Beamesteem

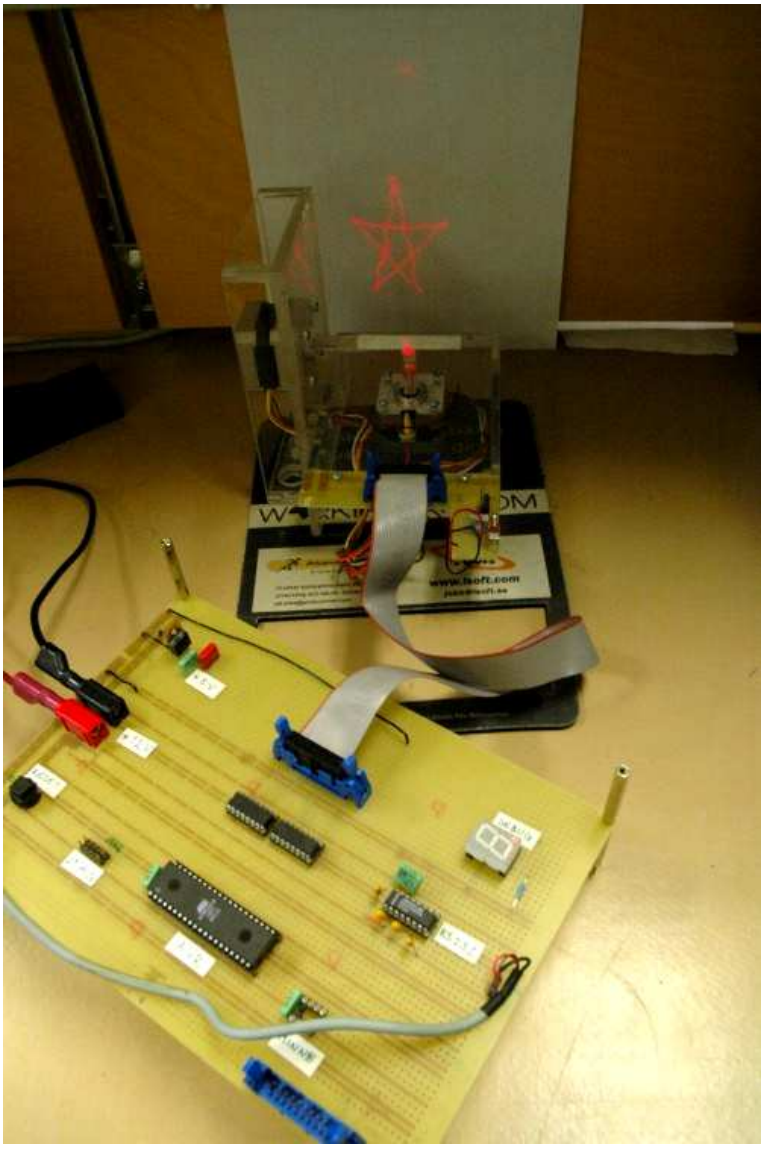
ett projekt av

Aron Kornhall
Magnus Starseth
Martin Jerling

Handledare
Bertil Lindvall

i

Digitala projekt Stor kurs 2005



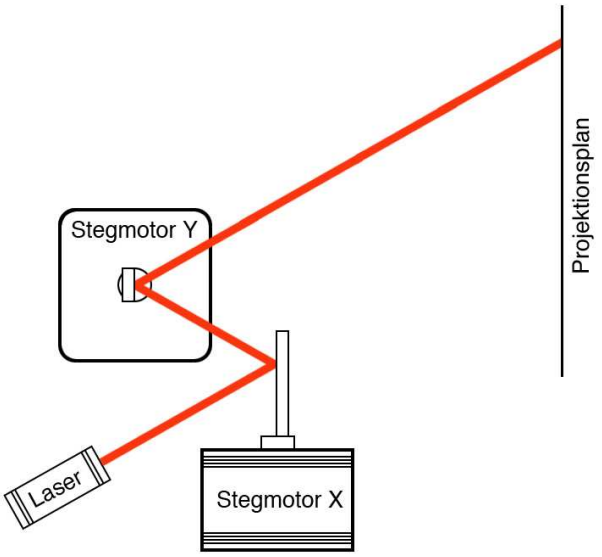
Innehållsförteckning

Idé

Ögat har en viss tröghet. Detta är vida omvittnat av alla som någongång viftat med ett brinnande tomtebloss. Istället för en punkt som rör sig ser vi något som kan liknas vid en linje. Idén bakom beamesteem är att utnyttja denna ögats tröghet för att med en laserpunkt rita figurer på en yta så snabbt att ögat uppfattar det som en bild istället för en kringflackande laserpunkt.

Konstruktion

Konstruktionen bigger på en laser och två st spegelförsedda stegmotorer. Laserstrålen reflekteras först mot den ena spegeln vars vinkel bestämmer laserpunktens x-koordinat och sedan mot den andra vars vinkel bestämmer y-koordinaten. Lasern kan även stängas av och sättas på av mjukvaran, så det är möjligt att rita icke sammanhängande figurer. Lasern och stegmotorerna styrs av en AVR-processor via effekttransistorer och matas med högsta ”tillåtna” spänning (12 V) för att kunna klara av de snabba accelerationer och retarationer som krävs.



Konstruktionen ansluts till serieporten på en PC där styrprogrammet körs. I styrprogrammet ritar användaren upp den figur som skall projiceras för att sedan skicka den till AVR-processorn. Styrprogrammet används också för att rikta int speglarna eftersom stegmotorerna inte är återkopplade och därmed inte heller kan riktas in automatiskt.

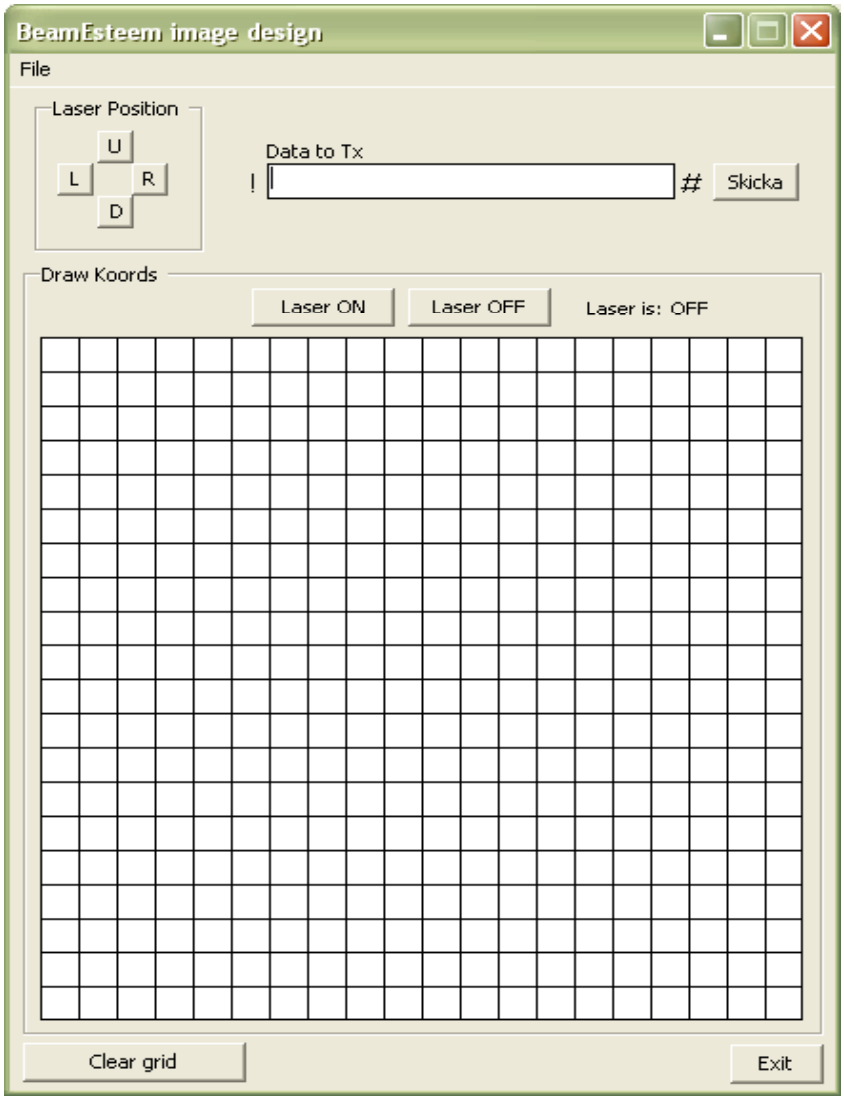
Mjukvaran

Mjukvaran i AVR-processorn utgör ett gränssnitt mellan stegmotorerna och styrprogrammet i PCn och omvandlar bilden den får från detta i form av en serie koordinater samt information om när lasern skall tändas respektive släckas till lämpliga utsignaler till stegmotorer och laser.

Omvandlingen sker i två lager: I övre lagret används en variant av Bresenhams algoritm för att beräkna kortaste vägen till nästa koordinat i form av en serie steg för de båda stegmotorerna. Varje sådant steg behandlas sedan i det andra lagret som håller reda på stegmotorernas aktuella positioner och därmed kan beräkna vilken binär fyrtippel som skall skickas till vilken stegmotor.

PC-Mjukvaran

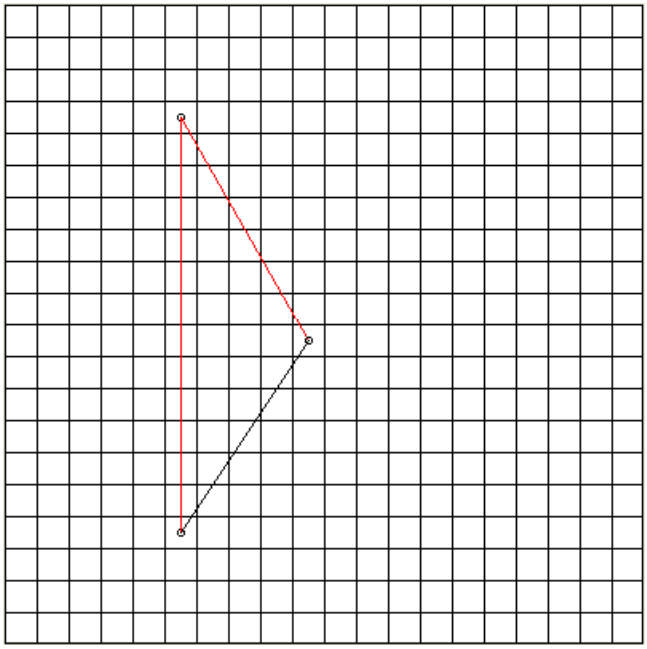
När mjukvaran för PC:n startas visas följande fönster.



När programmet har startats så kopplar det upp sig till hårdvaran via serieporten. De fyra knapparna längst upp i fönstret, **U**, **L**, **R** och **D** används för att positionera laserpunkten till sitt utgångsläge.

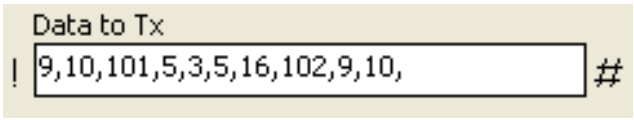
Nu kan man även börja att sätta ut punkter. Det gör man genom att klicka med musknappen i rutnätet. Ovanför rutnätet finns två knappar och en statustext som visar om lasern är på eller av för den linje som kommer att ritas ut. Genom att trycka på knapparna byter man läge för lasern, vilket då kommer att indikeras av statustexten. En linje som ritas med lasern på kommer att ha röd färg medan en linje där lasern är av kommer att bli svart.

Här är ett exempel på hur det kan se ut.



Den här bilden visar två linjer som ritas med lasern igång och en linje där lasern är av.

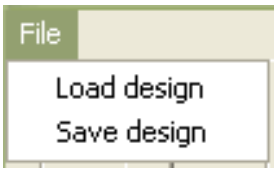
Samtidigt som man bygger upp en bild i rutnätet kan man se den data som kommer att skickas till hårdvaran när bilden är kvar. Den informationen visas i textrutan överst i fönstret.



Den data som visas här motsvarar bilden i rutnätet här ovanför. Här syns att vi har börjat med att placera ut en punkt i **(9,10)**, därefter har vi satt igång lasern **(101)** och satt nästa punkt till **(5,3)**, sedan till **(5,16)** och sedan har vi stängt av lasern **(102)** och förflyttat oss tillbaka till punkten **(9.10)**. Första och sista tecknet som kommer att skickas till hårdvaran är de tecken som står på vardera sidan om textrutan, ”**!**” är ett starttecken och ”**#**” är ett sluttecken.

När bilden är klar kan den skickas över till hårdvaran genom att trycka på knappen ”Skicka” som finns precis till höger om textrutan med koordinatinformationen.

Nu kommer en dataström att skickas över till hårdvaran via serieporten. Under tiden detta sker kommer hårdvaran att blinka med lasern för att visa att en överföring är aktiv.
En design kan även sparas till och laddas in från disk. Detta görs från **File**-menyn



Kommunikation med hårdvaran

All kommunikation mellan mjukvaran och hårdvaran sker via serieporten. Den inbyggda UART:en i AVR-processorn är konfigurerad för att köra med 4800 baud, 8 bitar, ingen paritet och en stoppbit. UART:en är väldigt enkelt uppbyggd och är klar att användas med bara några få C-instruktioner. Alla tecken som skickas till hårdvaran över serieporten skickas även tillbaka så att mjukvaran på PC:n kan verifiera att rätt tecken har kommit fram.

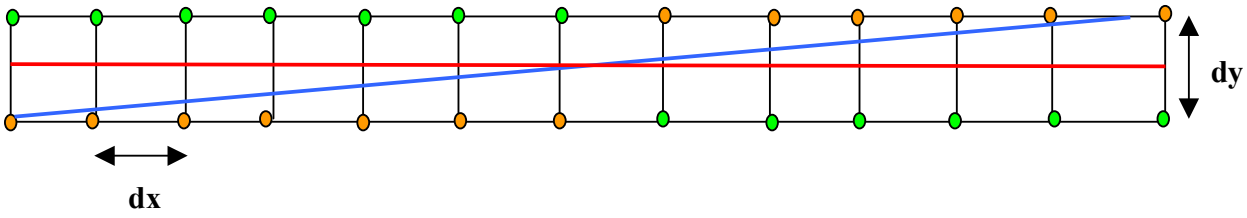
Serieporten som används i mjukvaran på PC:n är hårdkodad till COM2, men det är lätt att själv ändra detta i källkoden.

Linjedragning med Bresenhams algoritm

En linje är definerad som den kortaste sträckan mellan två punkter. I verkligheten består denna linje av oändligt många punkter. Dock när vi använder vårt grafiska gränssnitt samt våra stegmotorer har vi bara möjlighet att representera ett ändligt antal lägen mellan dessa två punkter.

Det är enkelt att konstruera en algoritm som beräknar vilka punkter som vi skall färga , dock blir det nödvändigt att använda flyttalrepresentation. Genom att använda Bresenhams algoritm kan vi undvika detta och samtidigt använda en heltalsrepresentation och endast använda addition i vår huvudloop.

Om vi vill dra en linje mellan två punkter (x1,y1) och (x2,y2) kan vi välja att färglägga pixlar beroende på vilken pixel som ligger närmast till linjen.



I figuren kan vi se att mellan varje pixel finns ett avstånd (dx,dy) som vi inte kan färglägga. Genom att kontrollera vilken pixel vi befinner oss närmast i y-led, d.v.s. om vi befinner oss ovanför eller under dy/2 (Den röda linjen i figuren), kan vi avgöra om vi ska färglägga pixeln ovanför till höger eller nedanför till höger.

Genom att utgå från linjens ekvation kan vi härleda en algoritm för att få reda på om vi befinner oss ovanför mitten linjen.

Linjens ekvation ges utav:

$$y = \frac{dy}{dx}x + B$$

Genom följande omskrivningar kan vi se att för en punkt ovanför mitten blir ett negativt nummer och ovanför ett positivt.

$$dx * y = dy * x + B * dx$$
$$0 = dy * x - dx * y + B * dx$$

Detta ger oss följande ekvation.

$$F(x, y) = 2 * dy * x - 2 * dx * y + 2 * B * dx \qquad \textbf{1}$$

Genom att multiplicera med två kommer vi att slippa en division senare.

Genom att i den första punkten i figuren (x1+1,y1+1/2) använda vår ekvation **1** kan vi avgöra om punkt (x1+1,y1) eller (x1+1,y1+1) skall färgas.

Detta ger oss

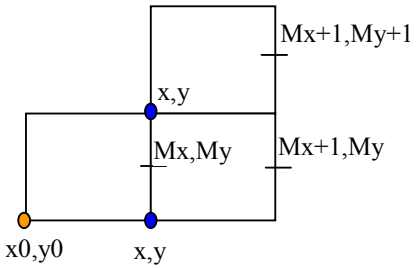
$$F(x_1 + 1, y_1 + 1/2) = 2 * dy * (x_1 + 1) - 2 * dy * (y_1 + 1/2) + 2 * B * dx$$
$$F(x_1 + 1, y_1 + 1/2) = 2 * dy * x1 + 2 * dy - 2 * dx * y1 - dx + 2 * B * dx \quad \textbf{2}$$

Genom att vi multiplicerat vår ekvation med 2 slipper vi flyttal i ekvation **2** vilket ger oss bättre och snabbare uträkningar.

Om vi nu subtraherar F(x1+1,y1+1/2) från F(x1,y1) får vi

$$F(x_1, y_1) = 2 * dy * x_1 - 2 * dx * y_1 + 2 * B * dx = 0$$
$$F(x_1 + 1, y_1 + 1/2) = 2 * dy * x_1 + 2 * dy - 2 * dx * y_1 - dx + 2 * B * dx$$
$$d0 = F(x_1 + 1, y_1 + 1/2) = 2 * dy - dx$$

d0 kallas för beslutspunkten . Den ger oss vilken som är vår nuvarande mittenpunkt.
Genom att utnyttja linjens egenskaper kan vi beräkna de nya beslutspunkterna genom att beräkna vad vi behöver addera till vår gamla beslutspunkt.



Om vi väljer punkten nedanför till höger skall vi addera följande värde till vår beslutspunkt

$$incE = F(Mx+1, My) - F(Mx, My) = 2 * dy$$

Om vi däremot väljer punkten ovanför till höger skall vi addera:

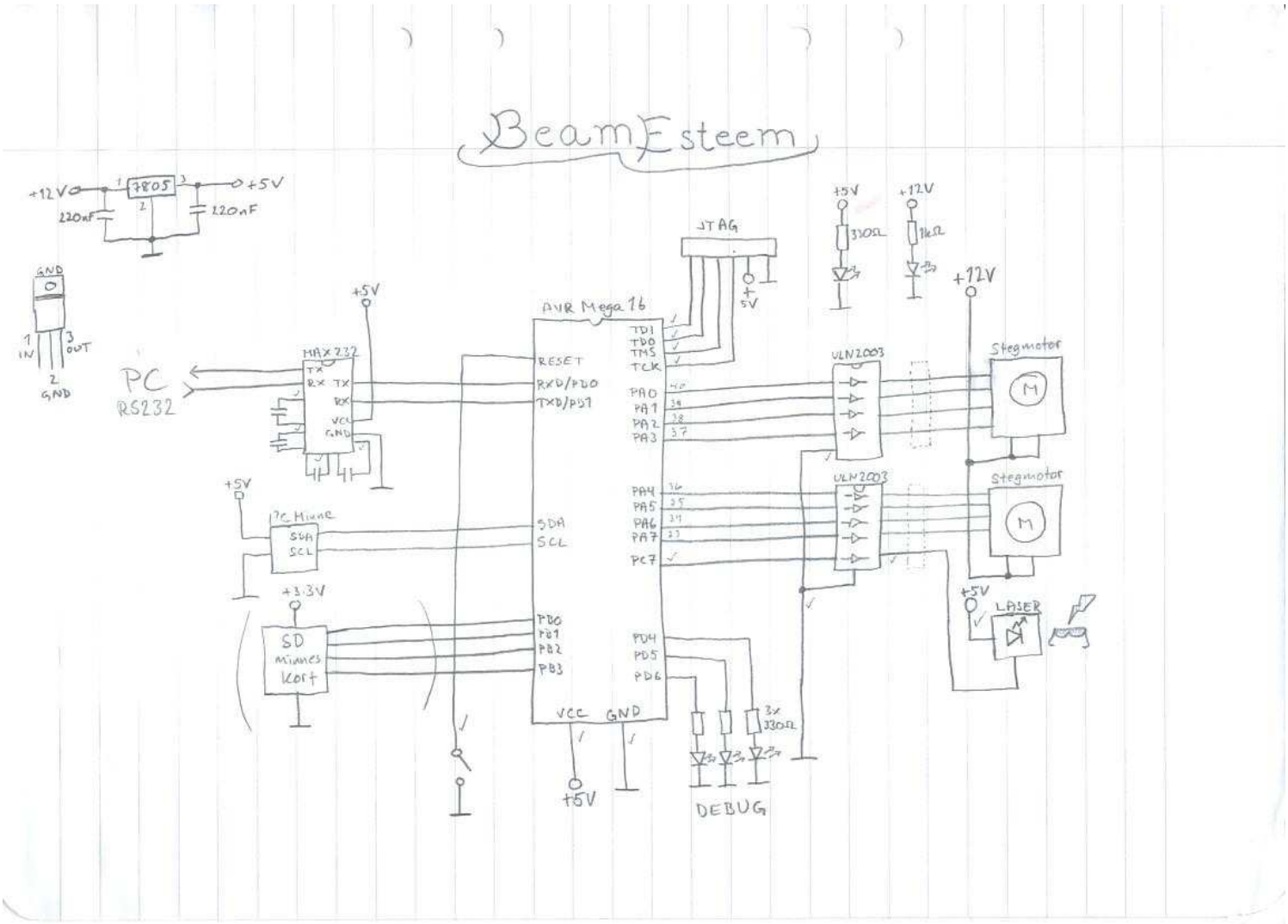
$$incNE = F(Mx+1, My+1) - F(Mx, My) = 2 * dy - 2 * dx$$

Det fina med Bresenhams algoritm som vi visat ovan är att *d0, incE* och *incNE* är heltal som bara behöver beräknas en gång då vi endast arbetar med raka linjer. När vi sedan itererar längs linjen behöver vi endast addera incE eller incNE till d0 och jämföra om d0 är negativ eller positiv för att kunna avgöra vilken pixel som vi ska färga.

Vi har gjort en del små modifieringar för att kunna använda algoritmen på våra stegmotorer. Vi är tvungna att beräkna om $|dy| > |dx|$ i detta fall är vi tvungna att invertera axlarna samt om derivatan är negativ längs någon axel är vi tvungna att vända på koordinaterna.

Kopplingsschema

Kopplingsschemat nedan är den ursprungliga designen och idag används inte i2c-minnet eller SD-minneskortet. Dessa var tänkta att kunna spara figurerna på mellan körningarna, alternativt innehålla en kortare filmsekvens.



Resultat

Systemet fungerar bra, dock finns det fundamentala begränsningar i designen såsom inbyggd tröghet och diskreta positioner i motorerna. Avsaknad av återkoppling av motorerna medför en manuell kalibrering samt svårigheter med att kompensera för trögheter i dessa, vilka ger sig tillkänna som olika typer av överslängar i skarpa hörn. Mindre komplexa figurer ritas upp snabbt och förhållandevis elegant, men allt eftersom komplexiteten ökas blir resultatet alltmer flimrande.
För att undvika flimmer vid komplexa figurer skulle en roterande spegelförsedd trumma kunna användas istället för stegmotorer.

Sammanfattning

Efter att ha fullföljt detta projekt har vi kommit fram till att stegmotorer kanske inte är det bästa sättet att styra speglarna. Dels har stegmotorer diskreta vinklar och dels är de inte snabba nog att rita upp komplexa figurer med tillfredsställande uppdateringsfrekvens. Kommersiella lasershower använder galvanometrar vilket ger ett betydligt bättre men också betydligt dyrare system.
Detta är ett billigt och enkelt system för att rita simpla figurer samt experimentera med stegmotorer en laserdiod.

Referenser

Komplett manual för ATmega 16
Drawing Lines- The Bresenham Algorithm, Graphics Tutorial by Hexar. PDF