerering med en radie runt kulan

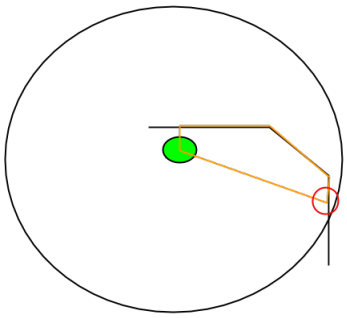
Denna metoden är snabb och håller sig mycket bra på linjen vid raksträckor men får en kraftig översväng vid skarpa svängar.



*Figur 5.1: Den gröna cirkeln symboliserar kulan. Den röda ringen är börvärdet som flyttas fram tills det ligger utanför den stora cirkeln*

### 5.2.2 Börvärdesgenerering med area och och radie

Tillägget av arean leder till att kulan rör sig lite långsammare än med bara en radiebegränsningen. Den följer också lite sämre vid raksträckor men har i gengäld inte lika kraftig översläng vid skarpa kurvor. Eftersom det ofta är hål strax efter en skarp kurva ger det ett mer önskvärt beteende.



*Figur 5.2: Den gröna ringen är kulan. Börvärdet (den röda ringen) flyttas fram tills det ligger utanför den stora cirkeln eller tills arean innanför den orangea linjen är mindre än ett valt värde*

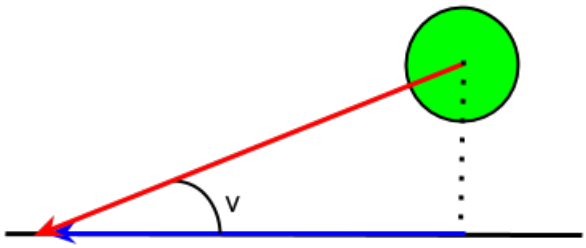
### 5.2.3 Börvärdesgenerering med konstant förflyttning

Denna metod fungerade bäst i simuleringen. Den var även helt överlägsen i prototypen. Detta på grund av att kulan som används inte rullar bra vid låga vinklar. Med metoden flyttas nämligen börvärdet även om kulan ligger stilla, felet växer då i takt med börsvärdets förflyttning. Detta leder till att vinkeln ökar och att kulan till slut börjar rulla. Metoden minimerar även risken att kulan stannar, något som inte är önskvärt på grund av dess vilofriktionen.

Nackdelen med denna metod i prototypen är dock att det finns en risk att börvärdet kommer för långt ifrån kulan. Detta leder till en ökad risk för att kulan fastnar bakom en vägg eller genar för mycket och faller ner i ett hål

### 5.2.4 Börvärdesgenerering med vektor

Denna metod fungerade bra på raksträckorna och hade liten översvängning. Metoden var dock långsam i kurvorna då vinklen mellan de två vektorerna genast blev stor vid skarpa svängar.




*Figur 5.3: Börvärdet flyttas fram så länge vinkeln mellen de två vektorerna (röd respektive blå) är tillräckligt liten*

# 6 Simulering

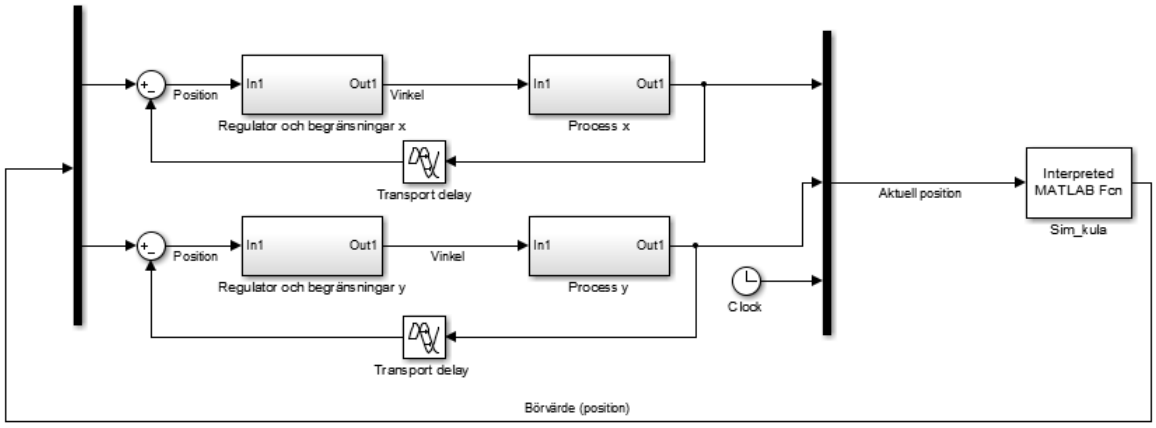
Simuleringen av labyrintspelet användes för att ta fram värden på regulatorparametrar samt att undersöka känslighet för följande fenomen:

- Tidsfördröjning i styrsystemet
- Motorernas maximala vinkelhastighet
- Diskontinuiteter uppkomna ifrån börvärdesgenereringen
- Brädets dynamik

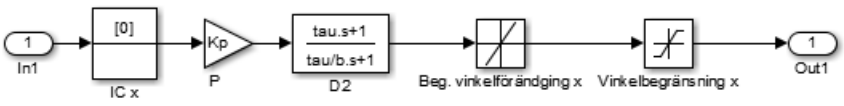
## 6.1 Simulink-modell

Modellen av systemet byggdes i Simulink enligt Figur 6.1 och undersystemen *Regulator och begränsningar* och *Process* visas i Figur 6.2 respektive Figur 6.3. Ett positionsbörvärde skickas från en MATLAB-funktion till regulatorn för respektive axel. Regulatorn beräknar en vinkel för var axel som brädet ska erhålla. De önskade vinklarna överförs via bräddynamiken till kulans modell. Kulans modell återkopplar sedan kulans position till regulatorn men skickar även den vidare till MATLAB-funktionen som beräknar ett nytt börvärde.

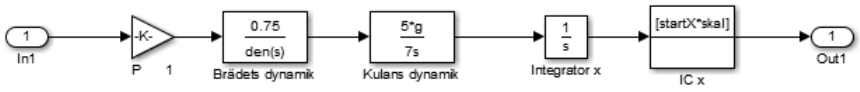
Med tillgång till uppgifter om brädets maximala vinkelutslag och motorernas snabbhet kan systemet anses vara simulerat med tillräckligt hög precision. Det maximala vinkelutslaget på brädet uppmättes med hjälp av en digital accelerometer. Motorernas snabbhet uppskattades med hjälp av tillverkarens datablad.




*Figur 6.1: Modell över systemet*

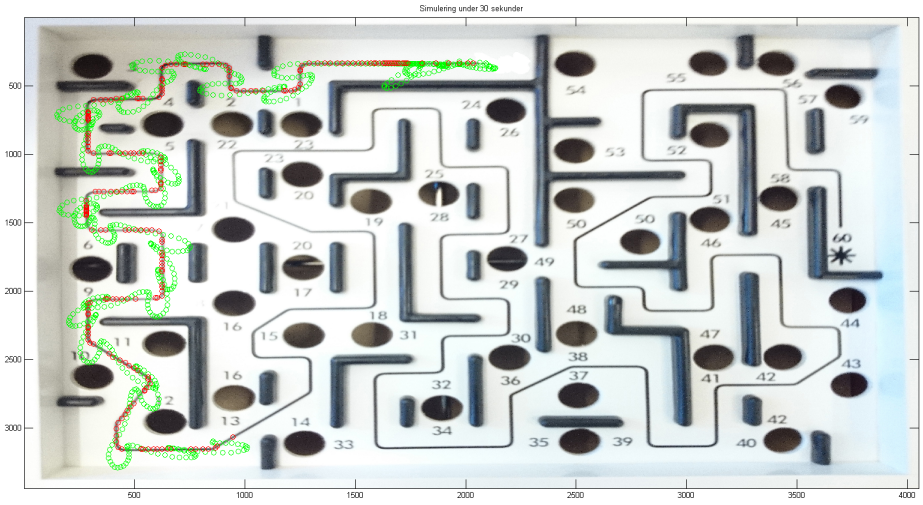



*Figur 6.2: Modell över regulator*

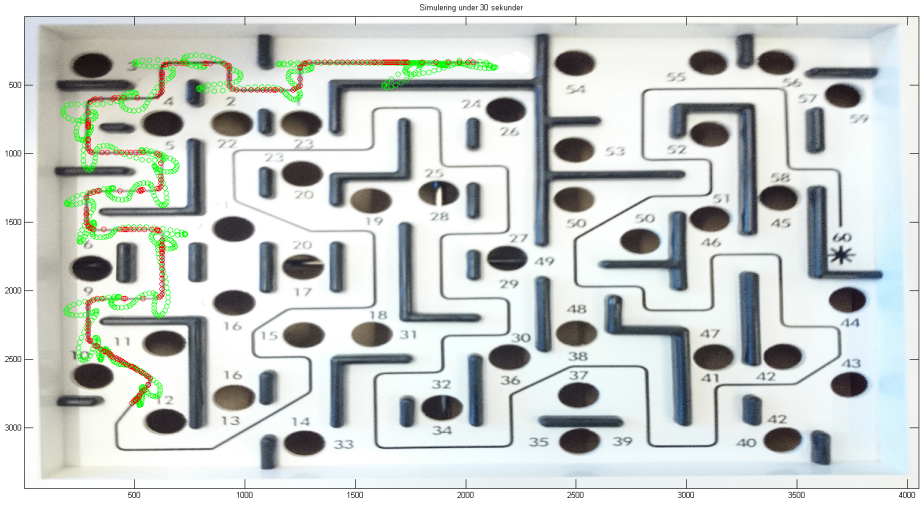



*Figur 6.3: Modell över processen*

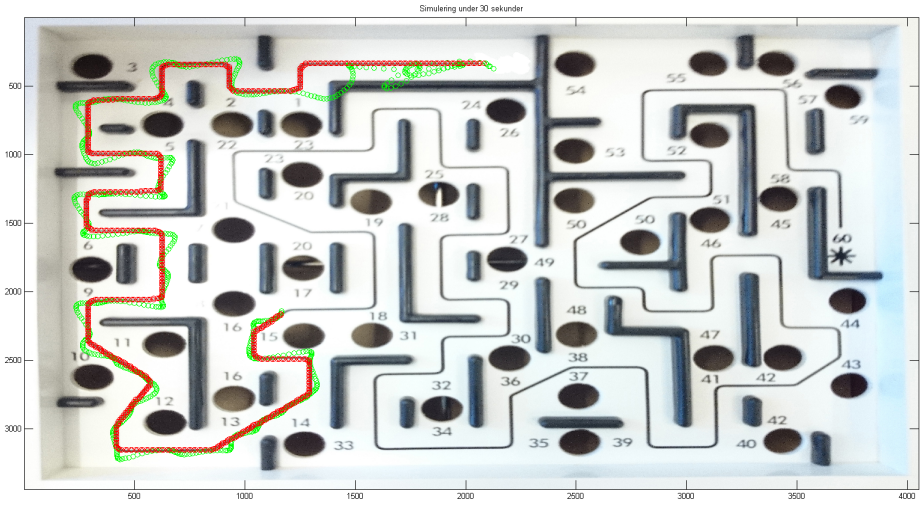
Simuleringsresultat under 30 sekunder visas i Figur 6.4, 6.5, och 6.6. Samtliga simuleringarna är framtagna med $\omega_c = 4.5\ rad/s$ och $\varphi_m = 65\,°$. Kulans positioner representeras av de gröna punkterna och börvärdena representeras av de röda punkterna. Ett stegsvar för systemet visas i Figur 6.7.



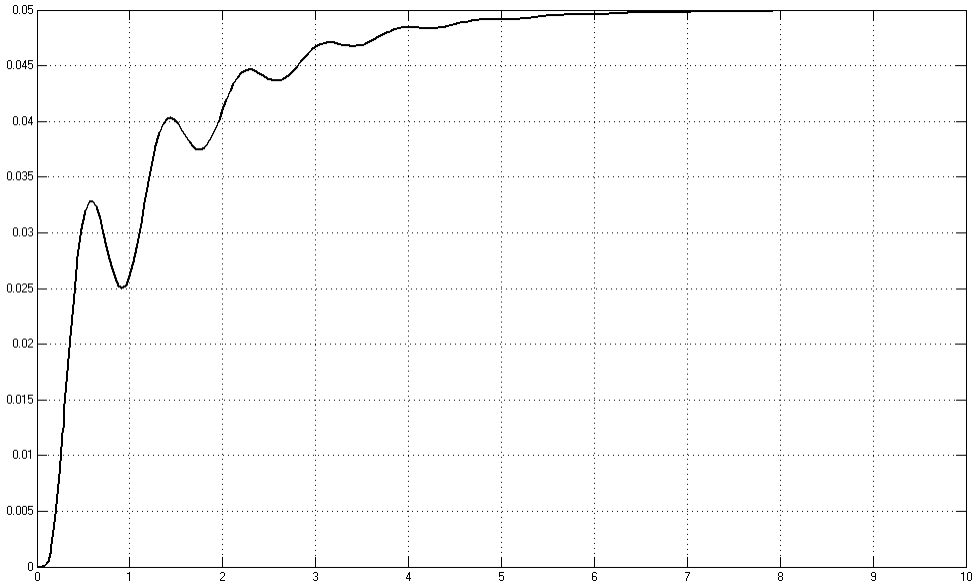

*Figur 6.4: Simulering av systemet med radiebaserad börvärdesframtagning*




*Figur 6.5: Simulering av systemet med radie och areabaserad börvärdesframtagning*



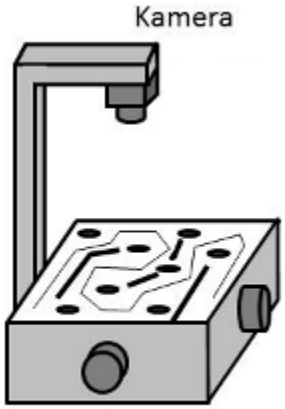

*Figur 6.6: Simulering av systemet med konstant hastighet på börvärdesframflyttningen*




*Figur 6.7: Systemets beteende vid ett steg på 5 cm*

# 7 Kamera som sensor

Kameralokalisering antogs snabbt som det enda rimliga alternativet för att lokalisera kulan och rutten. Skälet till det är att en kamera är den enda sensor som både kan läsa in rutten och lokalisera kulan utan att modifiera brädet nämnvärt. Dessutom är dagens kameror relativt billiga.




*Figur 7.1: Schematisk figur över kamerans placering*

## 7.1 Val av kamera

Kameran som används till projektet är en industrikamera från företaget Imaging Source. Kameran har en upplösning på $640 \times 480$ pixlar. Eftersom spelbrädet mäter $270 \times 228$ mm ger kameran en precision på 228/480 = 0.48 mm/pixel. Denna upplösning bedöms ge tillräckligt hög precision samtidigt som beräkningsarbetet hålls på rimliga nivåer. Uppdateringsfrekvensen på kameran är 60 Hz vilket är mer än tillräckligt då regulatorns uppdateringsfrekvens är 25 Hz. Kameran kommer med MATLAB-kompatibla drivrutiner vilket gör den lämplig för vårt ändamål.

## 7.2 Lokalisering av kula

Bilderna från kameran användas för att lokalisera kulans nuvarande position i realtid. Hädanefter kallas detta ärvärde. Det är viktigt att koden är snabb och hinner med att identifiera objekt i bilden, då en fördröjning i återkopplingen försvårar den reglertekniska konstruktionen och riskerar att göra systemet instabilt. För att hitta kulan på brädet används MATLAB och Computer Vision System Toolbox, Image Processing Toolbox samt Image Acquisition Toolbox$^{TM}$som är integrerade med MATLAB.

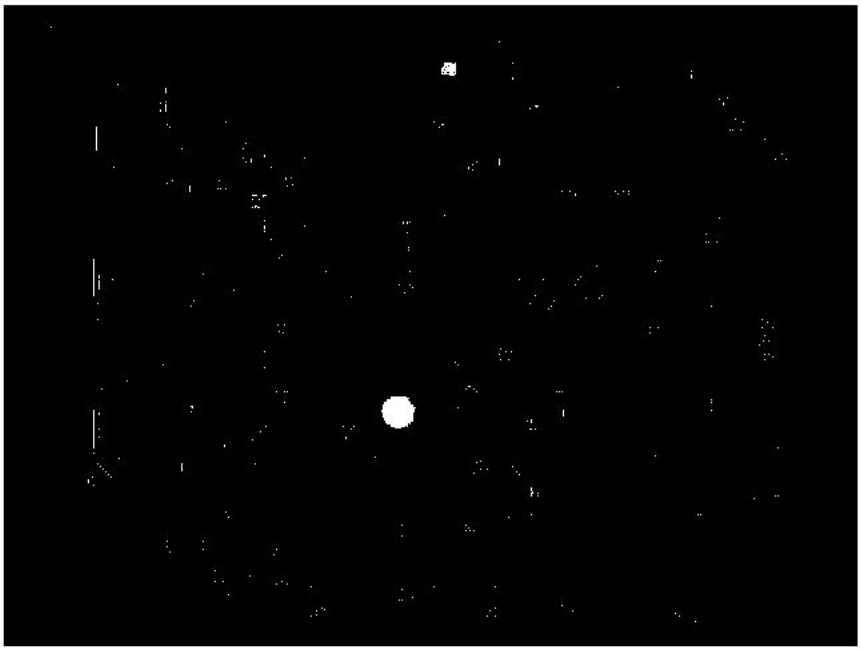
För att lokalisera kulan i en kontinuerlig bildström från kameran har olika inbyggda funktioner i MATLAB använts. Det program som skrivits behandlar bilderna en åt gången i den ström av bilder som kameran skickar till datorn. Bilderna kommer till datorn i RGB-format. Detta format fungerar så att varje pixel är beskriven av tre värden. Dessa tre värden är R (rött) G (grönt) och B (blått). I MATLAB har dessa parametrar värden mellan 0 och 1. Till exempel så är alla tre värdena ett när pixeln är vit och noll när pixeln är svart. Med dessa värden kan man beskriva alla möjliga färgkombinationer.

Tester utfördes på kameran i systemet för att undersöka färginnehållet i bilderna. Dessa tester visade att vanligt ljus från lampor innehåller mycket blå färg. Detta ljus reflekteras i den vita bakgrunden på banan och

ger ett blått skimmer som kan förväxlas med en blå kula. Labyrinten har även vissa röda delar som störde den röda färglokaliseringen. Dessa tester ledde till slutsatsen att grön var den mest fördelaktiga färgen att använda på kulan.

För att urskilja kulan från omgivningen används två tekniker. Den första är att undersöka vilka pixlar i bilden som har ett högt G-värde. Detta görs genom att hitta alla pixlar som har ett G-värde som ligger över ett visst tröskelvärde. Eftersom vit färg innehåller höga värden på både R, G och B komponenterna, krävs det ytterligare en teknik för att skilja de pixlar som är gröna mot vita. Detta löses genom att jämföra R och B värdet med G värdet. För att en pixel ska räknas som grön behövs både att G-värdet är tillräckligt högt och att R och B-värdena är tillräckligt låga jämfört med G-värdet. Denna metod valdes för att den inte var lika ljuskänslig som andra metoder som testats. Detta ansågs fördelaktigt eftersom systemet helst ska fungera var som helst, vilket leder till att systemet kommer utsättas för många olika ljusförhållanden.

De pixlar som i bilden räknas som gröna skapar en binär bild där gröna pixlar får värdet ett och resten noll, se Figur 7.2. Den binära bilden körs sedan i objekt `Vision.Blobanalysis`. Detta objekt undersöker den binära bilden och letar efter områden som har en hög koncentration av ettor. Det som detta objekt sen skickar ut är en så kallad Blob (binary large object) vilket är centrumkoordinaterna till de områden som innehåller mycket ettor. Detta objekt kan också justeras med hjälp av olika parametrar. Bland annat kan man ställa in ett intervall på storleken på Bloben man letar efter. Detta gjordes med ett intervall motsvarande kulans storlek för att minimera risken för att få fel objekt identifierat.

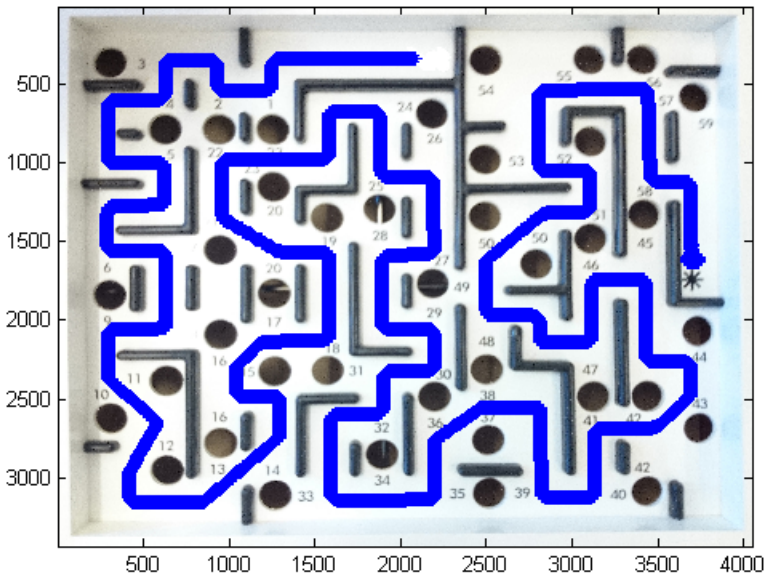


*Figur 7.2: Ett exempel på hur den binära bilden ser ut efter färganalys där man tydligt kan se kulan(den stora vita cirkeln). Man kan även se en liten blob som är startpunkten på labyrinten.*

## 7.3 Lokalisering av bana

För att lokalisera banan bedömdes att två möjliga alternativ fanns. Båda använder kameran för att identifiera en rutt. Alternativ ett identifierar väggar och hål på spelplanen. Detta är genomförbart då hål samt väggar är mörkare än övriga områden på spelplanen och det är därmed möjligt att identifiera dem på liknande sätt som kulan identifieras. Programmet skulle med alternativ ett räkna ut de områden som är ”säkra” för kulan att rulla på. Med säkra områden menas vägg- och hålfria områden och på så sätt ta sig genom labyrinten. Alternativ två vid val av lokalisering av bana var att ta ut koordinaterna för den svartmålade linje som utgör rutten. Detta alternativ bedömdes bäst på grund av en rad olika faktorer. Dels var alternativet lättare att programmera, då rutten är en sammanhängade linje med samma tjocklek. Alternativ ett kräver att man idetifierar väggar som är olika långa och positionerade på flera olika platser. Samt att hålens färg beror mycket på vilka ljusförhållanden som finns i rummet. Andra fördelar med alternativ två är att börvärdena man får är mer exakta. Detta då man får koorinater som ska följas istället för ett område som kulan skall hålla sig inom.

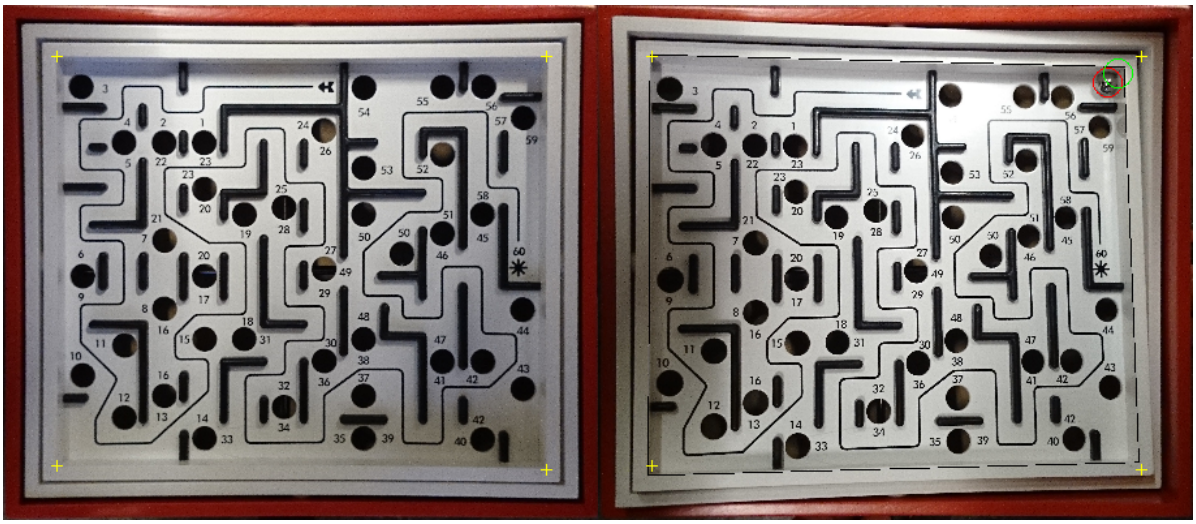
För att minska exekveringstiden under körningen hittas hela rutten innan start. Start och slutpunkterna hittas genom att de är målade i en annan färg än brädet och rutten. Även här fungerade det bäst med grön färg, då den skiljer sig mest ifrån det övriga spelets färger. För att inte blanda ihop den gröna startpunkten med den gröna kulan användes en mindre prick för startpunkten, samt olika parametrar för blobarea i elementet `Vision.Blobanalysis`. Detta gör det lätt att hålla isär koordinaterna för startpunkten och kulan. Därefter hittas rutten genom att avgöra i vilken riktning det är mörkast. Alltså beräknas det genomsnittliga RGB-värdet i en kvadrat med en sida ca 20 % mindre än linjens tjocklek i fyra vinkelräta riktningar. Den koordinat som är mörkast och som inte är i den riktning som förra börvärdet låg i läggs till som nästa börvärde i en lista med börvärden. Processen upprepas tills dess att slutet på banan nås. För att kunna urskilja slutet används två metoder, den första är ett gränsvärde på hur ljust området får vara. Det andra hindrar från att lägga till samma börvärde två gånger. Tillsammans hindrar detta processen från att dels hitta börvärden utanför linjen, det vill säga på det vita området, dels från att vända tillbaka och ”slingra” sig på slutet av linjen.



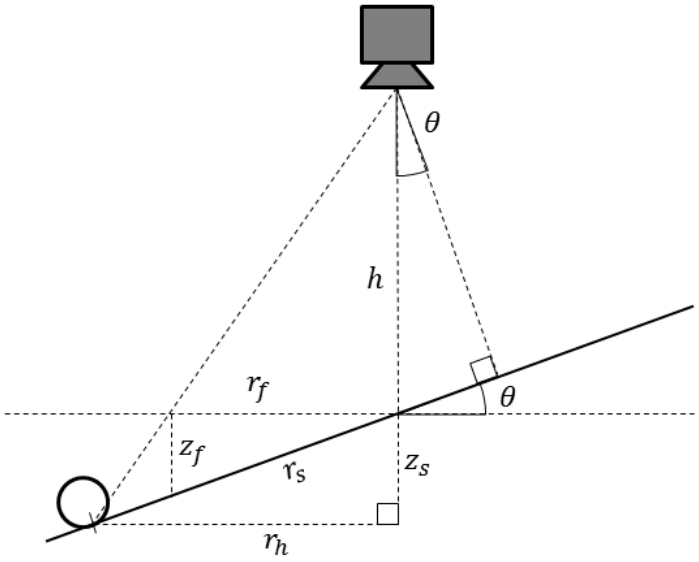

*Figur 7.3: Labyrintspel med bana identifierad*

## 7.4 Koordinattransformation

Eftersom kameran inte kan mäta avstånd och djup blir brädets koordinatsystem förvrängt när brädet vinklas. Kulans position enligt kameran blir då inte den position som kulan egentligen har vilket illustreras i Figur 7.4. Därför måste koordinater, som ges av kameran, justeras för att kunna användas som ärvärde. Fortsättningsvis kommer de koordinater som kameran ger kallas falska koordinater.



*Figur 7.4: Till vänster syns labyrinten i horisontellt läge med gula markörer som visar referenspunkter. Till höger, en bild som visar en lutande labyrint med referenspunkterna fortfarande synliga, tydlig förvrängning ses. I labyrinten till höger ses också två ringar. Den röda symboliserar vilken koordinat kameran tror att kulan har i referensplanet. Den gröna ringen symboliserar den koordinat som kulan egentligen har i referensplanet.*




*Figur 7.5: Figur över koordinatfelet. Enligt bilden från kameran befinner sig kulan längden $r_f$ från origo men i själva verket befinner den sig längden $r_s$ ifrån origo.*

Beräkning av de riktiga koordinaterna sker med definitioner enligt Figur 7.5. Först definieras ett koordinatsystem med origo i labyrintens mittpunkt. Det är här viktigt att kameran är placerad rakt över origo. Sedan beräknas den falska koordinatens z-värde, alltså utböjningen från xy-planet. Denna beräknas med Ekvation 7.1 där $x_f$, $y_f$ och $z_f$ är de falska koordinaterna och $\alpha_y$ och $\alpha_x$ är brädets vinklar kring vardera axel.

$$z_f = x_f \sin(\alpha_y) + y_f \sin(\alpha_x) \tag{7.1}$$

En vektor $r_f$ med de falska koordinaterna i $xy$-planet kan användas för att beräkna den totala vinkeln som brädet lutar jämfört med referensplanet tillsammans med den falska $z_f$-koordinaten enligt Ekvation 7.3.

$$r_f = [x_f \; y_f] \tag{7.2}$$

$$\theta = \arctan\left(\frac{z_f}{\parallel r_f \parallel}\right) \tag{7.3}$$

Från Figur 7.5 kan triangellikformighet nyttjas till Ekvation 7.4

$$\frac{r_f}{h} = \frac{r_h}{h + z_s} \tag{7.4}$$

där $r_h$, den horisontella förskjutningen, och $z_s$,den sanna utböjningen i $y$-led, kan beräknas med Ekvation 7.5 och 7.6:

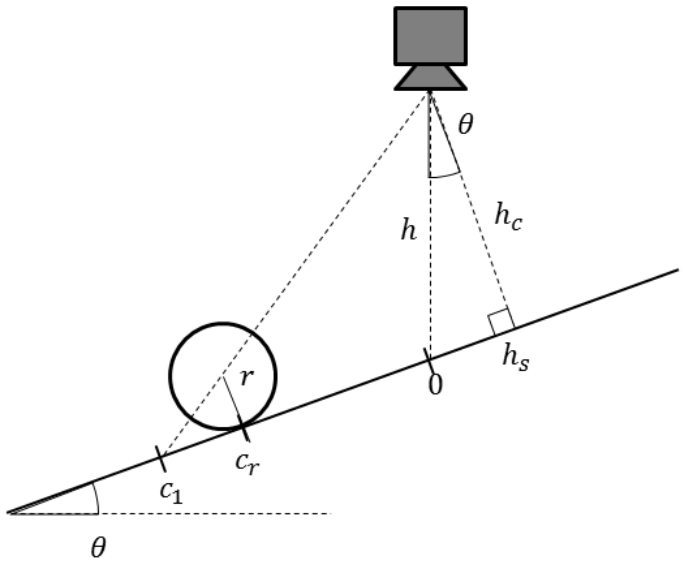
$$r_h = r_s \cos(\theta) \tag{7.5}$$

$$z_s = r_s \sin(\theta) \tag{7.6}$$

Dessa ekvationer kan användas för att härleda Ekvation 7.7 för att räkna ut den riktiga längden $r_s$.

$$r_s = \frac{r_f h}{h \cos(\theta) - r_f \sin(\theta)} \tag{7.7}$$

Hänsyn måste också tas till parallaxfelet som uppkommer av att koordinaten som kameran ger sitter i mitten på kulan, Se Figur 7.6.




*Figur 7.6: Figur över parallaxfelet. Figuren visar att längden $c_1$ som kameran ger är längre bort än den riktiga längden $c_r$.*

Den riktiga längden $c_r$ kan härledas från triangellikformighet i Ekvation 7.8 och ger efter förenkling Ekvation 7.9 för $c_r$.

$$\frac{c_1 + h_s}{h_c} = \frac{c_1 - c_r}{r} \tag{7.8}$$

$$c_r = c_1\left(1 - \frac{r}{h\cos(\theta)}\right) - r\tan(\theta) \tag{7.9}$$

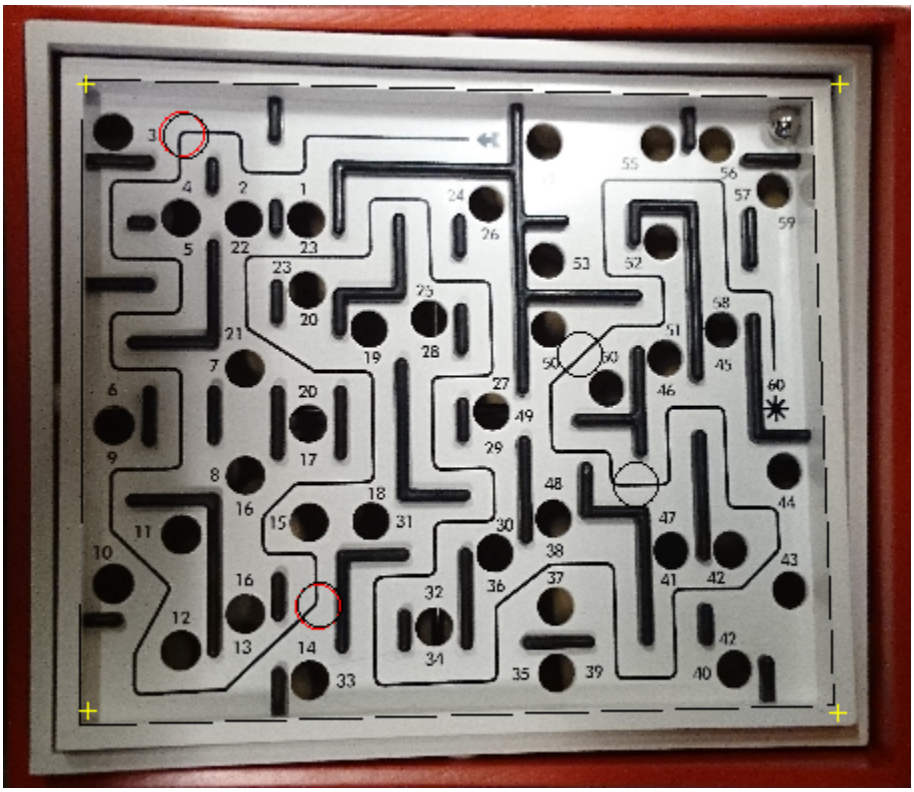
De riktiga koordinaterna kan nu beräknas genom att multiplicera den riktiga längden $c_r$ med enhetsvektorn för de falska koordinaterna enligt Ekvation 7.10

$$\begin{pmatrix} x \\ y \end{pmatrix} = c_r \frac{r_2}{\| r_2 \|} \tag{7.10}$$

Dessa koordinater är nu kulans riktiga ärvärde.

### 7.4.1 Analys av koordinattransformationer

För att utvärdera behovet av att implementera koordinattransformationen i det slutgiltiga systemet utfördes tester. Dessa utfördes genom att fotografera labyrinten rakt ovanifrån. På olika bilder lutades labyrinten åt olika håll för att testa så många utfall som möjligt.



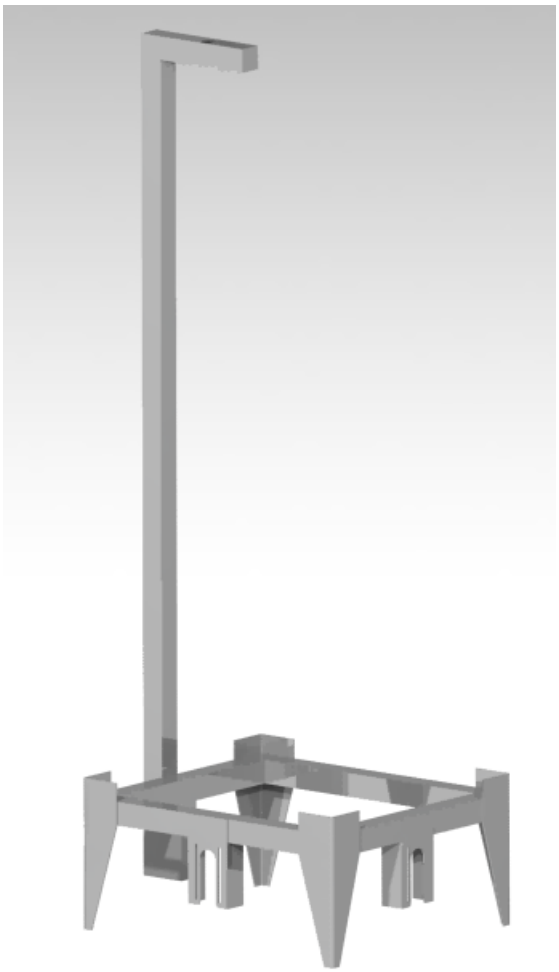
*Figur 7.7: Bild från test av koordinattransformation.*

Figur 7.7 visar hur testen utfördes. I MATLAB laddades två bilder in. En på labyrinten utan lutning och en bild på labyrinten i lutning. Genom att ta ut koordinaterna för labyrinthörnen i den olutade bilden kan man beräkna referensplanet. Detta syns i Figur 7.7 som gula kors. Sedan testades att ansätta fyra kulpositioner vilka syns i figuren som fyra svarta ringar. De röda ringarna som syns nära två av de svarta ringarna är det beräknade koordinatfelet för vardera koordinat. Anledningen till att det inte syns några röda ringar vid de andra två koordinaterna är att koordinatfelet var så litet. Det som gick att utläsa ur detta test var att koordinatfelet är väldigt litet. Detta test gjordes med kraftigare vinklar än vad prototypen använder så felet blev i verkligheten ännu mindre. Slutsatsen blev att implementering av dessa kordinatjusteringar inte var nödvändig.

# 8 Prototypen

För att kunna realisera prototypen behövs en konstruktion som binder ihop mjukvaran med hårdvaran. Detta består i att bygga en ram där kulspelet placeras och kamera och motorer är monterade. Eftersom prototypen skall vara presenterbar på mässor skall den även vara estetiskt tilltalande. Figur 8.1 visar den CAD-ritning som legat till grunden för prototypen. För detaljritningar se  Bilaga III: CAD-ritningar.

Det är kritiskt att den beräknade vinkeln på brädet uppnås samt att vinkeln på brädet kan regleras tillräckligt fort. Då systemet annars riskerar att bli instabilt. Detta ställer krav på både hastighet och precision hos motorerna. Prototypen slutade i ett system där en PC analyserar bilden samt gör alla beräkningar. PC:n kopplas sedan till ett drivsteg som skickar styrsignaler till motorerna.



*Figur 8.1: Översiktsbild på systemet*

## 8.1 Konstruktion av ram

Ramen i Figur 8.2 är tillverkad av fyra stycken fyrkantsprofiler med dimensionen 25x25mm. Profilerna kapades till längderna 330mm och 275mm. med 45 graders vinkel i varje ände. Fyrkantsprofilerna svetsades därefter samman med TIG(Tungsten Inert Gas)-svets.

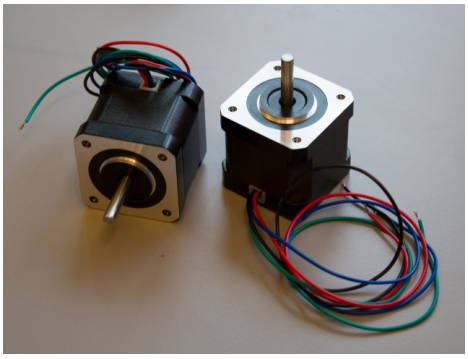
*Figur 8.2: Labyrintspelet monterat i ram*

Kameran monterades rakt ovanför centrum av brädet. Tester visade att en monteringshöjd på minst 70 cm är nödvändig för att kamerabilden skall fånga hela brädet. Till ben användes fyra stycken 1mm plåtar som bockats 90 grader och TIG-svetsats fast runt hörnen av ramen. Benen höjer upp ramen 10 cm. Utrymmet mellan mark och ram rymmer motorfästen samt motorer och remdrivning.

## 8.2 Motorval

Motorerna behöver vara både snabba och precisa, samtidigt är det viktigt att veta vilken vinkel motorerna står i. Stegmotorer passar in på dessa kriterier då de roterar ett antal önskade steg. Man vet då motorns vinkel och kan därmed driva dem i open-loop, dvs utan positionsåterkoppling.

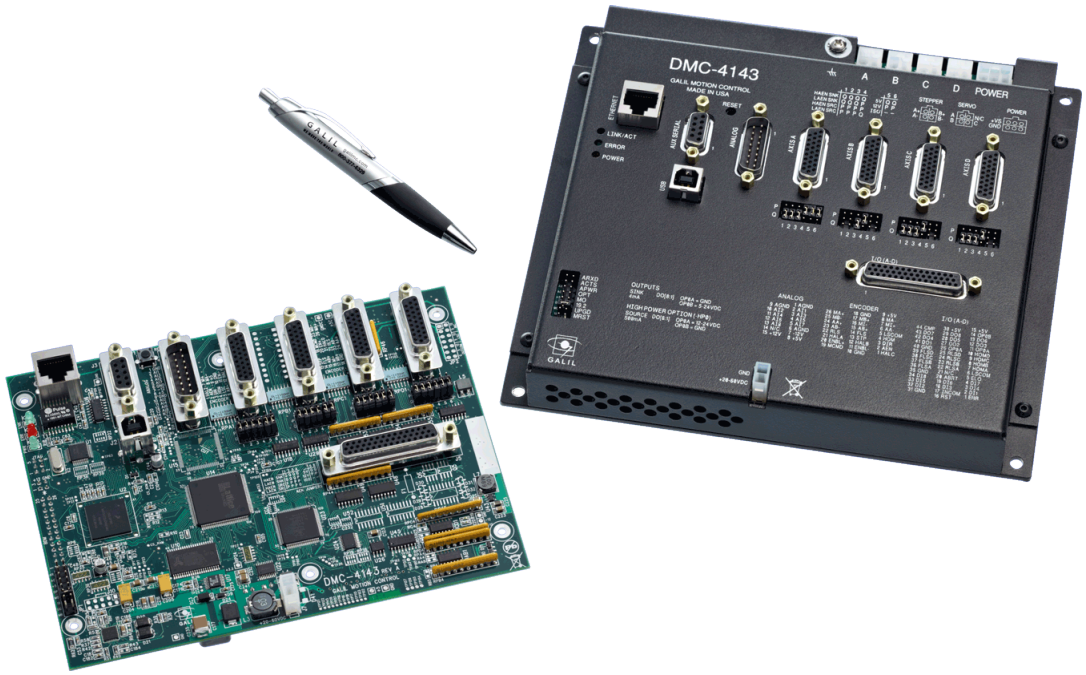
Företaget Compotech [4] sponsrade projektet med tre stycken 2-fasiga hybrid-stegmotorer från DING's motion [5]. Motorerna har en stegupplösning på 1,8 grader med en felmarginal på +/- 5%. De levererar ett maximalt moment på 0,48 Nm, vilket kan jämföras med kulspelets momentbehov som inte överstiger 0,015 Nm. Den låga lasten ledde till att motorerna var tillräckligt reaktionssnabba, och svängde in till önskad vinkel fort.



*Figur 8.3: Stegmotor 17H2039-120-4A*

## 8.3 Motorstyrning

Ett drivsteg från GALIL [3] används för att styra motorerna, även detta sponsrat från Compotech. Drivsteget kopplas ihop med en dator och kan styras från MATLAB. Drivsteget har möjlighet att microsteppa vilket medför en högre upplösning på stegen samt en mjukare motorgång. För att driva drivsteget krävs 20V och till detta användes en extern strömadapter.




*Figur 8.4: GALIL DMC-41x3*

## 8.4 Kraftöverföring

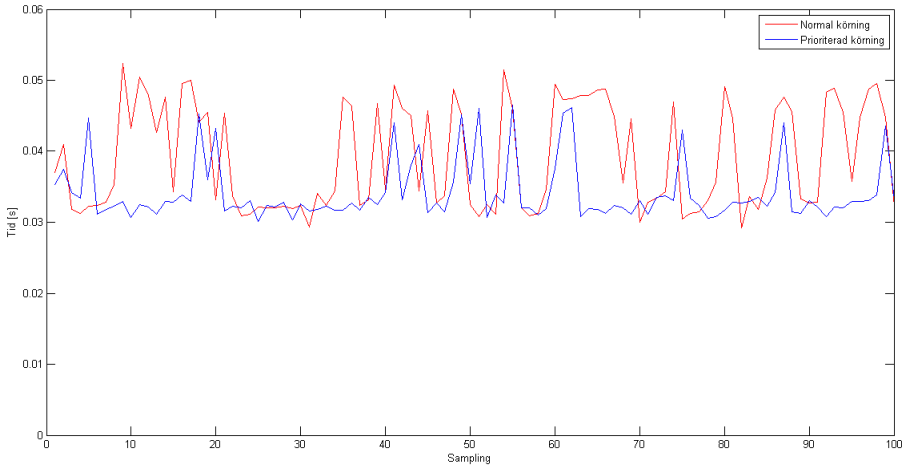
För att överföra kraften från motor till axlarna på kulspelet används remskivor och kuggremmar. Utväxlingen är en avvägning mellan stegupplösning och snabbhet; en låg utväxling ger en hög upplösning men ett långsammare system och vice versa. När utväxlingen bestämdes användes data från simuleringarna. Från tidiga simuleringarna kan man utläsa att brädet regleras med en upplösning på cirka 0,2 grader. Eftersom utväxlingen mellan brädet och reglagen har förhållandet 1:9 krävs alltså en upplösning på 1,8 grader på reglagen för att uppfylla simuleringens behov. Då motorerna har upplösningen 1,8 grader per steg kommer utväxlingen 1:1 mellan motor och reglage vara fullgod. Microsteppningen bidrar även med ytterligare högre upplösning.

## 8.5 Mjukvaruimplementering

Hela systemet exekveras i ett MATLAB script. Koden hämtar en bild från kameran där först banan identifieras och läggs in i en börvärdesvektor. Koden går sedan in i main-loopen, där bilder fortlöpande tas ut från kameran och kulans position beräknas. Efter detta beräknas ett börvärde och jämförs med positionen på kulan, felet används sedan i regulatorn som beräknar vinklarna som brädet skall ställa sig till. Koden skickar sedan dessa vinklar till drivsteget via USB. Sedan börjar nästa loop.

För att få en uppfattning om tiden ett varv i loopen tar har MATLABs egna tidtagningsfunktioner används. Samplingstiden som registreras för varje sample sparas i en vektor och därefter plottats över för ett antal samplingar. Sedan jämförs resultatet med tidigare data. Man kan se tydliga skillnader. Från de praktiska försöken har dessa parametrar visat sig ha mest negativ effekt på samplingstiden:

- Om plottning sker under tiden programmet körs
- Om MATLABs process inte är prioriterad i Windows




*Figur 8.5: Sampletid normal prioritet*

I Figur 8.5 beskriver y-axel tiden i sekunder och x-axel antal samples, alltså där tiden tas på nytt för nästa varv i while-loopen. Den röda kurvan är data tagen då Windows kör MATLAB i normal prioritet, medan den blå är med hög prioritet. En viss förbättring kan ses i Figur 8.5 då MATLAB körs med hög prioritet utan bakgrundsprocesser i gång. Skillnaden blir dock större ju fler program som körs i bakgrunden.

### 8.5.1 Regulatorstrukturen i MATLAB

Regulatorn i Ekvation 5.3 implementerades i MATLAB på följande sätt.

$$Ang_k = 13.02 * E_k - 12.67 * E_{k-1} + 0.444 * Ang_{k-1} \tag{8.1}$$

Där $Ang_k$ är vinkeln på brädet och $E$ är skillnaden mellan börvärde och position. Regulatorns filterdel motsvaras av $Ang_{k-1}$. Detta gjordes separat för x- och y axlarna.

## 8.6 Modifiering av labyrintspelet

För att minimera diverse glapp och instabiliteter i spelet har följande modifikationer på spelet genomförts:

- Byte av snöre

- Byte av fjädrar
- Placerat pexiglasskiva över banan för att kulan ska rulla på ett jämnare bräde för att kulan ska rulla på ett jämnare bräde

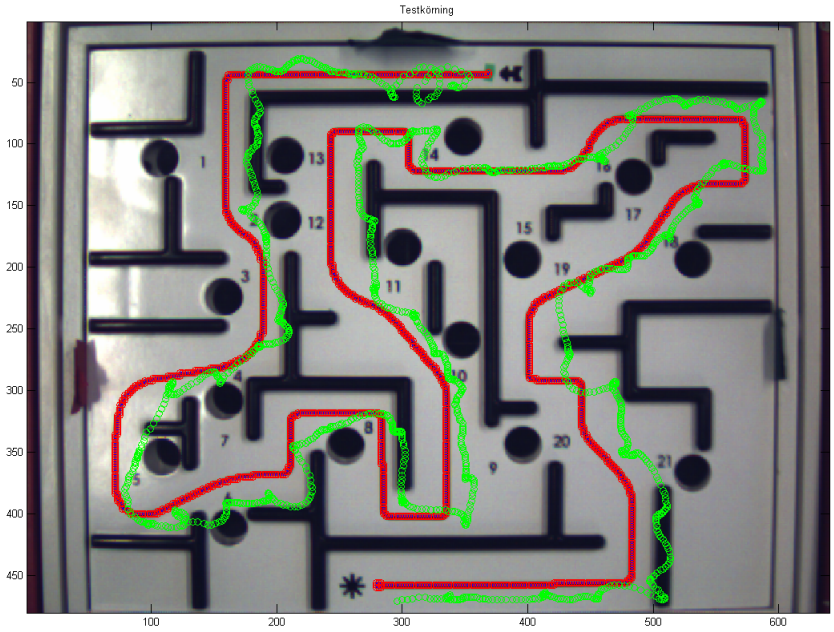
På de två axlarna som vrids löper två snören som virar upp sig. Då friktionen mellan axel och snöre är så låg finns risk att snöret "glider" på axeln. Risken finns då att motorerna "tappar" sin vinkel, och förlorar sin synkronisering. Därför har snörena byts ut mot tandtråd, som ger betydligt mer friktion mot axeln. Detta förhindrar glidning och säkerställer därmed att reglagets vinkel överensstämmer med brädets. För att minimera svängningar vid snabba vinkeländringar på brädet byttes originalfjädrarna ut mot kraftigare fjädrar.

På grund av stora ojämnheter i spelplanen har en plexiglasskiva lagts över planen. Med hjälp av skivan blir systemet av med en av störningarna som tidigare varit ett problem för att få stabilitet och repeterbarhet.

Kulan har lackerats grön för att möjliggöra lokalisering med kameran. Detta ledde dock till en yta som var ojämn och något klibbig vilket försvårade det reglertekniska ytterligare. Dels uppkommer det problem med vilofriktion, att kulan i det stillastående läget tenderar att fastna. Det krävs då en kraftig vinkel för att den skall börja rullla, något som leder till att kulan får en för kraftig accelaration när det väl sker. Ojämnheterna i kulan gör även att den oberäkneligt kan ändra riktning. Tyvärr har inga färgade matta kulor hittats, utan de som funnits att tillgå har varit transparenta, vilket gör de svåra att lokalisera då de ändrar färg beroende på det kringliggande området. Därför har inte kulans ojämnhet kunnat kringås.

## 8.7 Slutresultat

Den färdigställda prototypen fungerade bra mekaniskt redan från start. Mycket testkörning krävdes dock för att lyckas analysera systemets svagheter. I Figur 8.6 ses resultatet av en körning på en plexiglasskiva över den medelsvåra banan.



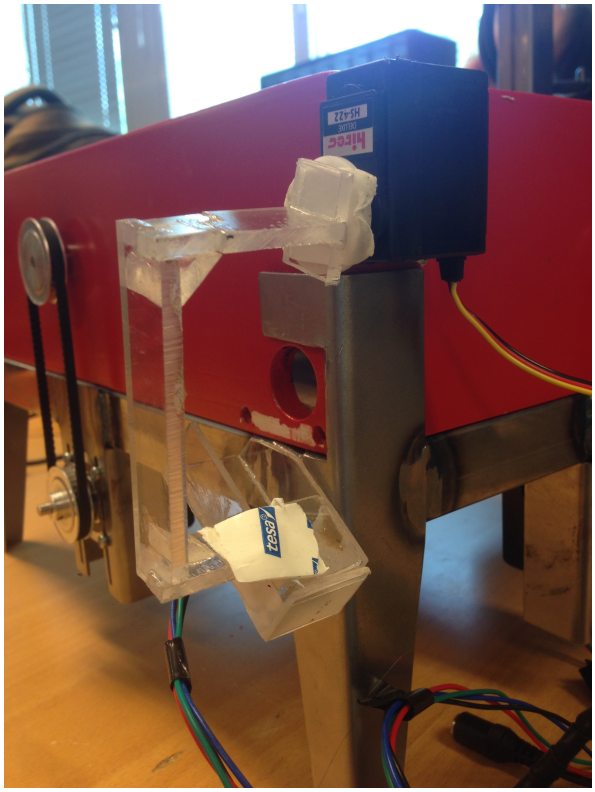

*Figur 8.6: Testkörning på plexiglasskiva. Det röda är börvärdena och det gröna är ärvärdena*

## 8.8 Uppfångstanordning

Företaget som har bistått med stegmotorer och drivsteg har uttryckt ett intresse för att använda labyrintspelet som utställningsobjekt på t.ex. en mässa. För att öka automatiseringsgraden har en matningsanordning för kulan tagits fram.

När kulan har nått sitt mål (eller trillar ner i ett annat hål) rullar den som vanligt på den lutande bottenplattan till hålet i sidan av spelet. Istället för skålen som ursprungligen satt under hålet fångas kulan nu upp av en behållare formad som ett rätblock med en öppning på ovansidan. Behållaren är monterad på en servomotor med en arm gjord av plexiglas.

När kulan trillar ner i behållaren sluts en brytare som är kopplad till ett Arduino-kort. Programmet på Arduinon gör då en loop. När programmet körs lyfter motorn upp behållaren som då färdas 155 grader medurs för att släppa av kulan i en stålränna monterad på labyrintspelets kant. Rännan tillåter kulan att färdas till sin startposition.




*Figur 8.7: Uppfångstarm*

# 9 Diskussion

Det finns många svårigheter med att automatisera ett kulspel. De flesta uppkommer ur störningarna systemet tillför. Störningar såsom; ojämnheterna i brädet, fjädersystemet mellan remhjul och bräde samt kulans ojämnheter. Dessa störningar bidrog till att vi inte lyckades reglera kulan inom marginalen för väggar och hål. Väggarna försämrar prestandan ytterligare då derivata-delen hos regulatorn registrerar ett stort utslag då kulan går emot. Under dessa förutsättningar börjar kulan ofta studsa mellan två motstående väggar med större och större vinkelutslag på brädet som följd. Detta fenomen gick till viss del att bli av med genom att filtrera d-delen i regulatorn, men vi lyckades inte bli av med fenomenet helt och hållet.

För att kringå problemet med det ojämna brädet användes en plexiglasskiva som yta att reglera kulan på. Detta fungerade bra även med en ojämn kula och fjädersystemet. Avsaknaden av väggar bidrog stort med att hålla nere störningarna. Test med fixt börvärde i mitten av plexiglasskivan visade på en väl fungerande regulator med snabb insvängningstid. Även när man körde banan ovanpå skivan fungerade systemet bra, med undantag då kulan slog i ytterväggarna. Systemet tenderar även att fungera sämre längre ut på brädets kanter, något som antagligen beror på att fjädrarna töjs ut en aning då vikten av kulan hamnar längst ut på brädet, vilket gör att vinkeln på brädet inte är helt synkroniserat med motorernas. Vinkeln gör även kulan mer benägen att rulla in i ytterväggarna.

# 10 Slutsats

Tidigt i projektet, då projektgruppen bestämde det önskade resultatet av arbetet gjordes vissa optimistiska antaganden. Detta har i efterhand gjort att vissa av målen förändrats för att fungera under de verkliga förhållandena. Detta nämns i kapitlet 8.6 Modifiering av labyrintspelet. Kapitel 1.2 Syfte beskriver syftet med projektet och vilka olika funktioner systemet skulle klara av att genomföra. Flertalet av funktionerna uppställda klarar systemet av att genomföra helt och hållet, andra delvis. De funktioner som systemet uppfyller är

- Generellt styrsystem implementerat
- Ökar intresset för tekniken
- Helautomatiskt

En av de funktioner som gruppen bestämde var att styrsystemet skulle vara generellt för de olika banor som medföljer BRIO spelet. Denna funktion uppfylls då kodningen klarar av att ta ut både bana och börvärden oavsätt bana. En annan slutsats av projektet är att det ökar intresset för tekniken bakom systemet. Detta har märkts bland annat genom att flera andra studenter har stannat till och titta lite extra eller frågat vad det är för något vi har gjort och hur det fungerar.

De funktioner som inte uppfylls är:

- Fungerande prototyp för demonstration
- Repeterbarhet

Prototypen som tagits fram är i nuläget inte helt fungerande, kulan klarar inte att köra hela banan från start till mål utan att falla ner i hål eller ”fastna” vid någon utav väggarna i labyrinten. Därmed är systemet inte heller repeterbart.

## 10.1 Lärdomar

Under projektets gång har många viktiga erfarenheter förvärvats. Den viktigaste erfarenheten, som är starkt kopplat till resultatet av projektet, är den inverkan störningar har. När gruppen först fick BRIO spelet observerades stora ojämnheter på brädet samt att brädet regleras med hjälp av fjädrar på undersidan, som nämns i kapitel 4.1 Brädets dynamik. Detta ansågs kunna regleras ”bort” med hjälp av bra regulatorparametrar, detta har inte lyckats till fullo. Det visade sig vara svårare att modellera systemet korrekt än vad vi först trott. Kulans inverkan på resultatet var en annan lärdom som gruppen förvärvat. Under programmeringsstadiet så bestämdes att kulan skulle behövas färgläggas för att kameran skulle kunna identifiera den. Detta visade sig vara svårare att genomföra än förväntat. Problemen var bland annat att få kulan rund nog samtidigt som den är tillräckligt färglagd. Ett område som gruppen har lärt sig mer om är hur man kan använda en kamera som sensor och programmeringen kring detta. Andra områden som gruppens medlemmar har lärt sig mer om är simulering i Simulink, som varit viktigt igenom hela projektet.

Den samlade bedömningen av kandidatarbetet är att det inte uppnått vissa av de mål som satts upp. Några av målen med projektet har nåtts då kunskapen om simulering, reglerteknik och MATLAB programmering har ökat.

## 10.2 Rekommendationer till fortsatt arbete

För att få kulspelet att uppfylla de resultat som ställts behöver störningarna minimeras. Man bör bland annat undersöka alternativ som löser problemet med planets ojämnheter samt hitta en kula med rätt färg och rundhet. Med dessa störningar åtgärdade bör problem med skillnad mellan simulering och verklighet bli mindre. För att minska störningar från spelplanen rekommenderas att tillverka en egen plan, identisk med den nuvarande men med planare yta. Man bör även undersöka noggrant huruvida det går att byta ut fjädrarna mot snören, alternativt undersöka lösningar med kulskruvar eller linjäraktuatorer.

# 11 Hållbar utveckling

Ett av målen som ställdes upp för projektet var att ta fram ett system som kan användas som utställningsobjekt på t.ex. mässor för att visa vad som kan åstadkommas med hjälp av reglerteknik.

Produkten i sig har inte tagits fram för att vara så miljövänlig som möjligt. Däremot kommer ett ökat intresse för reglerteknik i förlängningen förhoppningsvis generera ett ökat intresse för teknik och ingenjörsyrket överlag. Det är något som ger goda förutsättningar till framsteg inom hållbar utveckling.

För att kunna fortsätta arbetet med hållbar utveckling är det viktigt att tillgodose framtidens kunskapsbehov. De senaste decenniernas utveckling inom teknik är något som indikerar att behovet av duktiga och drivna ingenjörer knappast kommer att minska. Ingenjörer behövs för att lösa problem som förbättrar miljön och levnadsstandarden för oss människor.

# 12 Referenser


[1] Boström Magnus, Sundin Niklas (2001). *Total Design and Development of a Labyrinth Demonstration System* Göteborg, Chalmers Teniska Högskola

[2] Fredriksson Magnus, Lagergren Niklas (1997). *Konstruktion av labyrint med vision-baserat reglersystem* Göteborg, Dept. of Mechatronics - Chalmers Teniska Högskola

[3] *GALIL MOTION CONTROL* http://www.galilmc.com/products/dmc-41x3.php, hämtad 2014-05-16

[4] *COMPOTECH* http://www.compotech.se/, 2014-05-15

[5] *DINGS MOTION* http://www.dingsmotion.com/en/productshow.asp?id=703, 2014-05-15

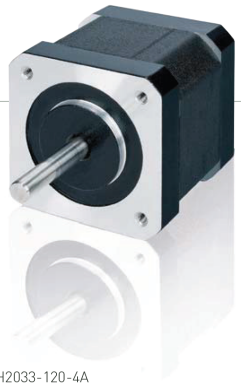
[6] *PSU, EXTERNAL, 24V* http://se.farnell.com/artesyn-embedded-technologies/ad8024n3l-001/psu-external-24v-3-25a/dp/1643236, hämtad 2014-05-16

[7] *EXEMPELHEMSIDA* http://www.hemsida.com, hämtad 2014-02-27

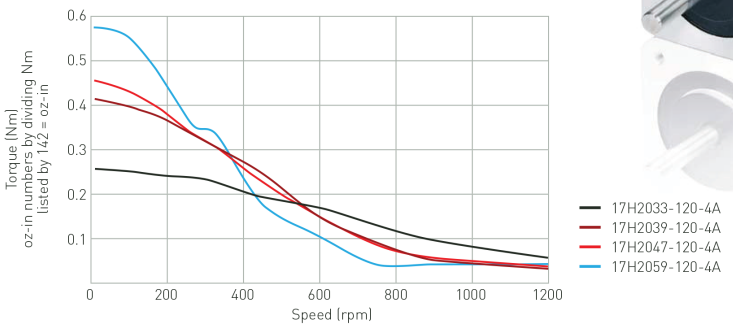
# A Bilaga I: Datablad för stegmotor



## SIZE 17H2 (42 mm) · 2 phase 1.8°
## Hybrid Stepper Motor



**Pull out torque-speed curves** 24 V DC Chopper driver, 2 phases

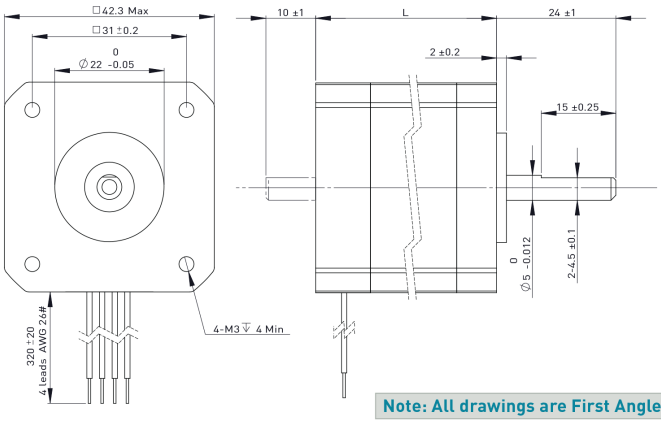



**Specifications** Please consult your authorized sales representative for custom specifications.

| Model | Holding torque Nm | Current A/phase | Resi-stance (Ω) | Step angle | Rotor inertia gfcm² | Weight kg | Motor Length L (mm) | inductance mH |
|---|---|---|---|---|---|---|---|---|
| 17H2033-120-4A | 0.32 | 1.2 | 2.8 | 1.8° | 35 | 0.22 | 33 | 4.2 |
| 17H2039-120-4A | 0.48 | 1.2 | 3.2 | 1.8° | 54 | 0.28 | 39 | 3.6 |
| 17H2047-120-4A | 0.5 | 1.2 | 3.5 | 1.8° | 77 | 0.35 | 47 | 6.0 |
| 17H2059-120-4A | 0.75 | 1.2 | 6.0 | 1.8° | 114 | 0.5 | 59 | 16.0 |

16

**Dimension (mm) · Size 17H2 (42 mm) · 2 phase 1.8°:**




**Note: All drawings are First Angle Projection – ISO Standard**



*Figur A.1: Datablad för stegmotorerna*

# B Bilaga II: Datablad för strömförsörjning

Embedded Power for
Business-Critical Continuity

Rev. 01.31.08
AD8024N3L-001
1 of 2

## AD8024N3L-001

78 Watts

**Total Power:** 78 Watts
**Input Voltage:** 90 - 264 Vac
**# of Outputs:** Single

RoHS 2002/95/EC

ENERGY STAR

## Special Features

- CE Mark EMC & LVD
- Universal AC input
- Fully regulated output
- Overcurrent, Overvoltage and Thermal protection
- Constant voltage
- High efficiency
- High MTBF
- IEC320 AC receptacle, 3 pin (type C14)
- Built in EMI filter
- (CISPR 22 Class B)
- AC Input fuse
- Complies with One Watt Input Energy Star / Blue Angel Requirement

## Safety

**UL:** UL 60950-1
**CSA:** CSA-C22.2 no.60950-1
**NEMKO:** EN/IEC60950-1
**CB:** Certificate and report

## Electrical Specifications

| Input | |
|---|---|
| Input range: | 90-264 Vac (wide range) |
| Frequency: | 47-63 Hz |
| Input current: | 2 A maximum |
| Efficiency: | 84% typical |
| EMI/RFI: | FCC Part 15, Subpart B Class B & EN55022 (CISPR 22) Class B |
| Safety ground leakage current: | 0.5 mA maximum@ 50/60 Hz, 264 Vac input |
| **Output** | |
| Maximum Power (Po): | 78 W |
| Hold-up time: | 10 ms. minimum at full load @ 115 Vac |
| Overcurrent protection: | Output short circuit protection auto recover<br>Overload protection @ 110 - 120% above maximum rating |
| Thermal protection: | Internally protected; output will latch off |
| Cable/connector: | DC cable length 2.5 mm center plug<br>DC plug center +v<br>DC plug outer -v |

## Environmental Specifications

| | |
|---|---|
| Operating temperature: | 0° to +40°C ambient |
| Storage temperature: | -40°C to +70°C |
| Electromagnetic susceptibility: | Designed to meet EN61000-4-2, -3, -4, level 2; EN61000-4-5, Level 4; EN61000-3-3 |
| Humidity: | Operating; non-condensing 5% to 90% RH |
| MTBF demonstrated: | >300,000 hours at full load and 25°C ambient conditions |

ASTEC

EMERSON Network Power

*Figur B.1: Datablad för strömförsörjning*

Embedded Power for
Business-Critical Continuity

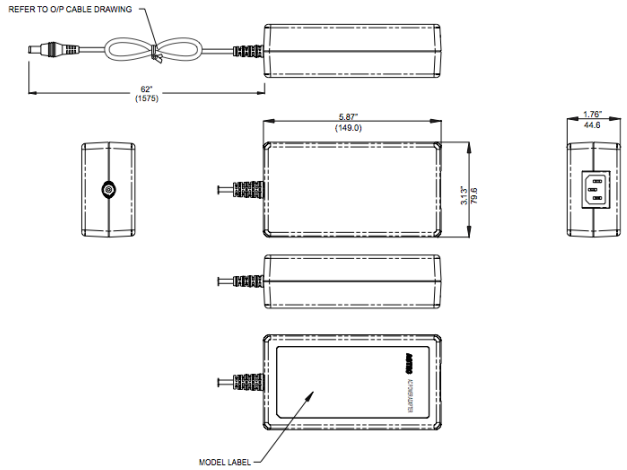


## Ordering Information

| Model Number | Maximum Power | Output Voltage | Maximum Load | Peak Load[1] | Regulation[2] | Ripple P/P (PARD)[3] |
|---|---|---|---|---|---|---|
| AD8024N3L-001 | 80 W | 24 Vdc | 3.25 A | 4.16 A | ±5% | <400mV |

1. Peak current lasting <4 seconds with a maximum 10% duty cycle.
2. At 25ºC including initial tolerance, line voltage, load currents and output voltages adjusted to factory settings.
3. Peak-to-peak with 20MHz bandwidth and 10µF (tantalum capacitor) in parallel with a 0.1µF capacitor at rated line voltage and load ranges.

## Mechanical Drawing




1.76" (H) x 3.13" (W) x 5.87" (L)
44.0 mm (H) x 79.6 mm (W) x 149.0 mm (L)
AC Input Connector: IEC320, C14
AC Input power cord sold separately

**Notes:**

1. Specifications subject to change without notice.
2. All dimensions in inches (mm), tolerance is ± 0.02" (±0.5 mm)
3. Warranty: 2 year
4. Weight: 0.91 lb./ 0.41 kg
5. AC input power cord sold separately.
6. Specifications at factory settings at 115VAC input, 25 °C unless otherwise stated

**Americas**
5810 Van Allen Way
Carlsbad, CA 92008
USA
Telephone: +1 (760) 930 4600
Facsimile: +1 (760) 930 0698

**Europe (UK)**
Waterfront Business Park
Merry Hill, Dudley
West Midlands, DY5 1LX
United Kingdom
Telephone: +44 (0) 1384 842 211
Facsimile: +44 (0) 1384 843 355

**Asia (HK)**
14/F, Lu Plaza
2 Wing Yip Street
Kwun Tong, Kowloon
Hong Kong
Telephone: +852 2176 3333
Facsimile: +852 2176 3888

For global contact, visit:
**www.powerconversion.com**
**technicalsupport@powerconversion.com**



**Emerson Network Power.**
The global leader in enabling business-critical continuity.

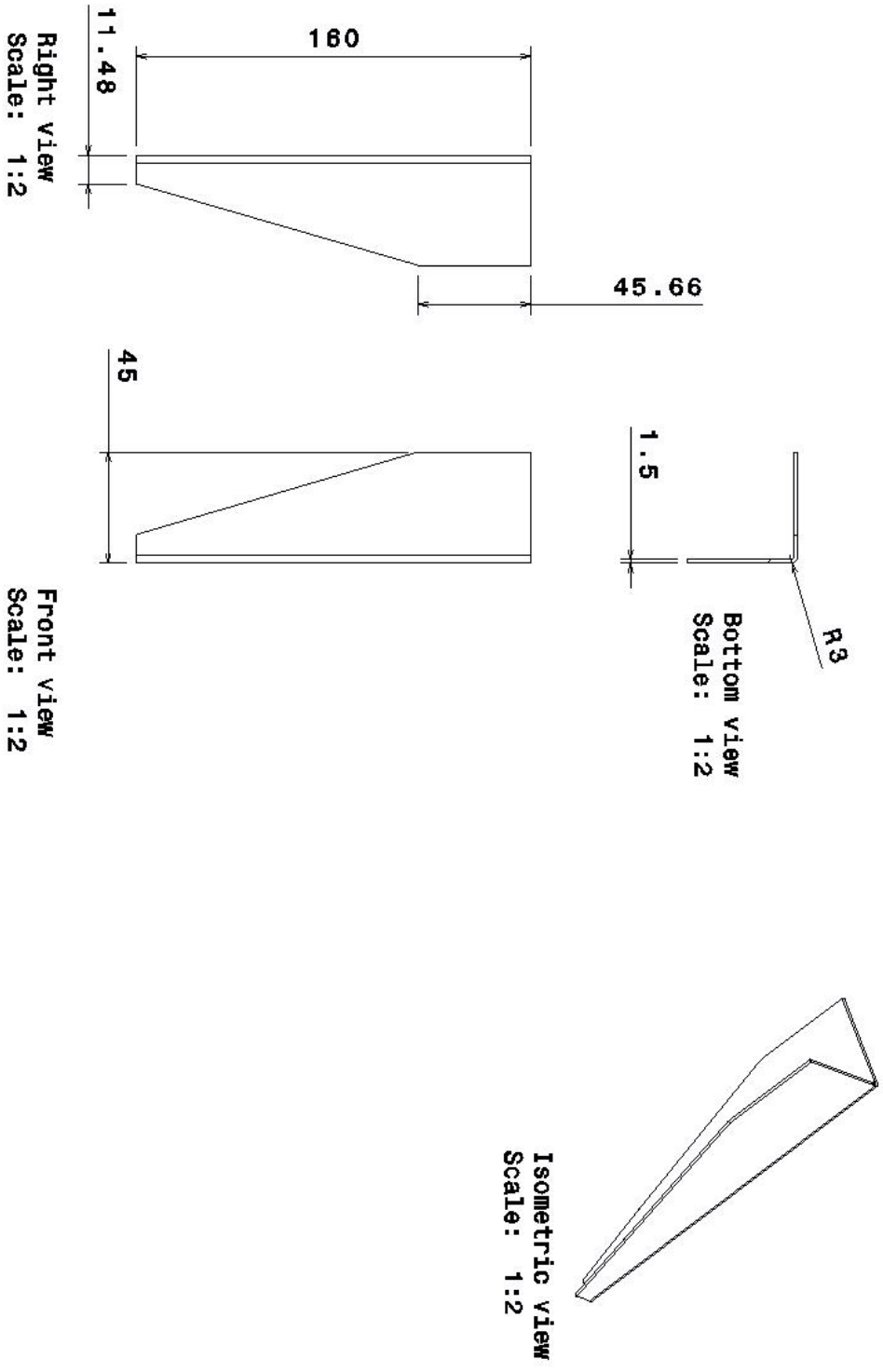
- AC Power
- Connectivity
- DC Power
- Embedded Computing
- **Embedded Power**
- Monitoring
- Outside Plant
- Power Switching & Controls
- Precision Cooling
- Racks & Integrated Cabinets
- Services
- Surge Protection

**EmersonNetworkPower.com**



*Figur B.2: Datablad för strömförsörjning*

# C Bilaga III: CAD-ritningar

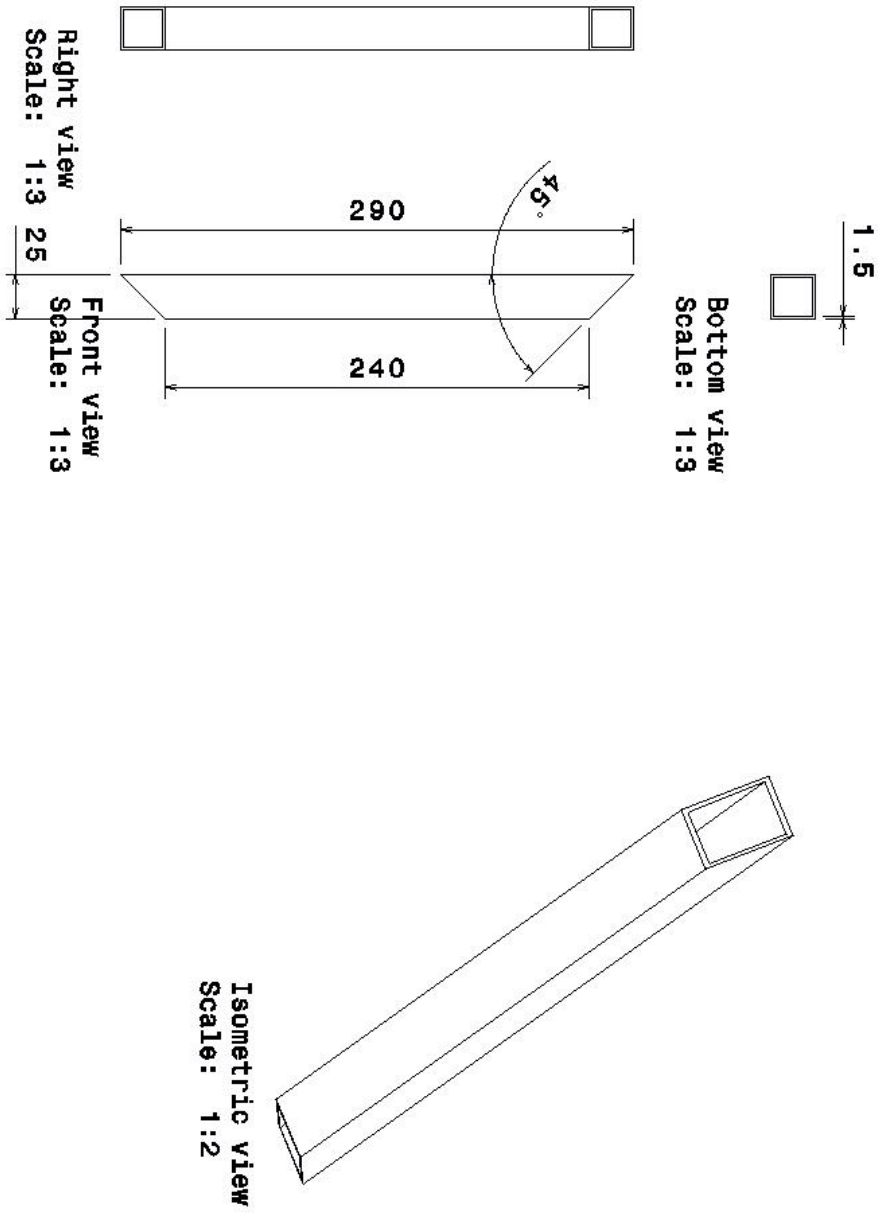



*Figur C.1: Fötter till ram*

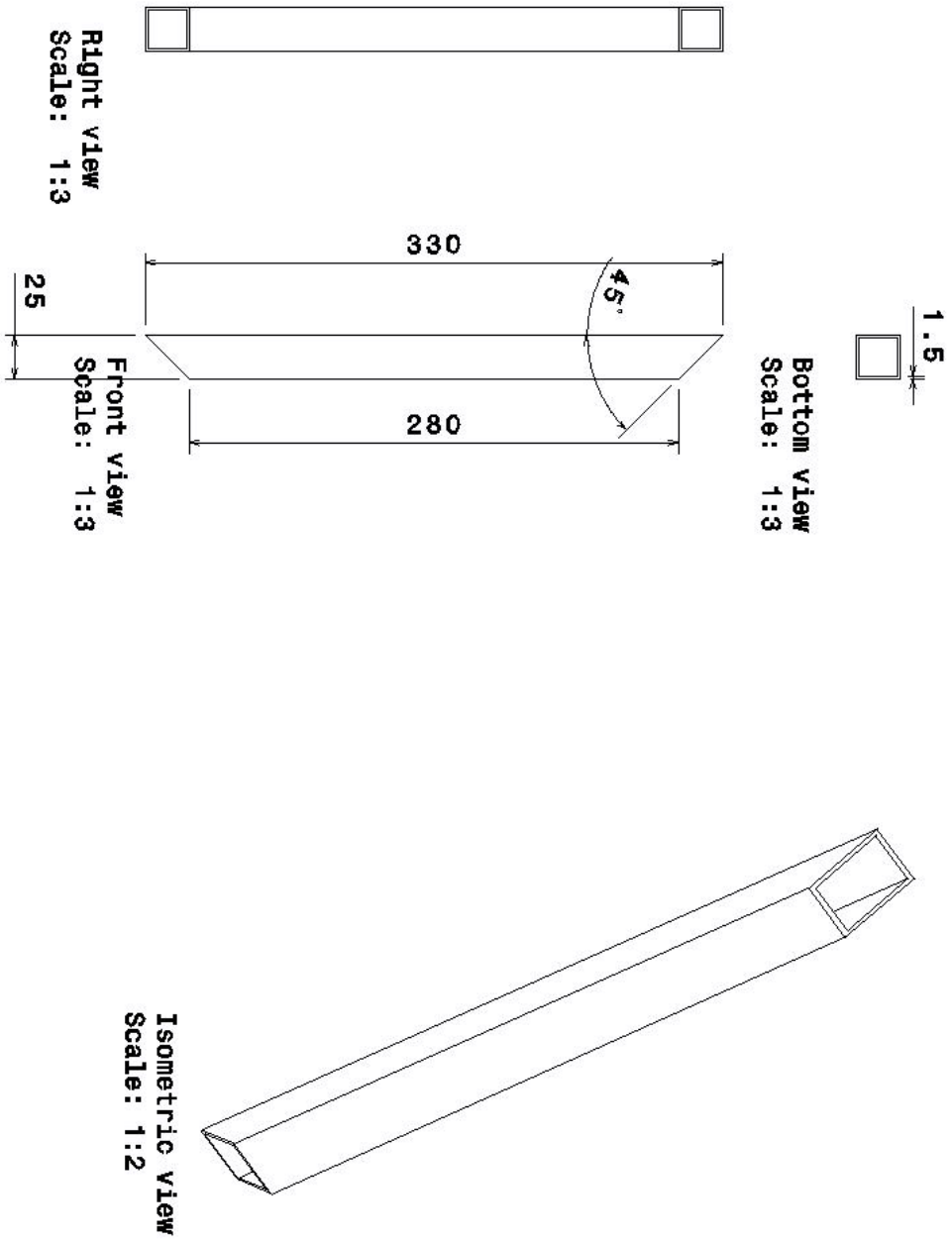
*Figur C.2: Kamerastativ övre del*

*Figur C.3: Kamerastativ nedre del*

*Figur C.4: Stöd stativ*




*Figur C.5: Kortsida ram*




*Figur C.6: Långsida ram*