

# *Tegnerobotten*

## *GSR1: Gruppe 1*

Uddannelse og semester:

*Robotteknologi - 1. semester*

Dato for aflevering:

*12. December 2014*

Vejleder:

*Johan Sund Laursen*

Gruppe medlemmer:

*Katrine Maria Nielsen, Andreas Espersen, Laus Skovgård  
Bigum, Mikkel Hornstrup og William Bergmann Børresen*



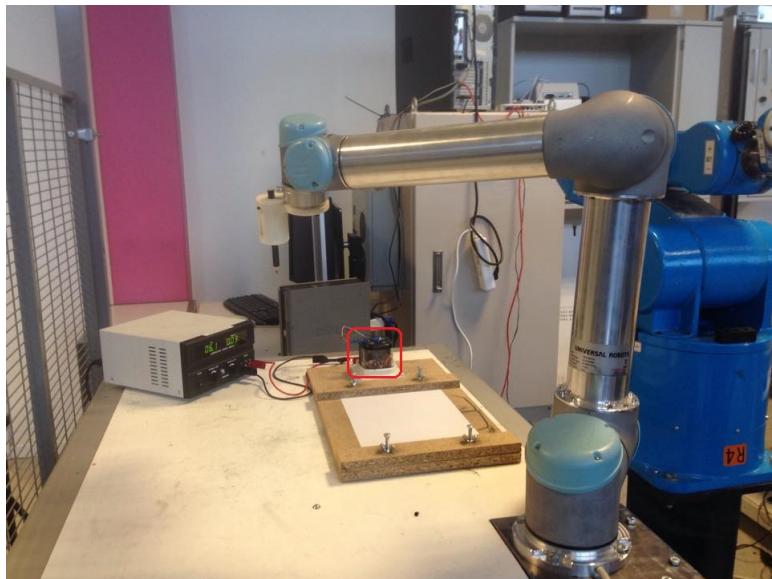
*Teknisk Fakultet  
Syddansk Universitet*

## Indholdsfortegnelse

<b>1 Resumé</b>	<b>4</b>
<b>2 Indledning</b>	<b>5</b>
2.1 Projektformulering . . . . .	5
2.2 Opbygning af rapporten . . . . .	6
<b>3 Beskrivelse af de elektriske komponenter anvendt i kredsløbet</b>	<b>6</b>
3.1 Komparator - generelt om operationsforstærkere . . . . .	6
3.2 LED . . . . .	7
3.3 MOSFET . . . . .	8
3.4 Fototransistor . . . . .	9
<b>4 Elektronik til blyantspidser</b>	<b>9</b>
4.1 Diagram over det elektriske kredsløb . . . . .	10
4.2 Overvejelser og beslutninger i elektrisk kredsløb . . . . .	11
<b>5 Teoretisk beskrivelse G-kodes opbygning</b>	<b>13</b>
5.1 Fordeler og begrænsninger ved G-kode . . . . .	15
<b>6 Program til konvertering af billede til G-kode</b>	<b>15</b>
6.1 Opbygning af Mathematica kode . . . . .	16
6.1.1 Konvertering af billede til G-kode . . . . .	16
6.1.2 Konvertering af logo til G-kode . . . . .	20
6.2 Overvejelser og valg i forbindelse med koden . . . . .	23
6.3 Illustrationer robotten kan tegne med G-koden . . . . .	26

<b>7 Fejl og usikkerheder ved robot og blyantspidser</b>	<b>27</b>
7.1 Test af eventuelle fejl . . . . .	28
<b>8 Resultat af tegning</b>	<b>29</b>
<b>9 Konklusion</b>	<b>30</b>
<b>10 Kildeliste</b>	<b>31</b>
<b>11 Indhold af DVD</b>	<b>32</b>

## 1 Resumé



Figur 1: Robotten anvendt til projektet ses holde en blyantspidser, og den modificerede blyantspidser ses i baggrunden af billedet, i den røde boks

Projektet omhandler en robot, (se figur 1), der anvendes til at tegne en billede med en blyant samt spidse denne blyant. Det forklares hvordan G-kode, som er et sprog, der fortæller robotten, hvorledes den skal bevæge sig, laves i CAS-værktøjet Mathematica. Den G-kode, som produceres i Mathematica, fortæller robotten, om den skal tegne eller bevæge sig frit i luften samt X-, Y- og Z-koordinater, som beskriver hvor i et tredimensionelt plan, robotten skal være, når den tegner. Foruden dette fortæller det også hvordan billedet skal tegnes, og når robotten skal spidse blyanten. Blyantspidseren, som anvendes i dette projekt er elektrisk og modificeret således, at den automatisk starter når blyanten stikkes i den. I dette elektriske kredsløb er anvendt forskellige komponenter, herunder to forskellige transistorer og en komperator. Det er beskrevet i rapporten hvorledes de anvendte komponenter virker, samt hvilken indflydelse de har i det nye elektriske kredsløb.

## 2 Indledning

Dette projekt er udført af 5 studerende, der går på SDU Robteklinje på 1. semester. Projektet er udarbejdet på to måneder og svarer til 10 ECTS point. Rapporten afleveres d. 12.12-2014, med tilhørerende DVD indeholdende rapporten, video af robotten mens den tegner samt koden til robotten samt G-kode til robotten.

### 2.1 Projektformulering

Dette projekt går ud på, at en robot, som kan ses på figur 1, skal tegne et billede med en blyant samt spidse blyanten når det er nødvendigt. Projektet er opdelt i to dele, elektronik til en blyantspidser, da denne skal automatiseres, og programmering af kode til robotten, der skal tegne en illustration. Robotten skal kunne tegne flere slags illustrationer med en blyant, samt fortælle hvornår blyanten skal spidses. En metode er således udviklet til at konvertere et billede til en illustration, som en robot kan tegne. Illustrationen er konstrueret af et sprog til robotter kaldet G-kode, som er genereret i CAS-værktøjet Mathematica. G-koden beskriver linjenummer, om robotten skal tegne eller lave luftkoordinater samt giver robotten X-, Y- og Z-koordinater, så den ved, hvor den skal befinde sig. Da der skal kunne laves flere forskellige illustrationer, er der lavet en generel kode, der både kan tegne gråtoner og logoer. Da robotten tegner med en blyant er der i G-koden taget højde for slid på blyanten og længden på blyanten. Foruden dette fortæller G-koden også, hvornår blyanten skal spidses. Spidsningen af blyanten gøres med en blyantspidser, som er modificeret så den automatisk starter, når blyanten sættes i den, og tilsvarende stopper, når blyanten fjernes.

#### Valg af robot

Til projektet har to robotter været til rådighed. Den robot, der hovedsagligt er arbejdet med, er en robotarm fra Universal Robots. Robotten har de nødvendige egenskaber for at opnå de bedste resultater. Den anden mulighed var en Panarobot, som havde den fejl, at den tegnede buer mellem hvert punkt, hvilket begrænser, hvor gode resultater, der kan opnås.

## 2.2 Opbygning af rapporten

Rapporten er bygget op således, at der først er en teoretisk gennemgang af komponenterne i det elektriske kredsløb, efterfulgt af en gennemgang og beskrivelse af de valgte komponenter. Det beskrives, hvorledes de hver især virker, efterfulgt af en forklaring på hvordan de har indflydelse i blyantspidserens elektriske kredsløb. Herefter følger en forklaring på hvordan G-kode virker, samt hvorledes G-koden er konstrueret for at få robotten til at tegne. Her efter diskutes eventuelle fejl og mangler i den praktiske del af projektet, til sidst kommenteres på resultatet.

## 3 Beskrivelse af de elektriske komponenter anvendt i kredsløbet

Følgende er en kort teoretisk beskrivelse af, hvorledes de forskellige komponenter, som er anvendt i det elektriske kredsløb til blyantspidseren, virker. Dette giver et overblik over, hvilke egenskaber og funktioner de enkelte komponenter bidrager med, når kredsløbet senere bliver beskrevet som en helhed.

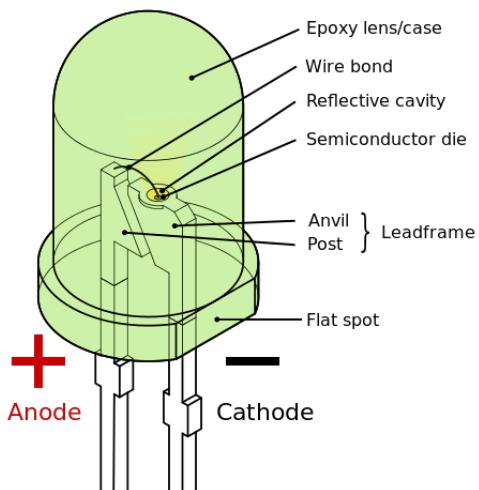
### 3.1 Komparator - generelt om operationsforstærkere

Operationsforstærkeren har et inverterende og et ikke-inverterende input. Ideelt set er operationsforstærkeren kendtegnet ved, at den har uendelige inputsimpedanser, forstærkningen er uendelig, og der er ingen udgangsimpedans. Der anvendes ofte negativ eller positiv feedback ved operationsforstærkere. Det betyder, at dele af signalet bliver sendt fra output og tilbage til det inverterende input ved negativt feedback, mens det ved positivt feedback sendes til den ikke-inverterende input. Hvis man anvender en operationsforstærker uden feedback, får man en komparator. Komparatoren er et element, som tager to strømme eller spændinger og sammenligner dem.

Operationsforstærkeren kigger på det inverterende input i forhold til det ikke-inverterende input. Hvis det ikke inverterende input er større end det inverterende input vil den forstærke outputtet så meget den kan, mens hvis det inverterende input er større end det ikke-inverterende input, vil outputtet gå ned til den negative grænse. Den maksimale forstærkning bestemmes af operationsforstærkerens spændingskilde (eks.  $\pm$

15V).

### 3.2 LED

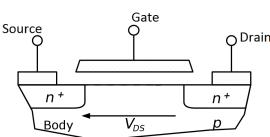


Figur 2: Oversigt over en LED (Light Emitting Diode)

Dioden har to terminaler (som ses på figur 2), anoden og katoden. Hvis man sætter en lille, positiv spænding til en diode vil den få en høj strøm. Dette kaldes forward bias. Hvis der på den anden side er en negativ spænding, skal denne være noget større for at få en strøm til at løbe. Dette kaldes reverse-bias. Dette anvendes til at lade en strøm løbe den ene vej gennem en diode, men blokere for strømmen i modsat retning. Dioder består af et knudepunkt mellem halvledermateriale. På den ene side af knudepunktet dannes n-materiale, hvor der er et stort overskud af frie elektroner, mens der på den anden side dannes p-materiale, hvor der er overskud af "huller". Dette betyder, at selvom man ikke sætter nogen strøm til, vil der stadig være et elektrisk felt, som holder de frie elektroner på n-siden og hullerne på p-siden. Hvis en ydre positiv spænding sættes på n-siden, kan ladningsbærerne ikke krydse knudepunktet, og der går ingen strøm. Hvis der derimod sættes en positiv spænding på p-siden, bliver feltet mindre, og der vil derfor kunne gå en strøm. En LED, som er en light emitting diode, er således en diode, som lyser, når den aktiveres. Dette sker, når en elektron flyttes til et hul, udgiver den energi i form af en foton.

### 3.3 MOSFET

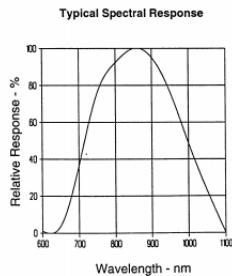
Transistorer anvendes i al elektronik, hvor hukommelse anvendes. Den mest anvendte transistor er MOSFET'en, som er en af to transistorer, der bliver anvendt i kredsløbet til styring af blyantspidseren. Til forskel fra en almindelig npn transistor, der er strømstyret, er MOSFET'en spændingstyret. Fordelen ved en MOSFET er, at den ikke kræver en særlig stor strøm (tæt på ingen) ind i gaten for at aktivere den, men den kan stadig (for nogle typer) leve op til 50 ampere eller mere til et load. Størrelsen af strømmen, den kan håndtere, afhænger af hvordan MOSFET'en er bygget, og dens fysiske størrelse.



Figur 3: Oversigt over en MOSFET (Metal-Oxide-Semiconductor Field-Effect Transistor)

MOSFET'en har fire terminaler (som ses på figur 3); source, drain, gate og body. Der bliver dog ofte kun refereret til de tre førstnævnte, da source og body er ofte forbundet med hinanden, og dermed danner én terminal (kaldet source). I kredsløbet for blyantspidseren bruges en npn-type, og den anvendes som en switch, der fungerer ved, at den har et ON- og OFF-stadie. Om den er ON eller OFF afgøres af spændingsfaldet over gate- og sourceterminalen. Ved OFF-stadiet er der intet spændingsfald over gaten, hvilket forhindrer elektroner i at strømme fra drain til source, da der dannes et bånd af ioner, kaldet en depletion region, som reflekterer elektronerne kommende fra drain til source. Sættes der spænding over gaten, vil MOSFET'en tillade, at der løber en strøm ved at hjælpe en vis mængde elektroner gennem depletion regionen. Denne mængde afhænger af størrelsen på spændingen over gaten. Jo højere spænding desto større flow af elektroner. I kredsløbet for blyantspidseren er spændingen på gaten altid enten nul eller en bestemt værdi, der ikke varierer (styret af operationsforstærkeren), så den som nævnt kan fungere som en switch, der enten afbryder strømmen, eller lader den passere til blyantspidserens behov ved at simulere en kortslutning. Source er forbundet til ground og har derfor et potentiale på 0V.

### 3.4 Fototransistor

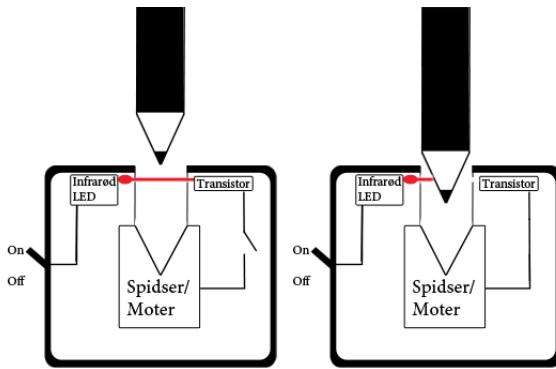


Figur 4: Diagram over fototransistorens respons på forskellige bølgelængder af lys

Fototransistoren fungerer stort set ligesom transistoren, som er beskrevet under afsnittet MOSFET. Dog har den ”almindelige” transistor tre andre terminaler end MOSFET’en: En base, en emitter og en collector. I modsætning til MOSFET’en, der behøver en spænding over sig for at starte strømfloret gennem transistoren, skal en fototransistor bruge lys for at åbne. Fototransistoren åbner gradvist op, afhængigt af hvilken bølgelængde den modtager - den kan eksempelvis åbne op for en hvis bølgelængde i det infrarøde spektrum. Forskellige bølgelængder har forskellige fordele ift. dens anvendelse. Vil man undgå, at fototransistoren bliver påvirket af almindeligt lys (ca. 400-700 nm), kan man eksempelvis vælge en, der åbner for infrarødt eller ultra violet lys. Graden, der bliver åbnet op med, afhænger af lysets indfaldsvinkel ift. sensoren, hvor kraftigt lyset er, og hvilke bølgelængder fototransistoren modtager. Kigger man f.eks. på figur 4, har fototransistoren en respons mellem 40% og 100% i bølgelængderne, der svarer til infrarødt lys. Ved ca. 860 nm vil transistoren have en respons på 100%, hvilket vil sige, at transistoren lader en maksimal mængde strøm passere.

## 4 Elektronik til blyantspidser

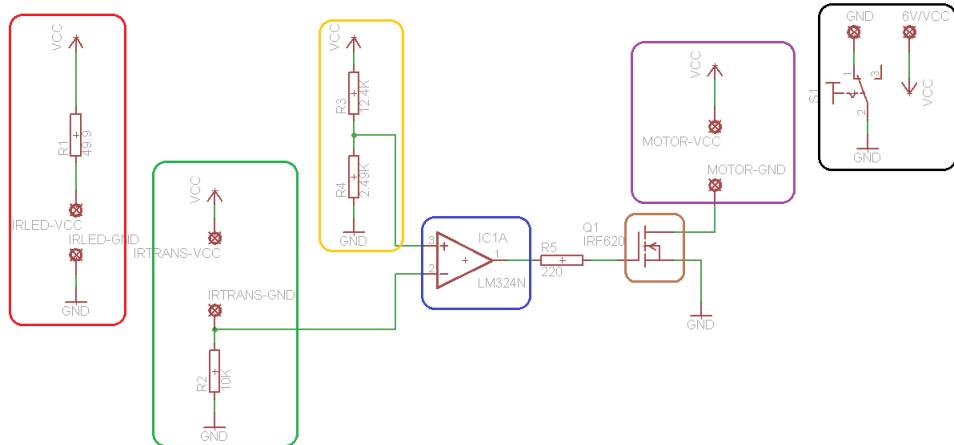
I nedenstående beskrives hvorledes en elektrisk blyantspidser er blevet automaticeret, så den starter, når en blyant sættes i den. På figur 5 kan det ses, hvordan det er lavet, så hverken knapper eller andre udefrakommende faktorer skal anvendes for at igangsætte blyantspidseren. Først kommer et afsnit hvor diagrammets opbygning er beskrevet, hvorefter der kommer et afsnit med forklaringer på, hvorfor de forskellige komponenter



Figur 5: Illustration af hvorledes blyantspidseren aktiveres

er anvendt.

#### 4.1 Diagram over det elektriske kredsløb



Figur 6: Kredsløbet til blyantspidseren

Figur 6 illustrerer det elektriske kredsløb i blyantspidseren. LED'en, den lysfølsomme transistor og motoren ses ikke på illustrationen, da disse ikke sidder på printet, men er trukket væk fra printet med ledninger. Det står dog skrevet, hvor i kredsløbet de er placeret.

Tilsluttet spændingskilden er en  $49.9 \Omega$  modstand. Denne er serieforbundet med en LED (OP298) der udsender infrarødt lys til en lysfølsom transistor (OP598). Mellem den lysfølsomme transistor og ground, er koblet en  $10k \Omega$ modstand. Parallelt med disse er koblet den inverterende terminal af en operationsforstærker (LM324N). På den

ikke-inverterende terminal er der tilkoblet to modstande i parallelforbindelse. Disse er dimensioneret til hhv.  $12.4\text{k}\Omega$  og  $2.49\text{k}\Omega$ . På outputtet er koblet en  $220\Omega$  modstand, der er forbundet til gate-indgangen på den efterfølgende MOSFET (IRL8113).

MOSFET'ens source går til ground, og drain går til blyantspidserens motor som er tilsluttet til vcc som er 6V.

Helt ude til højre på figur 6 kan man se de to steder hvor ground og 6V er tilsluttet. Mellem ground og vores kredsløb har vi placeret en 3 punkts vippe kontakt.

## 4.2 Overvejelser og beslutninger i elektrisk kredsløb

Som det ses af den røde boks markeret på figur 6 starter kredsløbet med en  $49.9\ \Omega$  modstand efterfulgt infrarød LED.

### 49,9 $\Omega$ modstand

Grunden, til at modstanden er netop denne størrelse, er, at det er aflæst af databladet, at der skal bruges max 100mA til LED'en. Foruden dette er det aflæst, at der omtrent vil være et spændingsfald på 1.5V over LED'en. Det kan således antages, at spændingsfaldet over modstanden vil være 4.5V, da ingangsspændingen er 6.0V. Sådan er størrelsen af modstanden bestemt.

$$\frac{U}{I} = R \quad (1)$$

↑

$$\frac{4.5V}{100 * 10^{-3}A} = 45\Omega \quad (2)$$

Da det nu vides, at modstanden mindst skal være  $45\Omega$ , er der valgt en lidt større modstand for at sikre, at strømmen holdes under - men nær - 100mA.

### Infrarød LED

LED'en, som sidder efter modstanden, er infrarød. For at undgå mest mulig støj og forstyrrelser fra udefrakommende lyskilder, der kunne sende lys ned i blyantspidseren og på den måde aktivere kredsløbet.

Som det ses i den grønne boks på ovenstående illustration kommer der efter LED'en en lysfølsom transistor.

### Lysfølsom transistor

LED'en udsender lys til en lysfølsom transistor med et filter, der kun lukker lys ind med bølgelængder over 650nm, men har toppunkt mellem 800nm og 900 nm. Så længe transistoren modtager lys vil den åbne for strømmen fra forsyningen - hvor stor en strøm der passerer, afhænger af hvilken bølgelængde transistoren modtager, og hvor præcist lyset rammer den. Transistoren lukker 100 % strøm igennem ved en bølgelængde på 860 nm, hvilket svarer til det lys der udsendes fra LED'en.

### 10k $\Omega$ modstand

Efter den lysfølsomme transistor kommer en 10K $\Omega$  modstand. Modstanden er sat, så der er noget at måle spændingen over. Størrelsen på denne er bestemt ud fra databladet, hvori det er angivet, at ved større spændinger kan man opretholde 100% strømgennemgang over større distancer. Desuden er denne størrelse også valgt, da der efterfølgende kommer en operationsforstærker i kredsløbet. Pga denne modstand vil den outputte 0V, hvis spændingen over det inverterende input er størst.

### Komparator/operationsforstærker

Det ses i den blå boks på figur 6, at der netop her i kredsløbet er en operationsforstærker. Strømmen løber fra fototransistoren og videre til det inverterende input på operationsforstærkeren, der sammenligner dette input med det ikke-inverterende input. Er spændingen over den inverterende indgang størst, vil operationsforstærkeren outputte 0V, da dens negative forsyningsspænding er forbundet til den tidligere beskrevede 10k $\Omega$  modstand, som er forbundet til ground. Er spændingen derimod størst over den ikke-inverterende indgang, hvilket den er, når blyanten er i blyantspidseren, da den lysfølsomme transistor ikke åbner for strømtilførsel, vil der outputtes den størst mulige spænding. Da den positive forsyningsspænding er sat til 6V, vil outputtet nærme sig dette.

### Spændingsdeler

På den ikke-inverterende indgang er koblet to modstande, som udgør en spændingsdeler. Dette ses på figur 6 tidligere i den gule boks. Disse modtande er 12.40 k $\Omega$  og 2,49 k $\Omega$ . De er valgt, da der ønskes at spændingen over den ikke-inverterende indgang skal være

1V. Ses af følgende:

$$\frac{R_4}{R_3 + R_4} * V_C C = V_+ \quad (3)$$

$$\frac{2.49\Omega}{12.40\Omega + 2.49\Omega} * 6V = 1.00V \quad (4)$$

Foruden dette er størrelsen på modstandene valgt til at være kilo. Dette skyldes, at der ønskes en meget lille strøm, og der skal derfor vælges nogle store modstande. Disse modstande må dog heller ikke være for store, da der i så fald ikke vil gå nogen egentlig strøm.

### MOSFET

Efter operationsforstærkeren, som maksimalt tillader et output på 6V, kommer en modstand på  $220\Omega$ . Denne er indsat for, at mindske spændingen over MOSFET'en. Til MOSFET'en skal være en spænding på mindst 2V, da netop denne MOSFET åbner for strømgennemgang ved spændinger større end dette. Motoren er koblet på MOSFET'ens drain og starter, når MOSFET'en lukker strøm igennem.

## 5 Teoretisk beskrivelse G-kodes opbygning

For at give en forståelse af, hvad der giver robotten informationer om, hvor den skal tegne, er følgende lidt grundlæggende information om det G-kode, der er arbejdet med. G-kode er en variant af programmeringssproget NC (numerical control) og bruges til at fortælle robotter, hvordan de skal bevæge sig. Dette indebærer blandt andet hvilke bevægelser, der skal udføres hvornår, hastigheden af en bevægelse og hvilken position robotten skal bevæge sig til. G-kode kan eksempelvis bruges til at styre bevægelserne for en robot, man vil have til at udskære et stykke træ (eller andet materiale) til en bestemt form, vha. et værktøj holdt af robotten.

### Eksempel på format af G-kode:

N10 G00 X-0.1 Y0.1 Z0.01

N20 G01 X-0.1 Y0.1 Z-0.001

N30 G01 X-0.1 Y-0.1 Z-0.002

N40 G01 X0.1 Y-0.1 Z-0.003

N50 G01 X0.1 Y0.1 Z-0.004

N20 G01 X-0.1 Y0.1 Z-0.005

N60 G98 Z-0.01

Kigger man på det fra en ende af, starter koden med et 'N' efterfulgt af et tal. Tallet angiver linjenummeret, som bestemmer, hvornår den givne kommando skal køres ift. resten af kommandoerne. Disse skal stå i rækkefølge, så N100 f.eks. ikke kommer efter N200. Selve værdien af tallet betyder ikke noget, så længe de står i den rigtige rækkefølge.

Efter linjenummeret kommer selve handlingen/funktionen, som robotten skal udføre. Dette angives med 'G' efterfulgt af en værdi, der refererer til handlingen. Eksempelvis bruges G00 som regel til en bevægelse fra et punkt til et andet, hvor robottens led bevæger sig med maksimal hastighed. Dette medfører, at den overordnede bevægelse (værktøjets bevægelse fra et punkt A til et andet punkt B) ikke altid er lineær, da ledenes bevægelser som udgangspunkt ikke koordineres med hinanden. G01 bruges, når robotten skal udføre et egentligt arbejde på en overflade. Bevægelserne er mere kontrollerede, da der interpoleres mere mellem koordinaterne for at opnå lineære bevægelser. I forbindelse med tegnerobot projektet bruges der også en G98 kommando, der i dette tilfælde er spidsefunktionen - eller nærmere koordinatet for blyantspidserens placering.

Til sidst skal der angives X-, Y- og Z-koordinater til hver funktion (bortset fra G98, som kun skal bruge Z-koordinaten). Her refererer Z til armens bevægelse i højderetningen, dvs. hvor tæt armen er på overfladen. Afstanden fra overfladen tager udgangspunkt i længden af værktøjet, så ved Z0 rører værktøjet ved papiret, men uden yderligere tryk/pres fra robottens side. I forhold til projektet (hvor værktøjet er en blyant), vil Z0 ikke længere referere til overfladepunktet, efterhånden som blyanten bliver kortere - Z-værdien bliver altså negativ for at tage højde for dette. X- og Y-koordinaterne angiver bredde- og længdekoordinaterne. Alle X-, Y- og Z-koordinater er angivet i meter.

Med denne viden kan man igen kigge på eksemplet på en G-kode, der blev givet tidligere, og på forhånd drage nogle konklusioner for, hvad den vil betyde for robottens bevægelse. G-koden vil producere en kvadratisk firkant med sidelængden 20 cm og til sidst spidse blyanten. Da X- og Y-værdierne er angivet ift. et forudbestemt nulpunkt - hvilket for tegnerobotten er bestemt til midten af det A4 ark, der tegnes på - vil firkanten have centrum i papirets centrum. Der er dog flere ting, der gør, at denne kode ikke vil

virke i praksis (bl.a. øges trykket med en millimeter for hvert koordinat (10cm), hvilket ikke er helt realistisk), men for simplicitetens skyld giver det et godt overblik.

### 5.1 Fordele og begrænsninger ved G-kode

G-kode har nogle få parametre, der kan manipuleres. X-, Y- og Z-koordinaterne giver mulighed for at bevæge værktøjet i tre dimensioner, men man kan eksempelvis ikke rotere værktøjet om sig selv (give det en vinkel). Ift. tegneprojektet er det en fordel ikke at have så mange variable at skulle styre, under forudsætning af at ledene selv sørger for at korrigere sine vinkler, så det hele passer, når X-, Y- og Z-værdierne varieres. For at tegne er det ikke nødvendigt at give blyanten en vinkel, og det er derfor en fordel ikke at skulle bekymre sig om vinklen af blyanten. For G98 funktionen er det ydermere kun Z-koordinaten, der kan manipuleres med, og hvis G98 ikke er defineret til det helt rigtige X- og Y-koordinat, kan det betyde, at værktøjet ikke rammer præcist det punkt, som det er tiltænkt. I afsnittet om fejlkilder senere i rapporten beskrives det, hvorledes dette kan give nogle problemer ift. blyanten og blyantspidseren.

## 6 Program til konvertering af billede til G-kode

I dette afsnit vil der først komme en gennemgang af koden, der er skrevet i Mathematica. Efter dette kommer et afsnit med overvejelser og valg i forbindelse med koden. I gennemgangen af koden vil der blive vist nogle brudstykker af koden - den fulde kode kan findes på den vedlagte DVD. Formålet, med den Mathematica kode der laves, er, at man kan konvertere et billede eller logo til G-kode. Koden fungerer ved, at man åbner og evaluerer Mathematica-filen. Herved åbnes et GUI, hvori man kan uploadet et billede. I dette GUI kan man indstille forskellige ting, som f.eks. hvor mange pixels, der skal være, og om det er et billede eller logo. Når man er færdig med indstillingerne, trykker man ”save G-kode”. Herved genereres G-koden ud fra indstillingerne og gemmes som en .nc fil, der kan bruges i simuleringen til robotten.

## 6.1 Opbygning af Mathematica kode

Selve programmet til at konvertere et valgt billede til G-kode, er delt op i tre dele og sat ind i et GUI. De tre dele består af en del til billeder, og to dele til logoer. De tre dele minder meget om hinanden - der er dog forskel på opbygningen af dem, og den måde de agerer på. Delen til billeder kører med tre forskellige gråtoner. De to dele til logoer kører kun med farverne sort og hvid. Den ene tegner kanten rundt om et logo, mens den anden fylder logoet.

Mange af de ting, der indgår i koden, er konceptuelle, og vil derfor ikke blive beskrevet her, mens der vil blive lagt mere vægt på de mere interessante dele af koden. De konceptuelle ting er bl.a. nogle konstanter, der har forskellige værdier - herunder startværdier for forskellige variabler, og konstanter for spidsning og lignende. Billedet, der skal laves til G-kode, bliver i alle tre dele importeret med kommandoen Import. Billedet bliver herefter behandlet forskelligt alt afhængigt af, om det er et logo eller et billede.

### 6.1.1 Konvertering af billede til G-kode

I dette afsnit vil delkoden, der konverterer billedet til G-kode, blive gennemgået.

#### Behandling af billedet

```
bil=ImageData[ColorQuantize[ImageResize[ColorConvert[g, "Grayscale"], a], 3, Dithering -> False]]
```

Figur 7: Behandlingen af billedet

Efter billedet er blevet importeret, bliver det behandlet med flere indbyggede funktioner fra Mathematica. Dette kan ses på figur 7. Der bliver brugt en funktion, der hedder ColorConvert. Denne funktion bruges til at lave billedet om, så det kun består af gråtoner. Dernæst bruges en kommando kaldet ImageResize, der gør det muligt, at bestemme hvor mange pixels billedet skal bestå af. Den sidste kommando, der bliver brugt til at behandle billedet, er ColorQuantize. Med denne kommando kan man lave billedet om til at bestå af et færre antal farver. I tilfældet med denne kode er det valgt, at billedet skal bestå af tre farver, og da det i forvejen var lavet til gråtoner, kommer det til at bestå af tre gråtoner.

Efter billedet er blevet behandlet, udtrækkes dataene fra billedet vha. en funktion kaldet `ImageData`. På denne måde fås en matrix, hvor hvert tal i matrixen beskriver en farve for en pixel, og da der er tre forskellige gråtoner, vil der indgå tre forskellige talværdier i matrixen - en for hver gråtone.

### **"Spidskode" til spidsning af blyant**

```
spidskode := "N"<>ToString[n++]<>" G98 Z"<>ToString[nulpunkt+bly]
```

Figur 8: G-kode linje for spidsning

For at styre linjenumrene i G-koden laves en variabel `N`, der tæller en op, hver gang der laves en G-kode linje. `Z`-værdien i G-koden er tilpasset således der bliver taget højde for slid og placeringen på tegneplanet. På den måde er der hele tiden et konstant tryk, så der kan blive tegnet. Ser man på en G-kode linje for en spidsekmando, ses det at denne kun skal have tre argumenter. Denne G-kode linje for spidsning kan ses på figur 8. `Z`-værdien er givet ved sliddet, der ved hver spidsning bliver regnet ud til et nyt nulpunkt. Til dette nulpunkt er der lagt et konstant tryk til, da blyantsspidsen ved `Z0` er i plan med blyantspidserens top. Denne G-kode funktion for spidsning af blyantspidseren er ens gennem hele programmet.

### **Yderste del af G-kode funktionen**

```
For[i=1,i<b,i++,
  For[iq=Union[bil[[i]]]; wq=3|,
    Length[Select[iq,#<bild[[wq]]&]]>0,wq--|,
    gg = den i'te liste med to af de lyseste punkter i hver ende
    If[wq==2,kode;kode,kode]]];
  ];
```

Figur 9: Yderste del af G-kode funktionen

Kigger man på koden udefra og ind, kan man skabe sig et lidt bedre overblik. Dette kan ses på figur 9. Det ønskes, at den mest lyse gråtone skal være hvid, og tegnes derfor ikke. Den midterste gråtone skal være grå og dermed kun tegnes en gang, og den tredje og mørkeste gråtone skal tegnes flere gange for at opnå den mørke farve. Da det ønskes at tegne som en printer, tages der udgangspunkt i en linje af gangen. Dette gøres ved at lave en for løkke, der løber fra 1 til antallet af rækker i matrixen med bilde dataene. Med dette kan man nu i for løkkens fokusere på en række af gangen. I denne for løkke er en anden for løkke, som først tester om den i'te række indeholder nogle gråtoner, der skal

tegnes. Derefter tester for løkken om rækken indeholder punkter med den mørkeste gråtone, der så skal tegnes flere gange. Hvis testen er sand, laves der en liste med den i'te række af billeddataene. I hver ende af denne liste bliver der tilføjet to værdier, som svarer til den lyseste gråtone, der altså ikke skal tegnes. Viser det sig, at der er noget i den i'te række, der skal tegnes, når løkken kører første gang, køres funktionen "kode" en gang. Hvis det er anden gang løkken kører, og der er nogle punkter af den mørke gråtone i den i'te række, så køres funktionen kode to gange. Alt dette vil resultere i, at den lyseste gråtone ikke bliver tegnet, den midterste gråtone bliver tegnet en gang, og den mørkeste gråtoner bliver tegnet i alt tre gange.

### Opbygning af funktionen "kode"

```

kode := If[Er linjen ulige, ww = Append[ww, Table[
    If[Er der et tegnepunkt,
        If[Er alle tre punkter tegnepunkter,
            counter++; If[Skal der spidses,
                {tegnluft1, luft1, spidskode, spidskode, spidskode,
                 luft1,
                 tegnluft1}, Ellers intet],
                If[Skal der spidses, {tegnluft1, luft1, spidskode,
                    spidskode,
                    spidskode, luft1, tegnluft1},
                     tegnluft1]], Ellers intet]
            , fra 2 til x dimensionen + 3]];
    Ellers tegne fra den anden vej];

```

Figur 10: Opbygningen af funktionen G-kode

Ser man lidt nærmere på funktionen "kode"s opbygning, ses det, at den er opbygget af hovedsageligt if og table kommandoer. Dette kan ses på figur 10. Først tester den om linjen skal tegnes fra den ene eller anden side - dette gøres vha. en OddQ funktion, der tester på en variabel, som tæller en op, hver gang der er tegnet en linje. Hvis OddQ er sandt, bruges der en Append. På den måde bliver der tilføjet en liste fra table til listen ww. Tablen kører hvor den ændrer j fra 2 til længden af rækken og lægger tre til. Inde i tablen testes det først med en if kommando om et eller flere af tre punkter, der kommer efter hinanden, skal tegnes. Hvis dette er tilfældet, testes der igen med en anden if kommando, hvis ikke det er rigtigt, sker der ikke noget. Vha. denne nye if kommando testes det, om alle de tre punkter skal tegnes; skal de det, tælles counter først en op. Counter er en variabel, der tæller en op, hver gang der bliver lavet en tegne G-kode. Efter counter er talt op, testes det om blyanten skal spidses - hvis den skal, laves der et nyt

nulpunkt, hvor sliddet er lagt til det tidligere nulpunkt. Counter sættes til 1, og der laves en liste af tegnluft1, luft1, spidsekode, spidsekode, spidsekode, luft1 og tegnluft1. Dette gør, at blyanten først bliver løftet og derefter spidses den tre gange. Herefter kommer den på plads med en luftkode, og fortsætter med at tegne. Begrundelsen for disse valg kommer senere under afsnittet om overvejelser og valg. Hvis testen viser, at blyanten ikke skal spidses, vil counter blive talt en op. Hvis ikke alle de tre punkter, der testes, skal tegnes, testes det alligevel om blyanten skal spidses. Dette gøres på samme måde som tidligere. Hvis den ikke skal spidses, køres funktionen tegnluft1. Går man tilbage til testen af, hvilken retning der skulle tegnes fra, kan man se, at når der tegnes fra den modsatte side en først beskrevet, er der indsat en konstant, og j er trukket fra. På denne måde tæller det ned i stedet for at tælle op.

### Tegne- og luftfunktioner

```
tegnluft1 := "N" <> ToString[n++] <>
    " G0" <> ToString[If[Er det et luft punkt, 0, 1]] <>
    " X" <> ToString[Pxd*(j - 2) - forskyda] <>
    " Y" <> ToString[Pxd*i - forskydb] <>
    " Z" <> ToString[If[Er det et tegnepunkt,
        counter++; (nulpunkt + slidprPx*(counter - 1) +
            konstanttryk + Z[i*Pxd - forskydb]),
        If[Er Z mellem 0.00001 og - 0.00001,
            substitueres 0.0001 fra luft; (nulpunkt + slidprPx*(
                counter - 1) + luft),
            (nulpunkt + slidprPx*(counter - 1) + luft)]]];
;
```

Figur 11: Opbygningen af funktionen tegnluft1

Det kan ses, at de fire funktioner luft1, luft2, tegnluft1 og tegnluft2 minder meget om hinanden. Derfor kigges her kun nærmere på funktionen tegnluft1. Tegnluft1 er en funktion, der laver en G-kode linje. Funktionen tegnluft1 kan ses på figur 11.

Linjenummeret bestemmes som tidligere beskrevet. Efter det testes det, om dette er en luft eller tegne G-kode linje, der skal laves, hvilket afgører, om der skrives G00 eller G01. Efter dette bestemmes X- og Y-værdierne ud fra koordinaterne til punktet. Der fratrækkes en forskydningsværdi fra koordinaterne til punktet, for at centrere billedet på midten af papiret, da X0 og Y0 ligger i centrum af papiret. Det sidste, der bliver beregnet i funktionen, er Z-værdien. Denne værdi bestemmes ved først at teste, om det er et tegne- eller luftkoordinat. Er det et tegne koordinat lægges der en til counter, og Z-værdien bestemmes som nulpunktet lagt sammen med sliddet, det konstante tryk og

den værdi, der er som følge af placeringen på planet. Hvis det er et luftpunkt, testes der først om værdien vil blive for lille. Er den det trækkes der 0.1mm fra variablen luft. Z-værdien bestemmes så ved at lægge nulpunktet, sliddet og luft sammen. Her gør konstanten, at luftpunktet bliver hævet ca 1cm over planet, så der dermed ikke tegnes.

### Eksportering af G-kode

Når alt dette er kørt igennem, vil listen ww indeholde al G-koden. Altså er hver punkt på listen en G-kode linje på string form. Denne liste eksporteres som en tekstfil til en valgt placering. Herefter oprettes der en batch fil, der kan lave den bestemte fil om til en .nc fil. Denne batch fil køres, hvorefter den slettes igen. Efter dette er G-koden færdig og skrevet som en .nc fil, der kan køres i simuleringen til robotten.

#### 6.1.2 Konvertering af logo til G-kode

Der kigges nu på den del af programmet, der arbejder med et logo i stedet for et billede. Denne logo-del kan opdeles i to dele: En del der udfylder logoet, og en del der laver kanten. Den del, der laver G-koden til fyldet af logoet, minder meget om den del, der laver et billede om til G-kode, da de begge vil tegne på samme måde -forskellen er, at der ved logoet kun tegnes én gråtone.

#### Behandling af logoet

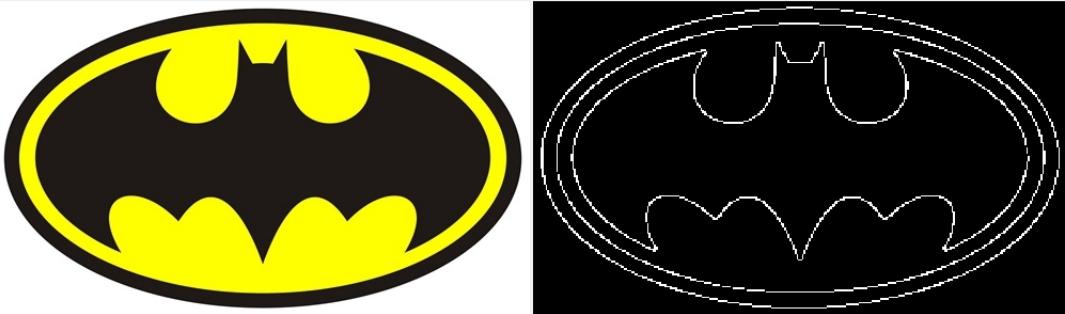
```
jk = MorphologicalBinarize[ImageResize[g, a], {s1, s2}]
```

Figur 12: Behandling af logo

Både delen til kanten og til fyldet skiller sig ud fra billeddelen i form af deres behandling af billedet. Efter det er importeret, bruges funktionen ImageRezise for at få billedet med det antal pixel, der ønskes. Herefter bruges der en indbygget funktion (kaldet MorphologicalBinarize) til at behandle billedet. Dette kan ses på figur 12. Funktionen laver billedet om til sort og hvid. Det fungerer ved, at den tager udgangspunkt i en øvre og nedre grænse for kontrasterne, og ud fra disse værdier bestemmes det hvor meget, der skal være sort. Man har via GUIen mulighed for at vælge automatisk - så funktionen selv finder grænserne - man kan vælge selv at bestemme den øvre grænse, eller selv at bestemme både den øvre og nedre grænse.

Da resten af den del, der laver G-koden til fyldet af logoet, minder meget om delen, der

laver billedet til G-koden, vil denne del ikke blive beskrevet yderligere. I stedet vil der blive kigget lidt nærmere på den del, der laver G-koden til kanten af logoet. I delen, der



Figur 13: Et eksempel på brugen af EdgeDetect

laver kanten, bliver logoet yderligere behandlet med en funktion, der hedder EdgeDetect. Denne funktion finder kanten hele vejen rundt om logoet. Et eksempel på dette kan ses på figur 13. Bagefter bruges funktionen ImageData, hvorved man får en matrix, der indeholder alle punkterne, hvor kanten af logoet er givet ved 1-taller, og resten er givet ved nuller.

```
list2=Prepend[Position[bil, 1][[Last[FindShortestTour[  
Position[bil, 1]]]]], {-2, -2}];
```

Figur 14: Opbygningen af list2

For at komme til at tegne kanten af logoet mest effektivt, anvendes funktionen FindShortestTour, hvilket kan ses på figur 14. Men for at denne kan udnyttes, bruges først funktionen position. Position bruges til at lave en liste af koordinatsæt, til de punkter der skal tegnes - altså ettallerne i matrixen. På denne liste bruges FindShortestTour, hvilket resulterer i, at man får en liste med tal, der angiver rækkefølgen af koordinatsættene for den korteste tur. For at få arrangeret koordinatsættene i denne rækkefølge, tages disse elementer fra listen med koordinatsættene. Til denne liste tilføjes et start punkt, der ligger i -2,-2. Dette bruges senere, når G-koden skal laves.

### **Yderste funktion af G-kode funktionen**

For at gøre det lidt mere simpelt ses der ligesom ved billedet til G-kode, på koden udefra og ind. Dette kan ses på figur 15. Selve G-kodefunktionen til at lave kantens G-kode består af en table, hvor der i enden bliver tilføjet en G-kodelinje for det sidste punkt som

```

gcodek[x_List] := Append[Table[
  If[Er afstanden mindre eller lig 2,
    If[Skal der spidses,
      {tegnekode, luftkode, spidskode, luftkode, tegnekode},
      tegnekode],
    If[Er det begyndelses punkt, {luftkode, tegnekode},
      {luftkode2, luftkode, tegnekode}]],
  , k fra 1 til anden sidste punkt],
  Sidste punkt som luftkoordinat];

```

Figur 15: Yderste funktion af G-kode funktionen

luftkoordinat. Tablen laver en liste ud fra den funktion, der står i den, hvor k løber fra 1 til antallet af koordinatsæt der skal tegnes. Inde i tablen testes der først om afstanden mellem to på hinanden følgende koordinatsæts X- og Y-værdier er mindre eller lig med 2. Dette gøres ved at trække henholdsvis X-værdierne fra hinanden og Y-værdierne fra hinanden, og se vha. MemberQ, om disse værdier er indeholdt på en liste, der består af heltal fra -2 til 2. Hvis afstanden er mindre eller lig to skal der tegnes mellem punkterne. Derfor testes der først om blyanten skal spidses. Skal blyanten spidses, sker der det samme som i koden til billedet, og skal den ikke spidses køres funktionen "tegnekode". Er afstanden mellem punkterne derimod mere end 2, skal der ikke tegnes mellem dem, og der skal luftkoordinater mellem dem. Er dette tilfældet, testes der, om det er det første punkt, hvis der er køres funktionerne "luftkode" og "tegnekode". Er det ikke det første punkt og afstanden er større end to køres funktionerne "luftkode2", "luftkode" og "tegnekode".

### Luft- og tegnefunktionerne

```

tegnekode := "N" <> ToString[n++] <> " G01" <>
  " X" <> ToString[Pxd*list2[[k+1, 2]] - forskyda] <>
  " Y" <> ToString[Pxd*list2[[k+1, 1]] - forskydb] <>
  " Z" <> ToString[counter++; (nulpunkt + slidprPx*(counter-1) +
  konstanttryk + Z[Pxd*list2[[k+1, 1]] - forskydb])];

```

Figur 16: Opbygning af funktionen tegnekode

Kigges der nærmere på funktionerne "luftkode", "luftkode2" og "tegnekode", minder disse meget om de funktioner, der var under billede til G-kode. Den eneste forskel er, at X- og Y-værdierne beregnes direkte ud fra koordinatsættene. Koordinatsættet k plus 1 tages, og så bruges det første tal til X-værdien og det andet til Y-værdien. Som eksempel

på dette er opbygningen af tegnekode vist på figur 16.

### Eksportering af G-koden

Når G-kode funktionen for kanten og/eller fyldet er kørt igennem, flettes de to lister med G-kode sammen, så kanten kommer først og derefter fyldet. Denne sammenfletning eksporteres og laves om til en .nc fil på samme måde som med billedet, så man ender med en .nc fil med G-kode.

## 6.2 Overvejelser og valg i forbindelse med kodden

I forbindelse med kodningen i programmet Mathematica, har der været forskellige overvejelser. De overvejelser og valg, der er blevet truffet på baggrund af dem, vil blive beskrevet i dette afsnit.

### Behandling af billedet

En af overvejelserne gik på hvilken rækkefølge billedet skulle behandles i, og hvilke funktioner der skulle bruges til dette. En ting, der gør det nemmere, at arbejde med billedet og bestemme hvilke områder, der skal være mørke, og hvilke, der skal være lyse er, at lave det om til gråskala. Fordelen ved dette er, at når man senere bruger ImageData, vil de værdier man får ud være 1 og 0, hvor 0 er hvid, og 1 er sort. Dette er meget nemmere at regne med, end når der er farver indblandet, for så vil dataene have alle mulige værdier, alt efter hvilken farve det er, og det er dermed sværere at sortere efter, hvor mørke de er. Ud fra hvordan og hvor mørkt blyanten tegner, fås det bedste resultat ved at tegne med to gråtoner. Derfor ønskes det at lave så billedet, så de kun består af tre farver, hvor den lyseste ikke tegnes, den midterste tegnes en gang og den mørkeste tegnes flere gange. Billedet skulle også behandles med ImageRezise, for at få det ønskede antal pixels. Ses der på rækkefølgen af behandlingen, ses det, at ColorQuantize er den sidste funktion, der behandler billedet. Dette er for at sikre, at der kun er tre farver. Blev den i stedet behandlet med funktionen før ImageRezise, ville der fremkomme flere end tre farver efter ImageRezise er brugt. Grunden, til at ImageRezise kommer efter ColorConvert i behandlingen af billedet, er, at det vurderes at give det bedste resultat, når farverne i billedet bliver lavet til gråtoner, så man kan se, hvor mørke de forskellige farver er, inden man laver om på antallet af pixels, og dermed også ændrer på farneværdien af de forskellige pixels.

### Tilpasning til planet

En anden vigtig overvejelse har været omkring en tilpasning, til det plan som robotten skal tegne på. Dette plan ligger ikke helt i XY-planet. Det er ikke ret meget, at planet hælder i forhold til XY-planet, derfor kan man vælge at gøre problemet an på flere måder. Man kan f.eks. vælge at tilføje et konstant tryk, så der vil blive tegnet på det laveste punkt. Dette vil medføre, at der er et væsentligt større tryk i det punkt hvor planet har den højeste Z-værdi. Denne løsning i øvrigt giver den ulempe, at der vil blive ændret lidt i den gråtone der bliver tegnet med. Den vil altså ikke blive ens over det hele. Det er derfor valgt, at løse problemet ved at lave en funktion, der tilpasser Z-værdien til planet. Denne tilpasning er fundet ved at lave et forsøg, hvor robotten fik en G-kode, så den holdte blyanten stille i tre af planens hjørner. I disse tre positioner bliver X- og Y-værdierne aflæst fra G-koden, og Z-værdien bliver målt. Ud fra disse tre punkter beregnes planets ligning, og det ses at X ikke indgår, hvilket vil sige, at planen er parallel med X-aksen. I denne ligning isoleres Z, og der fås en Z-værdi som funktion af Y-værdien. Ved at bruge denne ligning i koden, bliver Z-værdierne tilpasset til planet, og det er dermed lettere at holde et konstant tryk over hele planet.

### Hæufigheds af spidsning

Der har været nogle overvejelser i forbindelse med, hvor ofte blyanten skal spidses. Dette hænger sammen med, hvor godt blyantspidseren fungerer. Der er også meget forskel på, hvor ofte den skal spidses alt afhængigt af om det er et logo eller billede, da blyanten ønskes mere spids ved tegning af et logo frem for et billede. Ved logoet er det blevet valgt, at der skal spidses en gang for hver 25cm. Dette er på baggrund af en slidtest, hvor der under tegneprocessen blev holdt øje med spidsen på blyanten, og hvornår det mentes, at den var for flad. Ved spidsningen bliver der trykket lidt ekstra, da blyanten kun er kort tid nede i blyantspidseren. Sådan som robotten kører, når den skal spidse blyanten, er, at den bliver løftet hen over blyantspidseren, hvor den holder en lille pause inden den hurtigt kører ned og op igen. Dette gør, at den ikke er i blyantspidseren ret længe. Men når robotten er kommet langt hen i programmet, og den negative Z-værdi er blevet en del større, kommer den til et punkt, hvor den i stedet for at holde en pause over blyantspidseren, og være kort tid nede i den, vil den også holde pausen nede i blyantspidseren. Dette medfører, at der er meget forskel på hvor lang tid blyanten er i blyantspidseren. Det har ikke nogen indflydelse på programmet fra logo til G-kode, da der ikke bliver slidt så meget af, at den får en Z-værdi, der er så stor, at det er et problem.

Derimod bliver det et problem i forbindelse med billedet til G-kode, fordi der flere steder vil blive tegnet oven i hinanden, og dermed bliver der slidt mere. Dette problem er blevet afhjulpet ved, at blyanten holdes i blyantspidseren i en bestemt Z-værdi, som ønskes at være det nulpunkt. Hver gang der skal spidses, gøres det tre gange. Dette er nødvendigt, da der ikke er et særligt hårdt tryk ned i blyantspidseren, så den spidser ikke særlig meget den korte tid blyanten er i blyantspidseren. Men tre gange er nok, til at der bliver spidset ned til det nye nulpunkt. Ved at gøre det på denne måde, sikres det, at når blyanten holdes i blyantspidseren i lang tid, spidser den ikke længere ned end til det nye nulpunkt. Det er valgt, at der i billede til G-kode skal spidses for hver meter. Denne værdi er foretaget ud fra en vurdering af hvad der ser godt ud, når der tegnes. Det er ikke lige så ofte som ved logo, fordi blyanten ved logo ønskes spids, og den ved billede ikke ønskes helt spids, da det ikke ser pænt ud, når der tegnes med flere gråtoner.

### **Bestemmelse af sliddet**

Under bestemmelsen af slid på blyanten var der nogle overvejelser. Først skulle der skabes et overblik over, hvor stort sliddet ville være. Dette blev gjort ved at lave en test, hvor der blev tage en spids blyant og tegner 8 meter med den. Det blev målt, at blyanten var blevet 3mm kortere. Denne test giver et slid på 0,375mm pr. meter for en spids blyant, der tegner 8 meter. Da blyanten bliver holdt spids under tegningen af et logo, vil blyanten også slide mere af, da en spids blyant bliver slidt mere end en flad. Denne teori blev testet efter, og det gav at sliddet er på 0,5mm pr. meter, når der tegnes et logo. Ved billedet blev der derimod taget udgangspunkt i, at sliddet ville være mindre, da blyanten var betydelig mere flad. Ud fra test med dette udgangspunkt, blev der fundet frem til, at sliddet ved tegning af et billede ville være 0,25mm pr. meter.

### **Tegne fra begge sider**

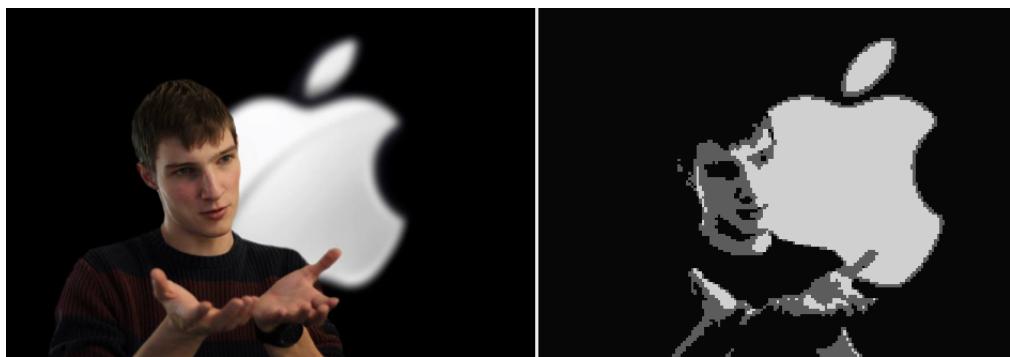
En overvejelse har også gået på, hvordan selve udfyldningen skulle foregå. Det blev besluttet, at robotten skulle udfylde ligesom en printer, da det er den simpleste måde blev vurderet til at give den pæneste løsning. Det kunne derefter vælges, om der kun skulle tegnes fra den ene side eller fra begge sider. Da det mest effektive ville være at tegne fra begge sider, og derved skifte retning for hver linje, blev dette valgt. Det blev i koden gjort ved at teste, om linjen har et ulige eller lige nummer.

### **Tegne i gråtoner**

Der var også en overvejelse om, hvordan billedet skulle tegnes i flere gråtoner. De to

bedste løsninger var at prikke med blyanten, eller tegne flere gange på samme sted for at skabe den mørkeste nuance. Den hurtige løsning blev her valgt, nemlig at tegne, da det er betydeligt hurtigere end at løfte blyanten for hvert punkt. Det blev valgt, at den lyseste gråtone ikke skulle tegnes, men bare forblive hvid. Den mellem-mørke farve skulle tegnes med én streg, og den mørkeste farve skulle tegnes tre gange for at en tydelig forskel skulle kunne ses.

### 6.3 Illustrationer robotten kan tegne med G-koden



Figur 17: Konverteringen af billedet giver ikke det ønskede resultat da hudfarven og baggrunden går i ét

Ud fra G-koden (beskrevet i ovenstående) er det nu muligt, at lave flere forskellige slags illustrationer. Det er muligt at lave stort set alle logoer, som består af hvid og én anden farve. Disse kan i øvrigt laves på forskellige måder som er at lave kanten, fyldet eller begge dele. Det kan i øvrigt justeres hvor mange pixels der tegnes med, så kvaliteten af logoet kan reguleres. Foruden det kan der også tegnes billeder med tre forskellige gråtoner: hvid, lys grå og mørk grå. Det er dog ikke alle billeder, som kan tegnes på denne måde. Det kræver, at billedet hverken er meget mørkt eller meget lyst, samt at der er tydelige kontraster i billedet. Hvis man eksempelvis prøver at tegne en meget mørk figur, som har en meget mørk baggrund, vil der ikke kunne skelnes mellem disse. For at opnå en højere kvalitet kan man ligesom med logoer også øge antallet af pixels. Desuden kan man med billeder også indstille grænserne for kontrasten i billedet. Dette betyder, at man kan bestemme, hvilken nuance der i billedet skal være hvid, lys grå eller mørk grå.

## 7 Fejl og usikkerheder ved robot og blyantspidser

- Blyanten følger ikke robottens præcise bevægelse, men sakker en smule efter robottens bevægelse. På grund af dette slør kan der også komme et mellemrum på eksempelvis en linje, hvis robotten skal over og spidse (Se figur 18). Skal en illustration udfyldes med farve, vil nogle kantlinjer være forskudt i forhold til fyldet. Dette problem kan ikke umiddelbart løses, men kan afhjælpes en smule ved at vælge at tegne med en blyant, der ikke er meget lang. Det er dog vigtigt at vælge en blyant, som er lang nok til, at den både kan slides og spidses mens der tegnes.
- Blyantspidseren spidser ikke ordentligt, da den igennem forløbet er blevet slidt. Desuden rammer blyanten skævt ned i blyantspidseren, da koordinatet for G98 ikke er defineret helt nøjagtigt ift. blyantspidserens indgang. Dette har den betydning, at det er nødvendigt at spidse flere gange. I koden skulle det i øvrigt tilpasses, at der i takt med, at blyantspidseren blev dårligere skulle ændres mindre i Z-koordinatet i G-koden, da den tog mindre af blyanten.
- Hvis blyanten er hel spids, bliver de grå nuancer mindre tydelige, da den tegner mørkere. Det bliver således uklart, hvilken nuance af grå der tegnes. Dette problem kan løses ved at starte med at tegne med en blyant, der ikke er hel spids, og i øvrigt spidse kortere så blyanten ikke bliver hel spids undervejs.
- Robotten mister nogle gange kontakten til computeren, der styrer den. Dette betyder, at tegningen skal startes forfra.
- Når robotten fra Universal Robots anvendes, er planet denne tegner på ikke helt lige, der skal derfor tages højde for dette når der tegnes med denne. Panets ligning skal derfor ganges på Z-koordinaterne i G-koden, for at tilpasse at robotten trykker lige hårdt alle steder. Man kunne også have undladet at finde en løsning på dette problem, men så ville der ikke være et jævnt tryk over hele planet.
- Modstandene i kredsløbet er ikke målt efter, samt der er ikke taget højde for modstand i ledninger og komponenter. Det må derfor antages at strømme og spændinger i kredsløbet ikke har de antagede værdier.

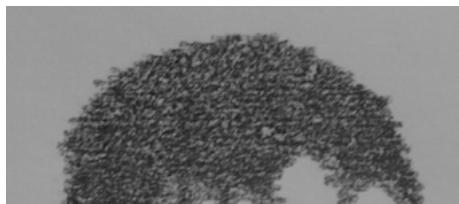


Figur 18: Her tegnes en lige linje med robotten, først fra det ene hjørne af papiret og herefter fra det andet hjørne af papiret, med samme midtpunkt

## 7.1 Test af eventuelle fejl

På figur 18 burde robotten ramme præcis samme midtpunkt fra begge sider, da den har fået nøjagtig samme koordinatsæt fra begge sider. Det gør den ikke og dette skyldes den tidligere nævnte fejkilde hvor blyanten ikke følger robottens præcise bevægelse, og det kan derfor ikke løses fuldstændigt. Det vil give et udfald når der tegnes tegninger, da den ikke vil have fuldstændig samme afstand mellem linjerne, når den ”printer” tegningen.

I fejkilderne er der beskrevet et slør i holderen på blyanten, som gør, at blyanten stopper for tidligt på en linje. Dette er tydeligt på tegningen, ved at hver anden linje er lidt forskudt fra den forrige. Dette skyldes, at robotten skiftevis, linje for linje, tegner fra hver sin side. Det kan ses lidt tydeligere, hvis der zoomes ind på tegningen, se figur 19.



Figur 19: Illustration af fejkilde

Det kan ses, at en masse af linjerne stikker længere ud end hvad meningen var: Dette er dog ikke tilfældet. Det er hver anden linje, der ikke er tegnet lang nok, i det blyanten bliver løftet, mens blyanten slæber efter holderen. Dette er den mest dominerende fejkilde, men for helheden af tegningen på figur 20a, skabes der stadig et flot resultat.

## 8 Resultat af tegning



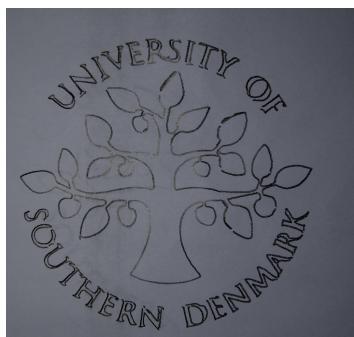
(a) Robottens tegning af figur 20b



(b) Orignal billede

Figur 20: Sammenligning af tegning og billede

Som det kan ses på figur 20, kan man tydeligt se forskel på de tre forskellige farver i billedet. Det giver den ønskede effekt og giver muligheden for at tegne mange forskellige billeder. Så længe billedet ikke er for monoton i farve og belysning, er der stort set mulighed for at tegne alt. Derudover kan det programmerede kode indstilles, til at tegne pæne logoer, hvor der ikke er fokus på farvenuancer, men på hårde og skarpe kanter. På figur 21 ses et logo tegnet med samme program, hvor logo er valgt som indstilling.



(a) Robottens tegning af figur 21b



(b) Orignal billede

Figur 21: Sammenligning af tegning og billede

Som det ses på figur 21, er det muligt at tegne logoer. Dette er SDU's logo, tegnet hvor kanten er tegnet. Det kunne også laves således, at både kanten og fyldet er lavet. Det ses på figuren som er lavet med 500 pixels, at tegningen er meget eksakt. Denne ville være betydeligt mere upræcis hvis den var lavet med færre pixels.

## 9 Konklusion

Dette projekt gik ud på at få en robot til at tegne et billede med en blyant, samt spidse blyanten når det var nødvendigt. Det er lykkedes at automatisere en blyantspidser ved at ændre på det elektriske kredsløb. Måden hvorpå dette lykkedes var ved hjælp af lysfølsom transistor overfor en infrarød LED. Kommer der lys igennem denne lysfølsomme transistor, som er koblet sammen med det inverterende input på en operationsforstærker, gør dette at det inverterende input bliver større end det ikke-inverterende input, og således lukkes der for videre strømtilførsel. I en modsat situation, hvor blyanten stikkes i, bliver det ikke-inverterende input større end det inverterende input, og der går derfor en strøm, som tænder motoren og dermed starter blyantspidseren. I Mathematica er lavet G-koder, der beskriver hvorledes robotten skal bevæge sig, og fortæller om den skal tegne eller ej, samt hvilke koordinater den skal tegne. Robotten kan både tegne logoer, samt billeder med grå nuancer. Desuden er det bestemt og angivet hvor ofte blyanten skal spidses i blyantspidseren. Opgaven er således opfyldt, tilfredsstillende resultater er opnået, og det er nu muligt at tegne flere forskellige slags illustrationer og spidse med den modificerede blyantspidser med en robot.

## 10 Kildeliste

### Figure 2

- [http://en.wikipedia.org/wiki/Light-emitting\\_diode](http://en.wikipedia.org/wiki/Light-emitting_diode)
- Dato: 10/12/2014

### Figure 3

- <http://en.wikipedia.org/wiki/MOSFET>
- Dato: 10/12/2014

### Figure 4

- <http://pdf1.alldatasheet.com/datasheet-pdf/view/89161/OPTEK/OP598.html>
- Dato: 10/12/2014

### Electrical Engineering

- Principles and Applications
- 6th Edition
- Allan R. Hambley
- ISBN-13 978-0-273-79325-0

### Mathematica guide

- <http://reference.wolfram.com/language/>
- Dato: 10/12/2014

## 11 Indhold af DVD

Her er det beskrevet hvilke ting DVD'en indeholder. Disse ting er givet ved den efterfølgende liste.

- **Rapport:** Rapporten er som pdf, hvilket gør det nemt at bruge links og lignende
- **Kode:** Den fulde mathematica kode der gør det muligt at lave billeder og logoer til G-kode
- **Video:** Video der viser robotten der tegner et billede
- **Billeder af tegninger:** Billeder af tegninger, der er blevet tegnet med robotten.