

MEKATRONIK A

JACOB
MAX
MAGNUS



Vejledere:
Palle Christoffersen
Henrik Nøhr



Projekttitel

Eksamensprojekt

Projektperiode

20. Februar 2024 - 19. April 2024

Fag

Mekatronik A

Vejleder

Henrik Nøhr og Palle Christoffersen

Semester

6. semester 2024

Uddannelsessted

Vejle Tekniske Gymnasium

Sideantal

144

Omfang

111.406 tegn m. mellemrum, svarende til
ca. 46 normalsider a' 2400 tegn m.
mellemrum

Rapporten omhandler en fremstillingen af en drinksmaskine, som forsøger at automatisere drinksudskænkning, således problematikken om at skænke en drink gøres bekvemmelig for brugeren. Igennem rapporten dokumenteres maskin- og el-tekniske overvejelser og argumentationer for forskellige produktvalg.

Indholdsfortegnelse

Indledning	6
Projektbeskrivelse	6
<i>Problemformulering</i>	6
<i>Tidsplan</i>	6
<i>Kravspecifikation</i>	7
<i>Ideer og løsningsmuligheder</i>	8
Løsningsforslag 1.....	8
Løsningsforslag 2.....	9
Løsningsforslag 3.....	10
Løsningsforslag 5.....	12
<i>Valg af løsning.....</i>	13
<i>Blokdiagram</i>	14
<i>Afgrænsning af produktet</i>	15
<i>Krav til endelig løsning</i>	16
Produktets virkemåde	17
<i>Teknisk afsnit for den mekaniske del.....</i>	18
Materialevalg	18
<i>Produktets dele i metal.....</i>	19
Stel	19
Roterende del	22
Enclosure.....	23
Produktets dele i plast	26
Flowdiagram over samlingen af produktets enkeltdele	27
Stykliste	28
Anvendte maskiner	29
Redegørelse for 3D-printede modeller	32
Konstruktionsberegninger	34
Teknisk afsnit for den elektriske del	39
<i>Komponenter og værktøjer</i>	40
<i>Styringsenheden - PIC16F873</i>	41

Programmering af PIC	42
<i>Kodekonstruktionen</i>	43
Assembly kode	43
Kodens opbygning.....	50
Debugging / Simulering af kode.....	53
<i>Brugergrænsefladen/USART-kommunikation:</i>	57
LCD-skærmens rolle i brugergrænsefladen.....	57
Potentiometer og trykknap.....	59
Seriel kommunikation mellem Arduino og PIC	61
<i>Roterende del</i>	62
ICM7555.....	63
BC337	64
Stepper Motor	65
4070 Xor-gate.....	66
HEF4027 Dual JK Flip-Flop	66
Hardware kontrol af Steppermotor ved brug af Flip-flop og XOR-gate.	66
MOSFET - IRF540N	68
Flybackdiode	69
Kodegennemgang:	69
Test og måling:.....	70
<i>Magnetventil</i>	71
Kode	72
Validering af signaloutput fra BC337	73
<i>Pneumatik</i>	74
<i>Reelle implementering</i>	75
Forsøg på analog kommunikation.....	75
Endelig løsning med Arduino	76
<i>Budget</i>	79
<i>Opfølgning på krav og produkt</i>	80
Kravopfølgning	80
<i>Produktopfølgning</i>	81
Magnetventil.....	82

Pneumatik	82
Stepper motor	82
Hvorfor der kun er 5 varianter af væske?	83
Konklusion	84
Perspektivering	84
Vurdering	85
Litteraturliste	86
Figur Liste	87
Tabel Liste	91
Bilag Liste	92

Indledning

Mekatronik åbner døren til et samarbejde med mekanik og elektronik, og det er netop inden for denne ramme, at dette projekt udfolder sig. Formålet er at udforske og implementere mekatroniske løsninger for at løse en specifik udfordring. Projektets formål er at kombinere teknologisk innovation med praktisk anvendelighed.

Projektbeskrivelse

Projektet har til formål at automatisere opgaver, som den almindelige borger udfører i køkkenet. På denne måde kan en del af arbejdsspresset og den tid, der bruges på forskellige opgaver, fjernes fra borgeren. Dette vil give brugeren mulighed for at anvende den frigjorte tid på andre aktiviteter, de finder mere interessante. Det valgte projekt vil fokusere på én specifik opgave, som er blevet udvalgt gennem mundtlig diskussion. Blandt de mulige opgaver, der blev diskuteret, inkluderer kaffebrygning, brødkæring, dip-blanding, krydderi-automatisering og drinks-blanding. Beslutningen er taget om at fokusere på opgaven "drinks-blanding", da denne opgave vakte størst interesse, motivation og indledende idéer til produktudformningen.

Problemformulering

Hvordan kan et avanceret automatisk system udvikles og implementeres i private hjem, så det effektivt kan automatisere gentagne husholdningsopgaver og frigøre tid for brugeren?

Tidsplan

Der opstilles en tidsplan for projektets forløb se QR kode, for på denne måde at kunne arbejde struktureret og med overblik for hvordan tiden skal bruges mest effektivt. Den fulde tidsplan ses i bilag 19.



Kravspecifikation

I tabel 1 præsenteres de specificerede krav til vores løsning. Alle krav er formuleret med relevans og inkluderer tests til at måle, om kravene er tilstrækkeligt opfyldt. Der stilles kun krav til produktet én gang, da problemformuleringen afgrænsrer løsningsforslagene til at være specifikt rettet mod denne drinksløsning. Udarbejdelsen af en omfattende kravspecifikation forud for udviklingen af løsningsforslagene muliggør, at valget af den endelige løsning bliver begrundet ud fra de foruddefinerede krav, der er fastlagt inden udarbejdelsen af løsningsforslaget. På denne måde sikres det, at valget der træffes, er et argumenteret valg.

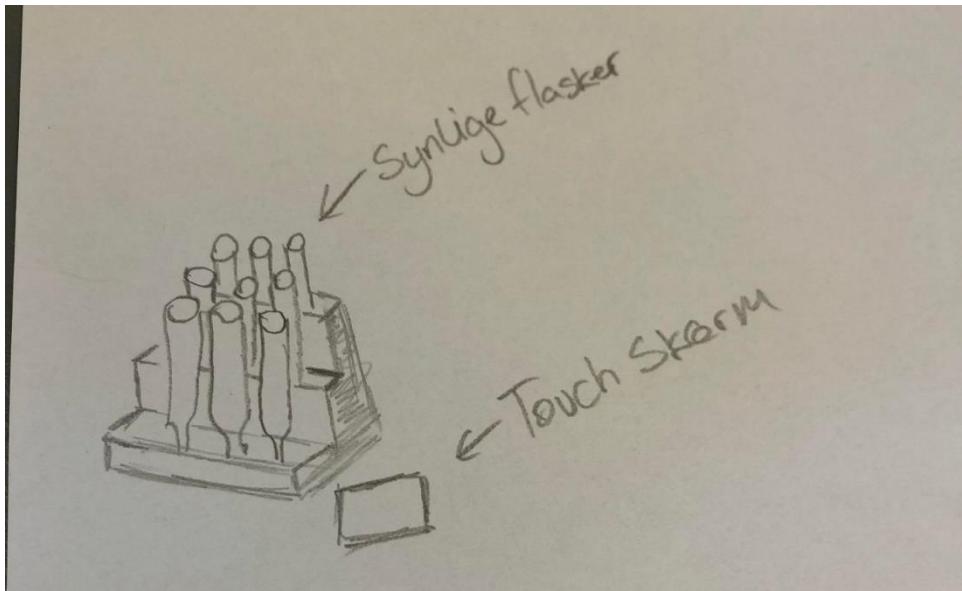
Tabel 1: Tabel over opstillede krav til løsning med relevans og beskrivelse af kravets test

Krav	Relevans	Kravet er overholdt hvis...
Brugeren skal kunne vælge mellem min. 3 forskellige drinks	7/10	Løsningen giver brugeren mulighed for at vælge mellem tre eller flere forskellige typer af drinks
Skal være tidsbesparende ift. manuel blanding	9/10	Løsningen tager mindre tid end ved manuel udførelse af opgaven.
Skal give brugeren en 'oplevelse' når der udskænkes	5/10	Løsning gør brug af bevægelige dele eller andre funktioner der gør det interessant for brugeren at afvente udskænkning
Skal være nem at fylde op	6/10	Løsningen kan nemt blive fyldt op igen.
Løsningen skal fungere uden menneskelig interaktion, efter den valgte drink.	10/10	Løsningen serverer drinken automatisk uden interaktion fra brugeren.
Løsningen skal være flytbar	7/10	Løsningen kan flyttes og stadig fungerer

Ideer og løsningsmuligheder

I følgende afsnit beskrives de ideer og løsningsforslag, som er opstillet for at kunne løse problemformuleringen.

Løsningsforslag 1

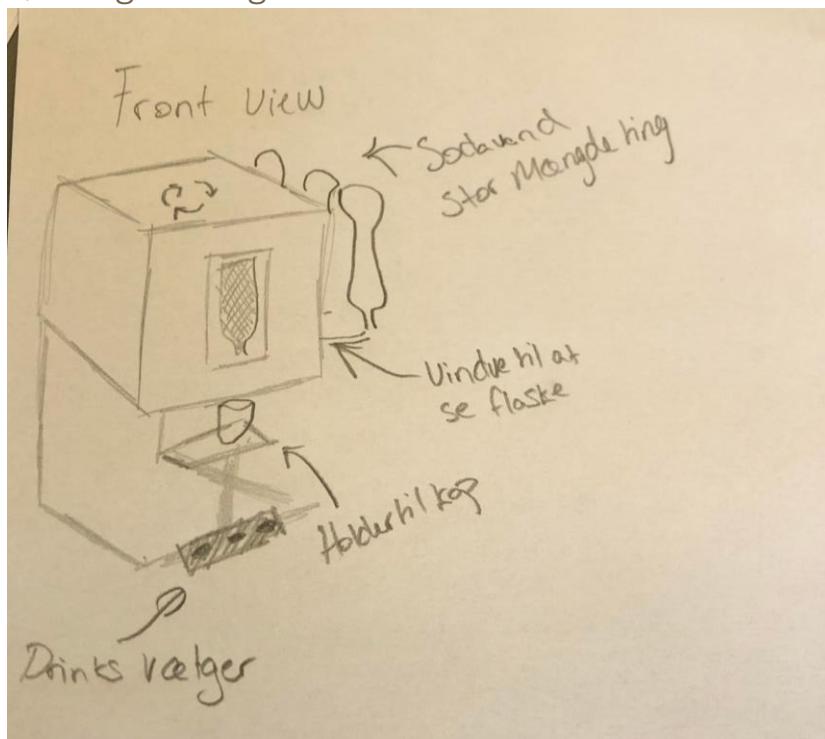


Figur 1: Skitse af løsningsforslag 1

Beskrivelse af løsningsforslag 1

Løsningsforslag 1 er baseret på ideen om visuel præsentation af samtlige ingredienser, der kan kombineres i ens drink. De ni flasker anbringes i en trappeformet base. I stedet for den originale flasketop monteres en specialdesignet kapsel på hver flaske. Ved isætning i basen fastlåses flasken via en klik-funktion i den nye kapsel. Basissystemet fungerer efter principippet i en Draughtmaster Flex, som er en fustagehane, hvad angår udtag af væske fra flaskerne og videre ud gennem dispenseringsmekanismen. Touchskærmen giver brugeren mulighed for præcist at konfigurere drikkens sammensætning efter egne præferencer, hvilket lagres til fremtidig brug. Ved hver af de ni flasker i basen er indbygget elektrisk regulerede dyser, der giver brugerkontrol over volumen af de individuelle ingredienser. Eksempelvis kan man vælge flaske 1 på skærmen og derefter angive det ønskede antal cl, der skal dispenseres ned i glasset.

Løsningsforslag 2



Figur 2: Skitse af løsningsforslag 2

Beskrivelse af løsningsforslag 2

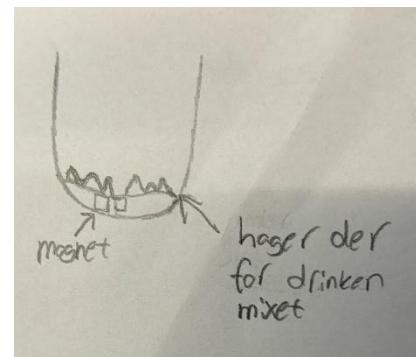
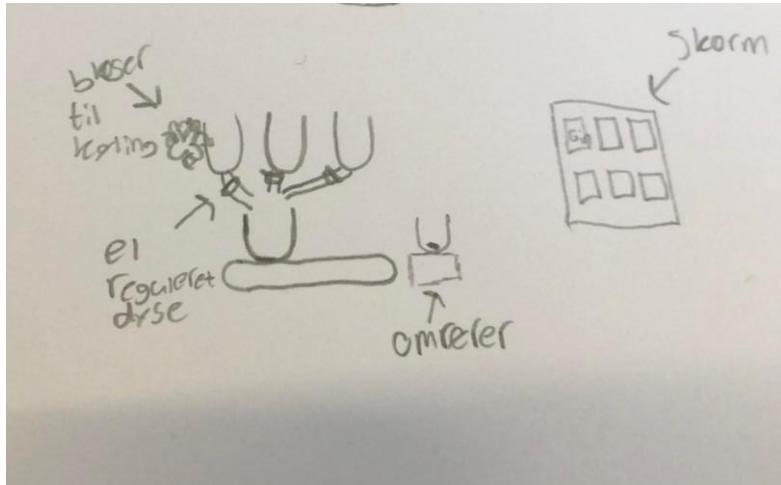
Løsningsforslag 2 er baseret på konceptet af en automatisk drinksmixer, som kan blande tre forudindstillede drinks i overensstemmelse med brugerens præferencer. Maskinen betjenes via en simpel drinksvælger placeret i bunden af produktet, hvor brugeren blot skal trykke på en enkelt knap for at få den ønskede drink. Maskinens funktionalitet er baseret på en cirkulær plade, der har plads til at montere 6-8 flasker med flaskehalsen pegende nedad. Denne cirkulære konstruktion er monteret på en stepper motor, der roterer pladen efter input fra **Piccen**, så den korrekte flaske positioneres over koppen i overensstemmelse med brugerens drinkvalg.

Når den ønskede flaske er korrekt placeret over koppen, hvorfra drinken skal blandes, tappes den nødvendige mængde væske fra flasken. Doseringen af den korrekte mængde sikres ved at skrue flasken direkte på en kuglehane, som kan åbnes og lukkes ved hjælp af en stepper motor. Dette gør det muligt at styre åbningsmekanismen for kuglehane gennem kode, og den nøjagtige mængde kan doseres ved at styre tidsintervallet, hvor kuglehane er åben. Hvis en drink kræver væsker fra flere flasker, skal maskinen også kunne dosere fra flere forskellige flasker, hvilket også styres gennem koden.

Som vist på figur 2, lægges der vægt på løsningsens æstetiske udtryk, hvilket blandt andet ses i detaljerne som et vindue, der giver mulighed for at se den forreste flaske, der tappes fra. Ønsket om et æstetisk udtryk skyldes den tilsigtede anvendelse af maskinen, som forventes at være placeret i køkkenet, ligesom en almindelig kaffemaskine. Dette vil altså være et B2C salg.

For at opfylde kravet om nem genopfyldning placeres sodavandsflasker (Cola, lemon, tonic mm.) på ydersiden, da disse væsker hyppigst vil skulle udskiftes. På denne måde skånes brugeren for at åbne enclosuren hvori glasflaskerne med alkoholiske drikke er placeret, hver eneste gang man er løbet, tør for opblanding.

Løsningsforslag 3



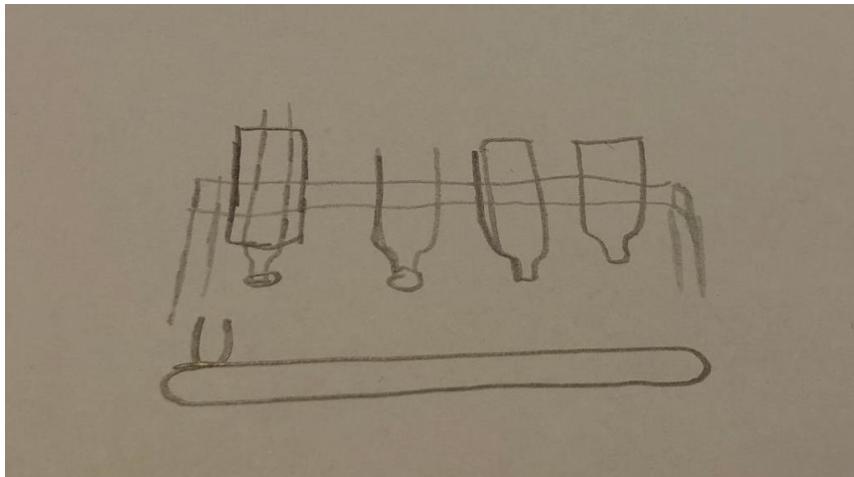
Figur 3: Skitse af det special designet glas til løsningsforslag 3.

Figur 4: Skitse af løsningsforslag 3

Beskrivelse af løsningsforslag 3

Den foreslæede løsning til B2B-salg er optimeret med henblik på hastighed og skalerbarhed frem for æstetisk design. Systemet fungerer ved, at glas bevæger sig på et transportbånd. Glassene er specielt designet med en magnetisk drejbar bund, der kan omrøre drikken uden brug af en ekstra magnet i glasset (se figur 3). Glassene opbevares i store mængder før opsætningen vist på figur 4. Hver gang en drikkevare bestilles, sættes et tomt glas i kø bag den igangværende bestilling. Glasset transportereres på båndet under nogle dispensere med forskelligt drikkevareindhold. Afhængigt af den bestilte drikkevare åbnes dyserne til de ingredienser der skal i drinken. Ved at tilfører det hele på engang optimeres hastigheden (Det skal dog også være muligt at åbne for dyserne individuelt, for derved at imødekomme kreationen af specielle drinks.) Derefter føres drikken videre til en magnetisk omrører, hvor den særligt designede bund roterer og blander drikken. Systemet styres af en skærm, hvorpå drikkevarer kan bestilles. På sigt kunne en fremtidig mulighed være trådløs styring, så drikkevarer kan bestilles via en hjemmeside. Virksomheder kunne da få systemet til automatisk at gå i gang med at lave den bestilte drikkevare, når kunden har betalt.

Løsningsforslag 4

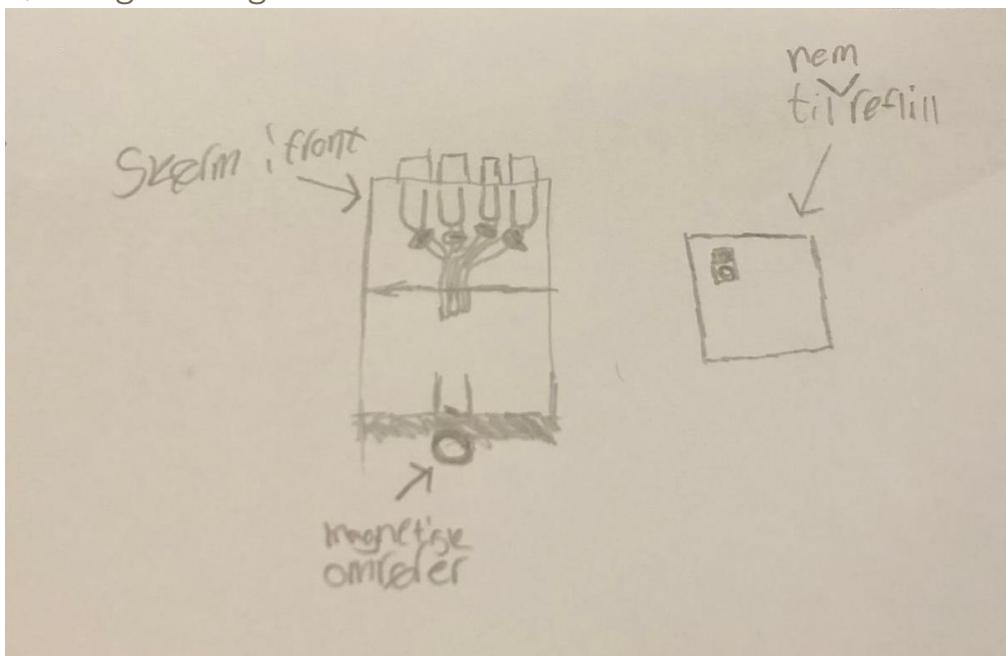


Figur 5: Skitse til løsningsforslag 4

Beskrivelse af løsningsforslag 4

Løsningsforslag 4 udforsker konceptet med at lade glasset bevæge sig i stedet for flaskerne, idet fokus rettes mod brugerens oplevelse under brugen af maskinen. Dette tilføjer en visuel dimension til brugeroplevelsen, idet det tydeligt kan ses, hvordan maskinen flytter glasset, før det ender i brugerens hånd, hvilket skaber en unik og visuel oplevelse af at få sin drink serveret. Maskinens funktionalitet er baseret på et langstrakt ophæng, hvor alle ønskede flasker er placeret i en linje. Under dette ophæng er der et transportbånd, som kan flytte glasset under den korrekte flaske baseret på brugerens valg af drink. Valget af drink bestemmes ud fra et antal forudindstillede muligheder. Som vist i skitsen på figur 5, er dette setup relativt nemt at udvide, da længden af ophængen og transportbåndet kan forlænges for at rumme endnu flere drinksmuligheder.

Løsningsforslag 5



Figur 6: Skitse til løsningsforslag 5

Beskrivelse af løsningsforslag 5.

Løsningsforslag 5 er baseret på funktionsmekanismen af en kaffemaskine, hvor glasset placeres under alle dyserne med alkoholiske og ikke-alkoholiske drikkevarer i maskinen. Via en skærm foran kan brugeren vælge den ønskede drink, dertil kan brugeren tilpasse alkoholprocenten efter eget ønske. For at optimere hastigheden kan alle dyser åbnes samtidig, sammen med at den magnetiske omrører kører kontinuerligt. Derudover er der nem genopfyldelse, da toppen af maskinen let kan åbnes, så nye ingredienser kan hældes i beholderne. Maskinen er tiltænkt B2C-salg, hvorfor dens industrielle udseende kan opfattes som en ulempe. Opsummerende optimerer løsningen hastighed og brugervenlighed via parallelisering af processerne og nem vedligeholdelse, omend vil det enkle design ikke appellere til alles æstetiske præferencer.

Valg af løsning

Det endelige valg af løsningsforslag tages med udgangspunkt i de opstillede krav sat forud for løsningsforslagenes udarbejdelse, se tabel 2. I nedenstående tabel tjekkes overholdelsen af de opstillede krav for på denne måde konkret at sikre et argumenteret valg. Under tabellen knyttes yderligere kommentarer til det konkrete valg.

Tabel 2: Tabel over løsningsforslagenes overholdelse af de opstillede krav.

Krav	Løsningsforslag				
	1	2	3	4	5
Brugeren skal kunne vælge mellem min. 3 forskellige drinks	✓	✓	✓	✓	✓
Skal være tidsbesparende ift. Manuelt*	✓	✓	✓	✓	✓
Skal give brugeren en 'oplevelse' når der udskænkes	✗	✓	✗	✓	✗
Skal være nem at fylde op	✓	✓	✗	✓	✓
Løsningen skal fungere uden menneskelig interaktion efter valgt drink.	✓	✓	✓	✓	✓
Løsningens skal være flytbar	✓	✓	✗	✗	✓

*Det antages at alle løsningsforslag vil være hurtigere end manuelt at blande drinksene

Tabel 2 giver en tydelig indikation af, hvilket løsningsforslag der bør prioriteres videre, idet det fremgår, at kun løsningsforslag 2 opfylder alle de fastsatte krav. Ikke desto mindre bør beslutningen om valg af løsning ikke alene baseres på dette faktum. Dette skyldes, at relevansen af hvert krav er blevet vægtet på en skala fra 1 til 10, hvilket betyder, at ikke alle krav har samme betydning. Det er særligt vigtigt at bemærke, at løsningsforslag 1 og 5 ikke opfylder krav nr. 3, som omhandler brugeroplevelsen, og dette krav er kun vurderet til en relevans på 5 ud af 10, og derfor stadig bør være i betragtning.

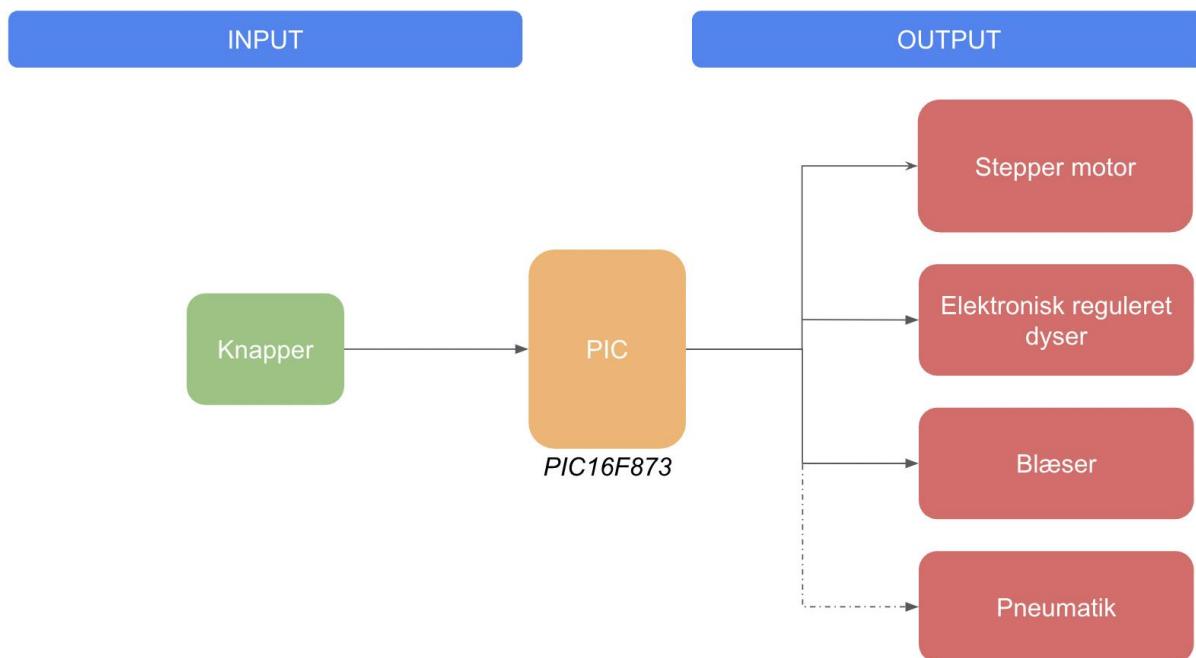
Derfor inddrages også andre overvejelser, før det endelige valg træffes. Der lægges vægt på muligheden for at integrere så mange relevante metoder som muligt fra både den elektriske og mekaniske del af mekatronikfaget. Det vurderes, at flere af løsningsforslagene kan imødekomme dette krav, men at løsningsforslag 2 gør det bedst. Dette skyldes, at der i den mekaniske del kan konstrueres en bærende

struktur til at bære vægten af alle flaskerne og samtidig en mekanisme til at rotere dem. I den elektriske del udarbejdes assembly-kode til styring af stepper-motorer gennem pic'en, og Arduino anvendes til at styre væskedoseringen i kuglehanerne for hver flaske.

Det endelige valg af løsningsforslag falder derfor på løsningsforslag 2, da det bedst muligt demonstrerer vores forståelse af både den elektriske og mekaniske side af faget i det kommende projekt. Desuden opfylder løsningsforslag 2 samtlige krav i kravspecifikationen.

Blokdiagram

For at danne en forståelse og skabe et visuelt overblik over hvordan det el tekniske skal fungere er der udarbejdet et blokdiagram i figur 7.



Figur 7: Blok diagram der illustrerer de el tekniske forbindelser

Hele systemet styres af en PIC16F873 mikrocontroller, valgt på grund af det nødvendige antal ben. PIC16F873 har 28 ben, hvoraf 22 er programmérbare. For at dække systemets behov kræves der 8 ben til input, 4 ben til stepper motorer, 8 ben til dyser og 1 ben til blæseren. Hvis der ønskes integration med en ismaskine til nedkøling af is, kræver dette endnu en stepper motor til slusen, hvilket vil kræve yderligere 4 ben. Derfor vil det i stedet være nødvendigt at anvende en PIC16F877 mikrocontroller med 40 ben, hvoraf 33 er programmérbare, for at opfylde systemets samlede behov.

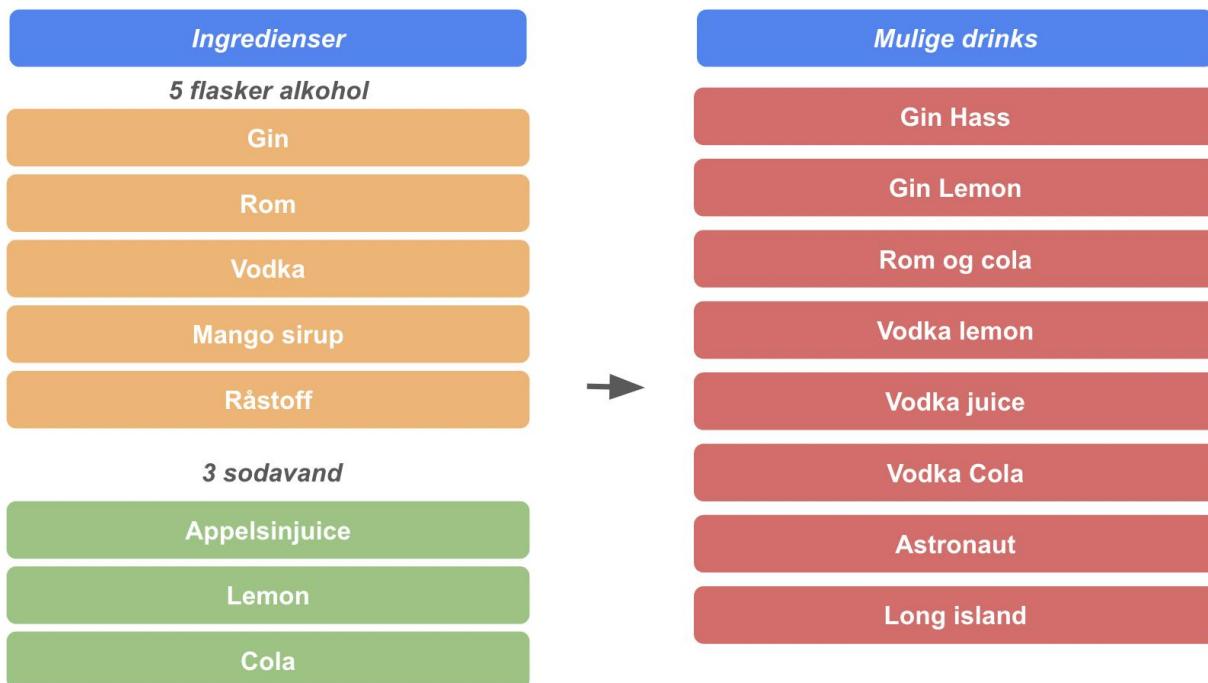
PIC'en styrer systemet på baggrund af de 8 knapper der bruges til valg af drink, herefter sender PIC'en signaler til stepper motoren om position/vinkel for adgang til den ønskede ingrediens. Derefter sendes signal til den elektronisk regulerede dyse ved den pågældende ingrediens, så den åbner i et bestemt antal sekunder for at opnå den nødvendige volumen i drinken. Stepper motoren bevæger sig til næste ingrediens og gentager processen. Til sidst vender stepper motoren tilbage til startposition, hvorefter

sodavandsdyserne aktiveres. Under hele processen kører en blæser for at illustrere, at en endelig løsning skal afkøle ingredienserne for at servere en kold drik. Som vist i blokdiagrammet, vil gruppen implementere dette som udgangspunkt, men hvis tiden tillader det, vil der inkorporeres en beholder til isterninger, som skal tilføres den færdige drink.

Hvis elektronisk regulerede dyser ikke er tilgængelige, tilsluttes i stedet en Arduino før dyserne. PIC'en og Arduinoen kommunikerer via seriel forbindelse, hvorefter Arduinoen styrer en steppermotor på hver dyse for at kontrollere åbning og lukning.

Afgrænsning af produktet

For at sikre en struktureret og præcis tilgang til problemet skal produktet afgrænses inden det videre arbejde påbegyndes. Derfor kortlægges det fra starten, hvilke ingredienser der skal være i drinksmaskinen, og derfra hvilke muligheder slutbrugeren skal have. Det skal tilføjes, at maskinen vil kunne ændre sine drinksmuligheder baseret på de flasker, der puttes i maskinen. Dette kræver dog, at den senere kode ændres, og derfor fastlægges ingredienslisten, der gælder for resten af dette projekt fra starten. De ønskede ingredienser består af 5 flasker spiritus og 3 flasker opblanding i form af sodavand, hvilket tilsammen giver maskinen 8 forskellige drinkvarianter at vælge imellem. Dette ses på figur 8.



Figur 8: Oversigt over mulige drinks baseret på valgte ingredienser til drinksmaskinen

Krav til endelig løsning

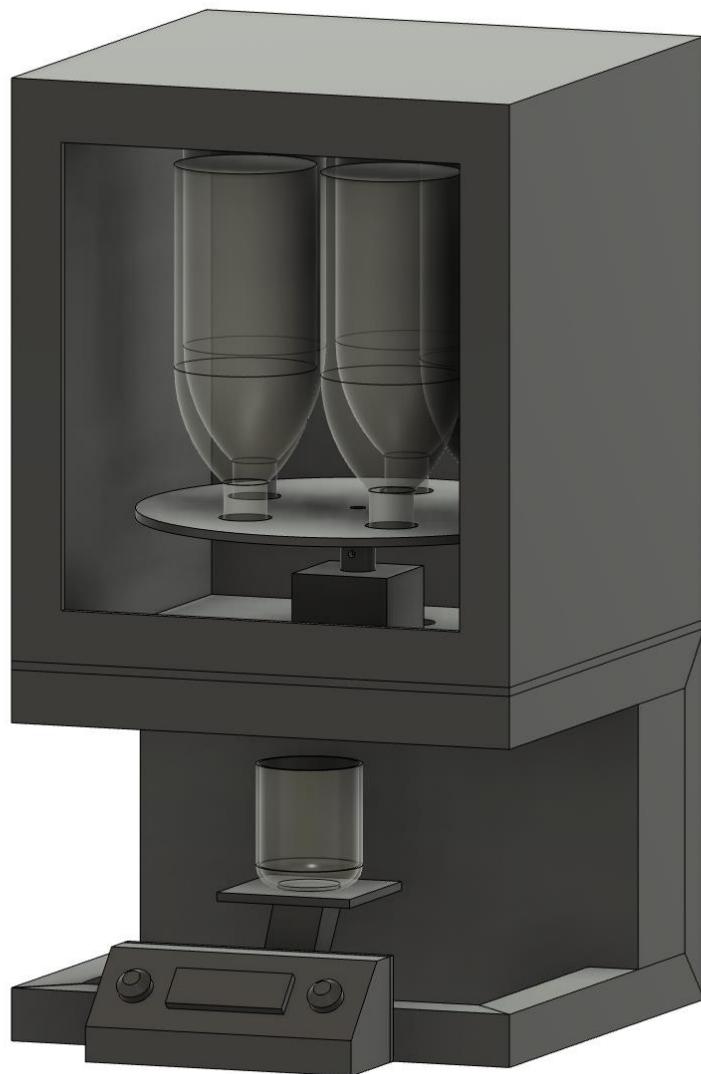
Inden den endelige udarbejdelse af produktets konstruktion herunder arbejdstegninger, elektriske diagrammer mm. opstilles der krav, se tabel 3, for at afgrænse arbejdet, så det er nemmere at arbejde direkte mod et fælles mål. Ydermere end at give input til den videre produktudvikling giver denne runde af krav også mulighed for at vurdere hvorvidt det færdige produkt så også lever op til de følgende krav.

Tabel 3: Krav til det endelige produkt

Krav	Relevans	Kravet er overholdt hvis...
Tiden fra drinkbestilling til servering må maksimalt være 30 sekunder	7/10	Maskinen maksimalt tager 30 sekunder om at serverer en drinksbestilling ved test.
Brugergrænsefladen skal være intuitiv og let at betjene for personer over 18 år	8/10	Brugergrænsefladen kan anvendes let og intuitivt af en testgruppe af personer over 18 år ved test
Løsningen skal kunne håndtere minimum 50 bestillinger i træk uden driftsstopp	9/10	Hvis løsningen kan håndtere 50 bestillinger i træk uden driftsstopp ved test
Støjniveauet må ikke overstige 70 dB målt 1 meter fra løsningen	7/10	Støjniveauet ikke overstiger 70 dB målt fra 1 meter ved test
Strømforbruget må maksimalt være 500 W i normal driftstilstand	6/10	Løsningen ikke bruger mere end 500W i normal driftstilstand ved test

Produktets virkemåde

Følgende afsnit uddyber produktets funktioner i detaljer med henblik på at give en omfattende forståelse af både den elektriske og mekaniske del af produktet. Figur 9, fra underafsnittet 'Teknisk afsnit for den mekaniske del', præsenterer en samlingstegning baseret på CAD-tegningen af produktet. Dette vises forud for den skriftlige beskrivelse for at lette forståelsen af produktets struktur ved at illustrere sammensætningen af konstruktionen.



Figur 9: Samlingstegning af CAD-konstruktionen hentet fra rapportafsnittet 'Den mekaniske del'

Teknisk afsnit for den mekaniske del

Materialevalg

I forbindelse med produktets konstruktion er det relevant at vurdere hvilke materialer der på bedste vis opfylder de krav der stilles til produktet. Derfor opstilles der krav til hvad materialet skal kunne, se tabel 4. Kravene der opstilles stilles med udgangspunkt i hvordan det endelige produkt, sat i produktion og klar til at blive brugt af slutbruger skal være konstrueret i. Det betyder samtidigt, at produktet der produceres på baggrund af denne rapport konstrueres i de tilgængelige materialer i værkstedet, og der dermed ikke tages højde for fødevaregodkendelse andre steder end skriftligt i denne rapport.

Krav til materialet

Tabel 4: Tabel over materialekrav

Krav	Relevans	Kravet er overholdt hvis...
Materialet skal være stærkt nok til at bære konstruktionen	10/10	Materialet kan bære konstruktionen når den er færdig bygget
Materialet som berører fødevare skal være godkendt hertil	10/10	Alle dele der berører fødevare er godkendt hertil
Materialet må gerne tilføre æstetisk appell til produktet	6/10	En testgruppe bestående af en potentiel målgruppe finder produktet æstetisk tilfredsstillende
Materialet skal være slidstærkt	8/10	Materialet er slidstærkt og ikke går i stykker

På baggrund af de opstillede krav kan det konkluderes at produktets konstruktion må opdeles i to, således det anvendte materiale giver mening i forhold til dets funktion. Konstruktionen opdeles således, så det omhandler materiale i kontakt med fødevare og materialer der ikke er i kontakt med fødevare. Det vælges af økonomiske årsager ikke at lave hele konstruktionen i fødevaregodkendte materialer, men i stedet kun de kritiske punkter i direkte kontakt med fødevaren, da dette fødevaregodkendte materiale, er betydeligt dyrere end det alment anvendte metal.

Materialeovervejelser til stelkonstruktionen

Materialer der anvendes til stellet og dermed ikke kommer i direkte kontakt med fødevare skal derfor kunne leve op til kravene om at tilføre æstetisk appell til produktet, være slidstærkt og stærkt nok til at bære konstruktionen. Det vurderes at den billigste og nemmeste løsning er at producere så meget som muligt af produktet i plastik. Det vigtigste af disse krav er dog, at produktet er stærkt nok til at bære konstruktionen. Derfor vælges det at lave steldelen i aluminium, for at sikre at den har den nødvendige bæreevne. Foruden stellet, vil enkelte aluminiumsdele bruges i stedet for plast, ikke grundet men dets funktionelle fordele, men deres æstetik. Dette sikre kravet og æstetisk appell til forbrugerne, mens at det vil være smart at udskifte de dele der oplever meget slid fra plast til aluminium for at forlænge levetiden.

Materialeovervejelser til fødevarekontakt

Da der er opstillet krav til at alle materialer som kommer i kontakt med fødevare, skal være godkendte hertil. I produktet vil de eneste materialer der kommer i direkte kontakt med fødevare i form af diverse væsker være beholderen hvori væsken opbevares, samt magnetventilen som leder vandet videre fra beholderen. Derfor kræver det altså kun, at virksomheden der skal producerer produktet er opmærksomme på at disse er godkendt.

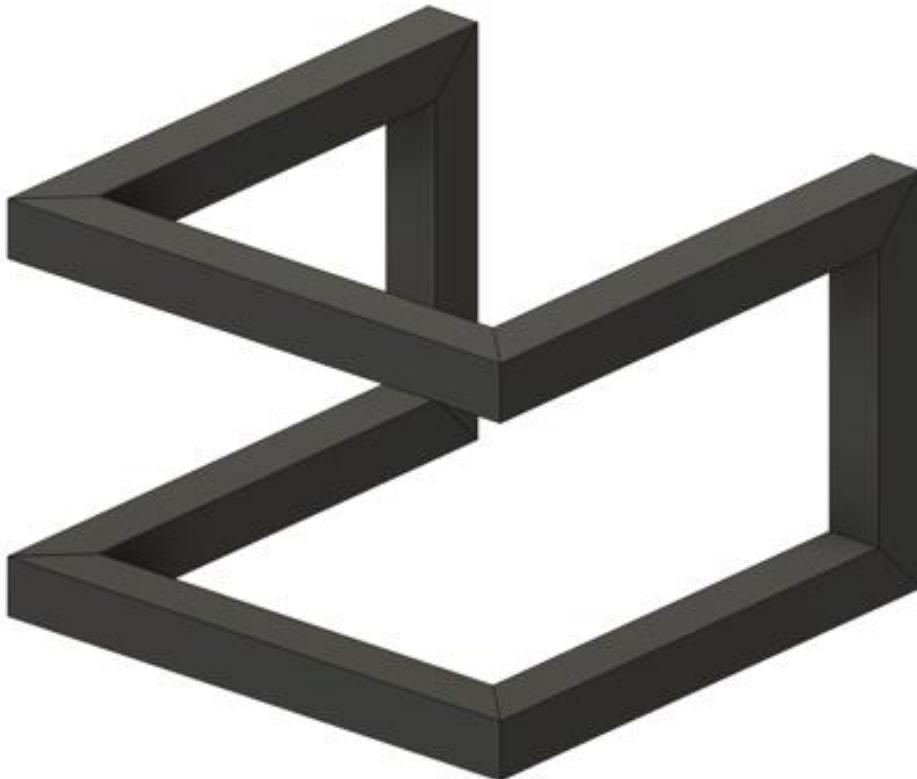
Produktets dele i metal

Produktet opdeles nedenstående i enkeltdele herunder metaldelene bestående af stel, kopholder, den roterende del og enclosuren til den roterende del. Til hver af disse dele laves arbejdstegninger i CAD-programmet Fusion 360, som opfylder kravene i forhold til dansk standard i forbindelse med arbejdstegninger. Der laves beregninger til relevante dele, samt produceres et flowdiagram over fremstillingsprocessen som slutteligt visualiseres med billede og beskrivelse. I flowdiagrammerne nævnes, at delen laves i processeres i korrekte mål, og her menes der efter de mål arbejdstegningen anviser. I den skriftlige beskrivelse af fremstillingsprocessen henvises der til et samlet afsnit om de anvendte maskiner benyttet i forbindelse med konstruktionen af produktet. Punktet '*anvendte maskiner*' placeret til slut i afsnittet '*Teknisk afsnit for den mekaniske del*' forklarer den relevante teori og sikkerhedsmæssige hensyn, som der skal tages højde for i forbindelse med anvendelsen af disse maskiner.

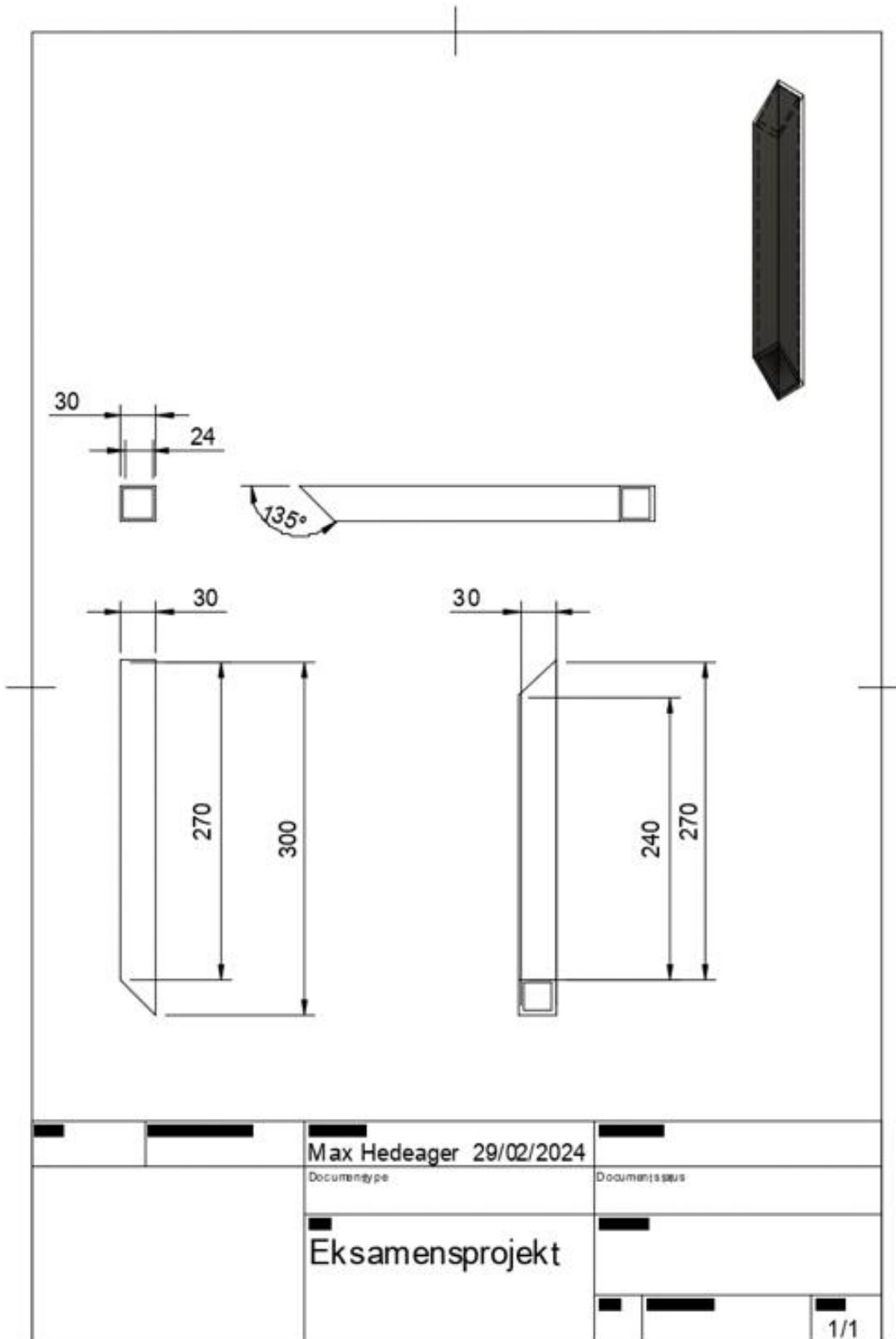
Stel

Arbejdstegninger

Stellet opbygges af 8 forskellige steldele, som alle skæres i smig og svejses sammen som vist på figur 10. Der skal laves 4 forskellige arbejdstegninger til stellet idet de enkelte steldele kommer i par af to ens. Der vises ét eksempel på et stelpar i rapporten, mens de resterende 3 er placeret i bilag, se bilag 1-3.

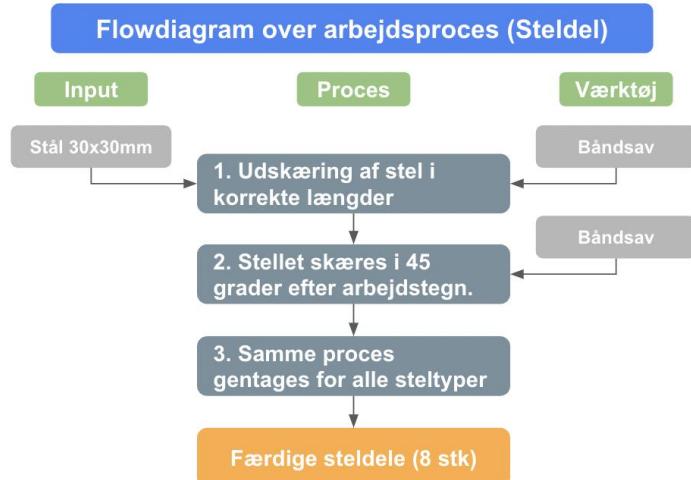


Figur 10: Visualisation af stellets udformning



Figur 11: Arbejdstegning over stelde type 1

Flowdiagram



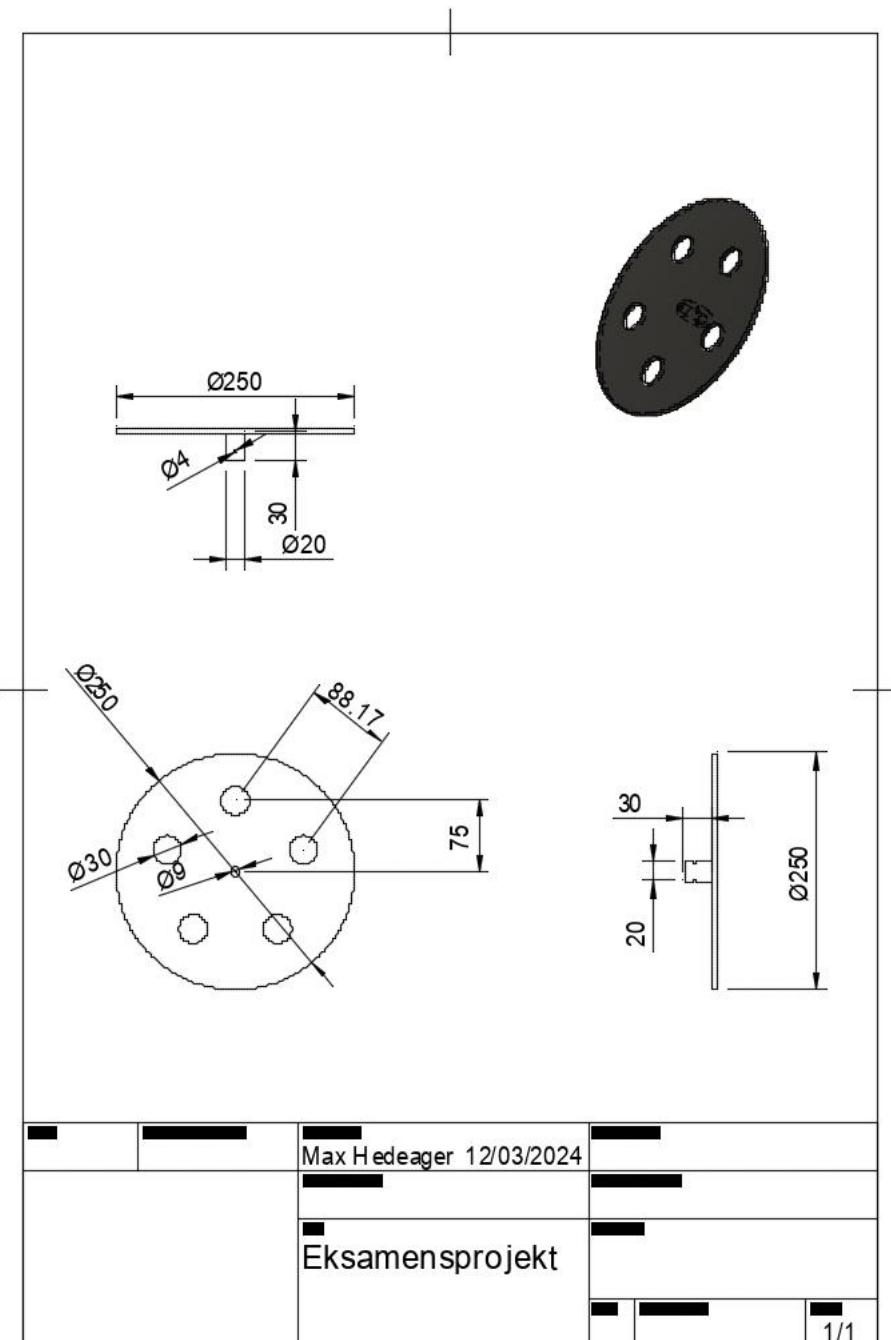
Figur 12: Flowdiagram over fremstilling af steldele

Flowdiagrammet viser hvordan steldelene fremstilles, de færdige dele i den orange boks føres med videre i flowdiagrammet over samlingen af hele produktet, se punktet 'flowdiagram over samlingen af produktets enkelt dele'.

Roterende del

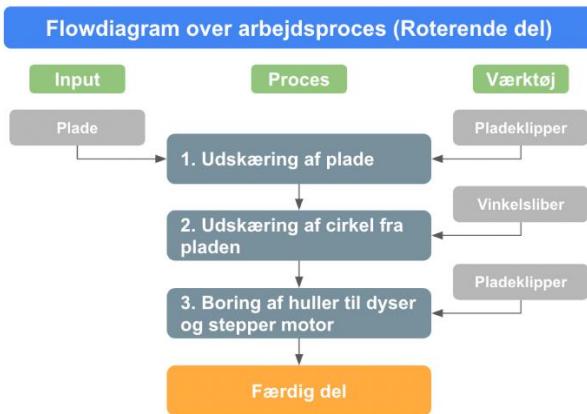
Arbejdstegninger

Der konstrueres en roterende del som monteres på stepper motoren, som samtidigt fastholder de 5 flasker med spiritus. Denne roterende del laves på en cirkulær plade med et hul i midten, hvori stepper motoren placeres. Rundt om dette centrum placeres et cirkulært mønstre af huller, hvori dysterne til hver af de 5 flasker skal placeres.



Figur 13: Arbejdstegning til den roterende del

Flowdiagram



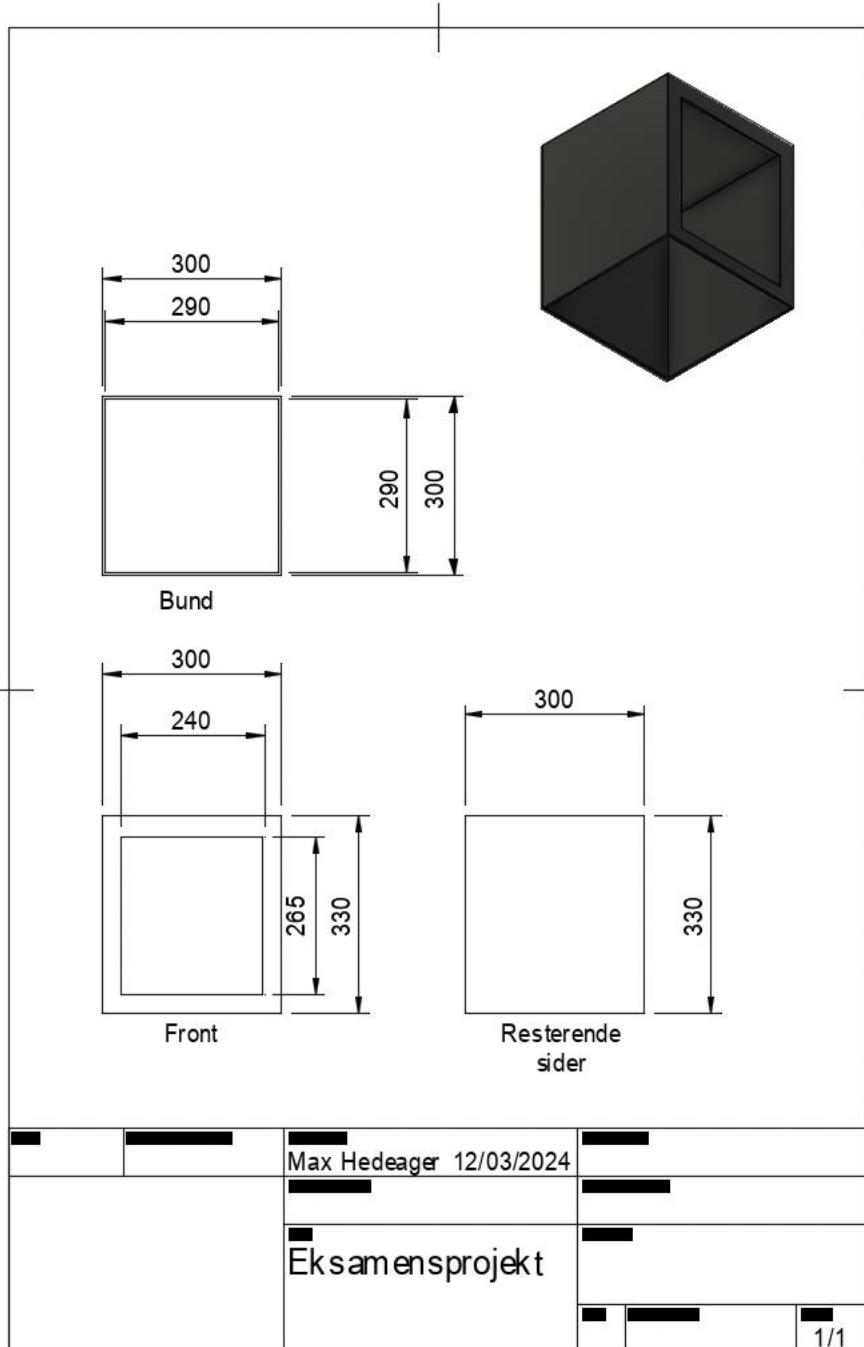
Figur 14: Flowdiagram over fremstilling af den roterende del

Flowdiagrammet viser hvordan den roterende del fremstilles, den færdige del i den orange boks føres med videre i flowdiagrammet over samlingen af hele produktet, se punktet 'flowdiagram over samlingen af produktets enkelt dele'.

Enclosure

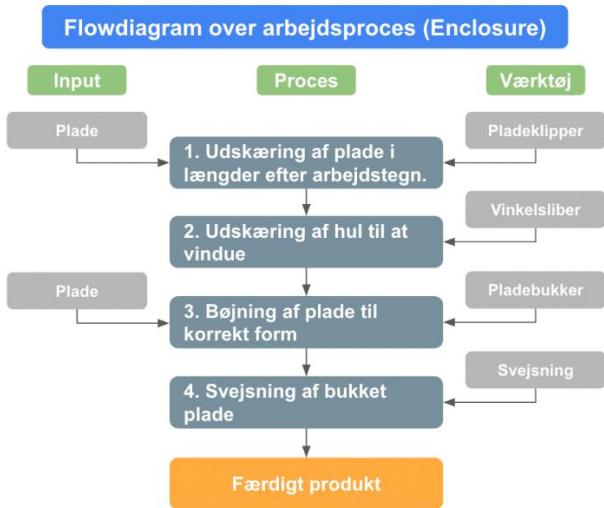
Arbejdstegninger

Der konstrueres en enclosure til den roterende del, for at skærme den for brugeren. Selve enclosuren fastmonteres ikke, da den skal være bekvemmelig at afmonterer for at servicer maskinen i form af opfyldning mm. I fronten udskæres et vindue, der gør det muligt for brugeren at se maskinen bevæge sig imens den skænker drinken.



Figur 15: Arbejdstegning til enclosuren

Flowdiagram



Figur 16: Flowdiagram over fremstilling af enclosure

Flowdiagrammet viser hvordan enclosuren fremstilles, den færdige del i den orange boks føres med videre i flowdiagrammet over samlingen af hele produktet, se punktet 'flowdiagram over samlingen af produktets enkelt dele'.

Produktets dele i plast

Det vælges at fremstille enkelte dele i plast, grundet de tidligere opstillede krav til produktet. Her er et af de opstillede krav til det endelige produkt nemlig at: "Brugergrænsefladen skal være intuitiv og let at betjene for personer over 18 år". For at lave en intuitiv brugergrænseflade som er let at betjene, skal den være simpel. Derfor findes en løsning der kombinerer brugen af analoge elektroniske komponenter med et digitalt display. På denne måde er det muligt at have en brugergrænseflade bestående af to knapper og et display, se figur 17. Displayet placeret centreret på huset der indeholder fumlebræt og den tilhørende elektronik har på hver side én knap. Begge knapper er ens for at danne symmetri, men har forskellige anvendelser. Knappen til venstre for displayet fungerer ved at dreje, hvor man på denne måde kan scroll gennem de mulige drinksvarianter. Knappen til højre for displayet fungerer ved at trykke ind på den, hvor dens funktion er, at bekære, at det er den drink som er vist på displayet der skal udskænkes.

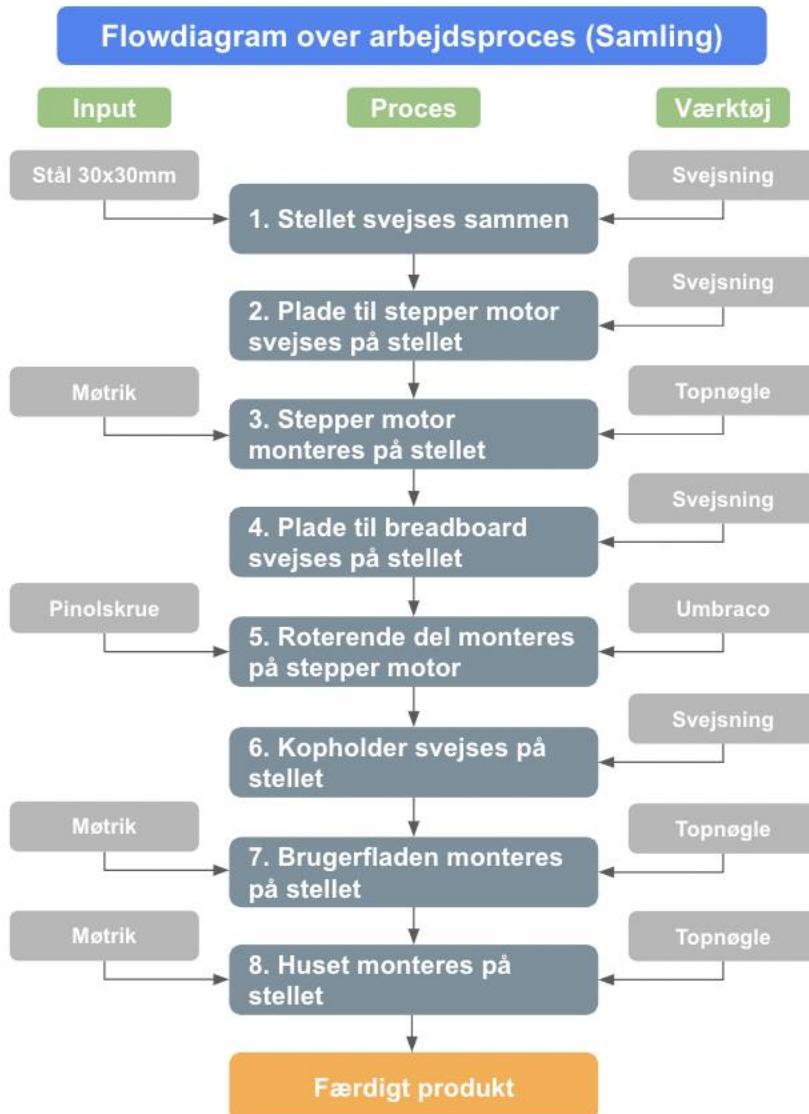


Figur 17: Visualisering af brugergrænsefladen

Netop denne del vælges at printes for at kunne montere de elektroniske komponenter i form af potentiometer og knappen under hver af knapperne, samt displayet imellem dem. Ydermere giver det mening rent æstetisk at printe netop denne del for at give slutbrugeren den bedst mulige oplevelse med produktet.

Flowdiagram over samlingen af produktets enkeltdele

Nedenstående figur illustrerer gennem et flowdiagram samlingsprocessen for produktet, hvor alle produktets enkelt dele er allerede er konstrueret hver for sig, se figur 12+14+16. Dette flowdiagram anvendes, hvs produktet skal bygges igen af nogle som ikke er blandt eksamensgruppen.



Figur 18: Flowdiagram over samlingsproceduren for produktets enkeltdele

Stykliste

Der opstillet en stykliste for at danne overblik over mængden af anvendt materiale i forbindelse med konstruktionen af produktet. Foruden dette bruges styklisten også i afsnittet om budget, hvori der beregnes en forventet produktionspris på produktet. I tabel 5, tages der ikke højde for 3D printede dele.

Tabel 5: Stykliste over materialer i maskin

Stykliste		
Beskrivelse	Dimension	Antal
Plade		
Stålplade bund	300x300mm	1
Stålplade til DC-motor	300x300mm	1
Stålplade til enclosure (Front)	300x900mm	1
Stålplade til enclosure (Øvre)	300x600mm	1
Stålplade til at skjule ledninger	1725x520mm	1
Stålplade til bunden	290x290mm	1
I alt: 0.805 m ²		
Profilrør		
Profilrør til steltype 1	30x300mm	2
Profilrør til steltype 2	30x300mm	2
Profilrør til steltype 3	30x300mm	2
Profilrør til steltype 4	30x210mm	2
I alt: 2.22 m		
Møtrikker og tilbehør		
Møtrik	M6	4
Bolt m. gevind	M6x60mm	4
Møtrik	M4	2
Bolt m. gevind	M4x40mm	2
Pneumatik modul		1
Elektronisk luftjustering		1
Slanger til pneumatik	1000mm	1

Anvendte maskiner

General sikkerhed i forbindelse med værkstedsarbejdet

Sikkerhed er særligt vigtigt ved arbejde i et metalværksted, hvor anvendelsen af værktøjer som pladeklippere, svejsemaskiner, søjleboremaskiner og båndsave kan udgøre potentielle risici for arbejderne. Ved korrekt håndtering og forståelse af sikkerhedsprocedurer kan risikoen for skader minimeres, samtidig med at effektiviteten og kvaliteten af arbejdet oprettholdes. Først og fremmest er det essentielt at bære passende beskyttelsesudstyr såsom sikkerhedshjelme, beskyttelsesbriller, høreværn, handsker og arbejdstøj, der kan beskytte mod mulige skader fra metaludskæringer, gnister, støj og andre potentielle farer. Dette udstyr bør bæres konsekvent under hele arbejdsprocessen. Desuden er det vigtigt at have kendskab til og overholde de specifikke sikkerhedsanvisninger og -procedurer, der er fastlagt for hvert værktøj. Dette inkluderer korrekt træning i brugen af værktøjerne samt forståelse for deres begrænsninger og potentielle farer.

Pladeklipper

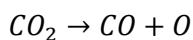
Pladeklipperen anvendes til præcis og effektiv klipning af metalplader i forskellige former og størrelser. Princippet bag anvendelsen af en pladeklipper er ret ligetil, men det kræver omhyggelighed og opmærksomhed for at opnå præcise resultater og undgå skader. Metalpladen placeres på pladeklipperens arbejdsflade, og det ønskede klipområde markeres med et mærkeværktøj. Når pladeklipperen er korrekt indstillet, kan operatøren aktivere værktøjet ved hjælp af kontaktknappen placeret ved fødderne. Det er vigtigt at anvende jævnt tryk og bevægelse under klipningen for at opnå en glat og præcis kant. For tykkere eller hårdere metalplader kan det være nødvendigt at gentage klipningen flere gange eller foretage flere passeringer med pladeklipperen. Efter klipningen skal operatøren deaktivere pladeklipperen og fjerne det færdige arbejdsstykke forsigtigt fra værktøjet. Princippet bag anvendelsen af en pladeklipper er baseret på skærekraften, der genereres af værktøjets skarpe kniv, som skærer gennem metalpladen ved hjælp af en skærekant. Ved korrekt indstilling og håndtering kan en pladeklipper opnå nøjagtige og rene klipninger, hvilket gør den til et uundværligt værktøj i metalværkstedet for fremstilling af præcise metaldele og komponenter.



Figur 19: Billede af pladeklipper

Svejsning

Svejsning er en metode til at forbinde to stykker metal, uden at skulle bruge møtrikker eller lign. De tilgængelige svejsere i værkstedet er CO₂ svejsere, som er nemmere at anvende for begyndere. Denne svejsetype kaldes MAG-svejsning, som dækker over at der svejses i en atmosfære af reagerende gasser. Det betyder gassen som anvendes, der oftest vil være en beskyttelsesgas nedbrydes i forbindelse med at beskytte lysbuen. I dette tilfælde, hvor værkstedet CO₂ svejsere er andet bliver CO₂'en nedbrydt til CO og O, hvilket også er illustreret ved nedenstående reaktionsskema og figur:



Selve samlingen af de to materialer ved hjælp af MAG-svejsning sker ved at varme materialer så meget op, så de ender med at smelte sammen til en. Med MAG-svejsning kan man svejse i aluminium, stål, rustfrit stål, kobber og enkelte kobber legeringer. Svejseren i metalværkstedet benytter sig af kortbuet materialeovergang, hvilket betyder at den bruger lave strømme sammenlignet med andre svejseteknikker. Dette er praktisk da det muliggør, at der nemmere kan svejses i små godstykkelser.



Figur 20: Billede af svejsestation

Båndsav

Metalbåndsaven bruges til at afkorte og skære i præcise længder, særligt i fladjern, rør, profilrør, cylindre af både stål og plast. Saven er en af de mest anvendte maskiner, idet der oftest bliver købt materialer hjem i store størrelser og derfor skal skæres i brugbare mål.



Figur 21: Billede af båndsav

Søjleboremaskine

Søjleboremaskinen anvendes til at bore huller alle dele af vores konstruktion. Disse huller bruges både til møtrikker, bolte og andet, samt til at føre ledninger rundt gennem produktet. Når man anvender søjleboremaskinen, er det vigtigt at følge de sikkerhedsmæssige anvisninger.

For at sikre et godt resultat ved boring, kan denne guide anvendes for at sikre den rigtige spåns tages i forhold til det bor der anvendes. Her er det muligt at se hvilken omdrejningshastighed der skal anvendes for at sikre denne korrekte spåns.



Figur 22: Billede af søjleboremaskine

Borehastighed/gevind tabel			Levet af Jens-Urik (3.U 2019)
\varnothing [mm]	Omdrejninger	Gevind [M]	\varnothing -bor [mm]
1	7320	2	1,6
2	3660	3	2,5
3	2440	4	3,3
4	1830	5	4,2
5	1464	6	5,0
6	1220	8	6,8
7	1046	10	8,5
8	915	12	10,2
9	813		
10	732		
11	665		
12	610		
13	563		
14	523		
15	488		
20	366		
25	293		
30	244		
35	209		
40	183		

$n = \frac{\pi \cdot 1000}{\varnothing \cdot d}$
 $v_{stål} = 23$
 $d = \text{Diameter bor eller diameter objekt}$
 $n = \text{Omdrejninger}$

Figur 23: Tabel med oversigt over borehastigheder

Pladebukker

Pladebukkeren bruges til at bukke stykker af plade i forskellige grader. Anvendelse af pladebukkeren sker manuelt, hvilket betyder at man selv skal være opmærksom på den vinkel som man får bukket pladen til. Dette gøres ved at pladestykket spændes fast i maskinen, og at man med hænderne bukker pladen op til den ønskede vinkel



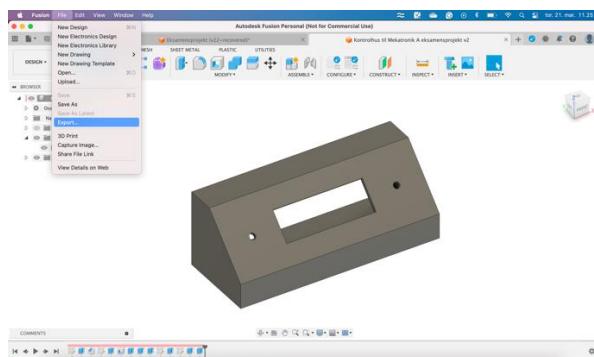
Figur 24: Billede af pladebukker

Redegørelse for 3D-printede modeller

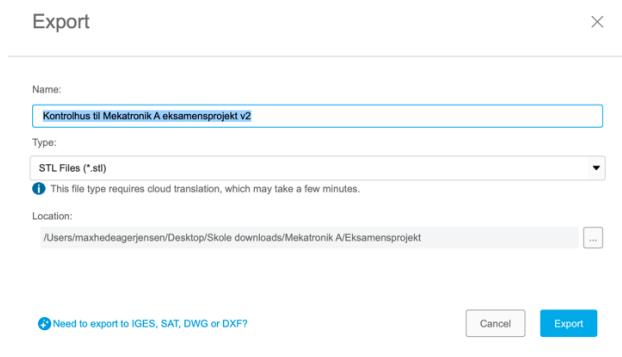
I det følgende afsnit gennemgås proceduren for 3D printning af de enkelte plastikdele der anvendes i produktet, herunder 1/2" møtrik i 5mm tykkelse, sodavandsflaske til 1/2" adapter og kontrolhuset til styring af systemet. Denne redegørelse tager udgangspunkt i printningen af vores kontrolhus, bruges til at huse vores elektronik og brugerens første direkte berøringsflade med produktet. Proceduren opdeles i tre dele – Tegning, generering af g-code og udprint, som beskrives yderligere herunder.

3D-tegning til STL-fil

Efter at have tegnet huset i fusion 360, kan denne eksporteres. Her er det muligt at vælge flere forskellige fil-typer med hver deres fordel. Til 3D print vælges fil-typen STL. Denne måde at eksporterer er vist nedenstående i figur 25 + 26.



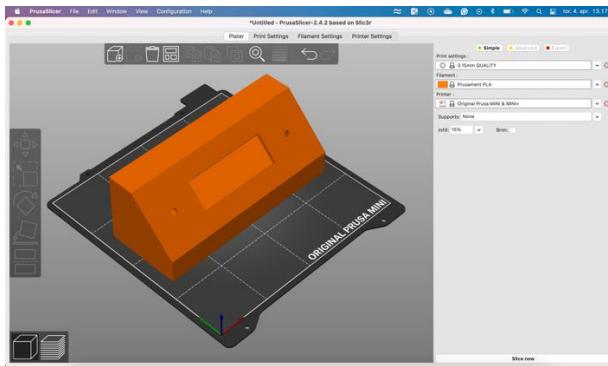
Figur 25: Eksportering af tegning



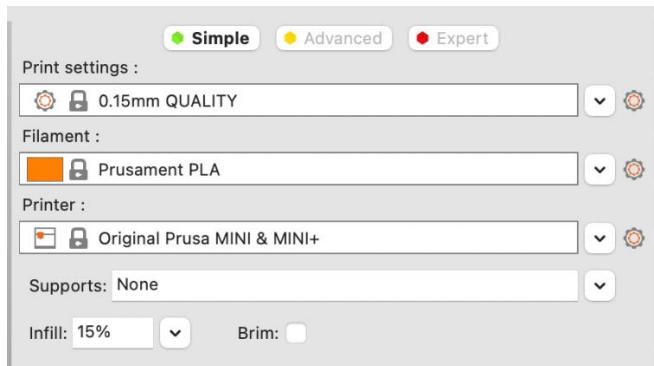
Figur 26: Valg af fil-type (STL)

Generering af g-code

For at printeren ved hvilken "rute" den skal køre, kræver det at man har en såkaldt g-code. Denne g-code genereres i et slicer-program som passer til den printer man har. Da skolen stiller Prusa mini printere til rådighed for vores print laves g-coden i prusa's egen slicer. Her importerer man sin STL-fil, se figur 27, og kan herefter justerer de forskellige printer indstillinger, se figur 28, så de bedst muligt passer til det specifikke print man ønsker printet. De printede dele til produktet har ikke særlige krav i forhold til styrke, som oftest styres med en infill mængde. I prusa sliceren er 15% infill standard, og har i dette produkt tilfælde vist sig at være rigeligt i forhold til de belastninger som delene udsættes for. Det har dog for flere af delene været nødvendigt printe med support, som betyder at printeren printer understøtninger de steder hvor der ellers skulle være overhæng, da den ikke kan printe ud i den frie luft. Dette support-materiale sidder kun minimalt fast til det reelle print og er derfor nemt at fjerne efterfølgende. Når man er tilfreds med indstillingerne til g-code, eksporterer denne til et USB-stik som passer til printeren.



Figur 27: Importeret STL-fil i prusa slicer



Figur 28: Menu til printerindstillinger

Udprint

Efter at have overført sin g-code til et USB-stik indsættes denne i 3D printeren som for skolen er en Original Prusa mini. Denne printer er god til skole anvendelse, da der er minimale ting man skal gøre for at få et godt print, idet den selv auto-leveler printbed'en, hvilket vil sige at man gør printfladen helt vandret. Udeover dette registrerer den også selv den seneste g-code fil som er tilføjet til USB-stikket, og det eneste man herefter skal gøre er at trykke print. Vil man gerne være helt sikker på et godt print er det en god idé at rense printfladen med sprit for at plastikken bedre klæber til overfladen.

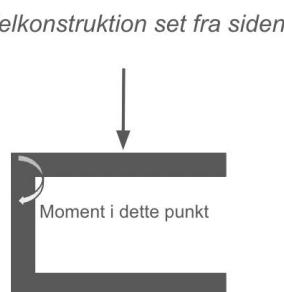
Konstruktionsberegninger

Det følgende afsnit laver beregninger på konstruktionen, hvor beregningerne tager udgangspunkt i stellet. Her beregnes nedbøjning og a-mål for at sikre dimensioneringen kan holde til lasten.

Beregning af nedbøjning

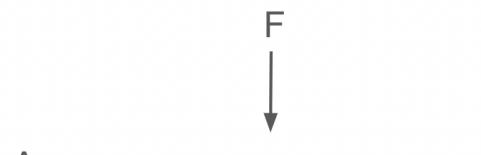
Når nedbøjningen af stellet beregnes, tages der udgangspunkt i at massen ovenpå stellet er 10 kg. Det betyder at kræften der påvirkes med er 98.2N idet den ganges med tyngdeaccelerationen. Ydermere antages det at kræften der påvirkes med, er placeret på midten af bjælken (150mm ude fra ankerplacering), hvilket også er det svageste punkt i forhold til nedbøjning. Dette har betydning for L i den senere formel. Nedenstående ses stelkonstruktionen fra siden, se figur 29, hvor der er indtegnet det moment der opstår i stellets svejsning. Figuren sidestående, se figur 30, illustrerer beregningsmodellen for nedbøjningen i den øverste bjælke i stellet

Visualisering af konstruktion



Figur 29: Visualisering af konstruktion

Beregningsmodel



Figur 30: Beregningsmodel for nedbøjning

Med udgangspunkt i den ovenstående beregningsmodel beregnes nedbøjningen ved brug af formlen for nedbøjning, se figur 31. I formlen for nedbøjning er E er materialets elasticitetsmodul, I er inertimomentet for RHS-profil, F er kræften der påvirkes med og L er afstanden fra ankerpunktet til der hvor kraften påvirker med.

$$u_{max} = \frac{F \cdot L^3}{3 \cdot E \cdot I}$$

Denne beregnes som vist nedenstående:

$$I := 0.0272 \cdot 10^6 \text{ mm}^4 \quad E := 0.21 \cdot 10^6 \text{ MPa} \quad F := 98.2 \text{ N} \quad L := 150 \text{ mm}$$

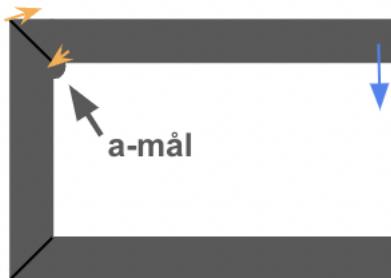
$$u_{max} := \frac{F \cdot L^3}{3 \cdot E \cdot I} = 0.019 \text{ mm}$$

På baggrund af denne udregning kan det konkluderes at nedbøjningen for den øvre bjælke i stellet er 0.019mm, og at dette altså ikke vil have nogen betydning for vores konstruktion.

Beregning af a-mål

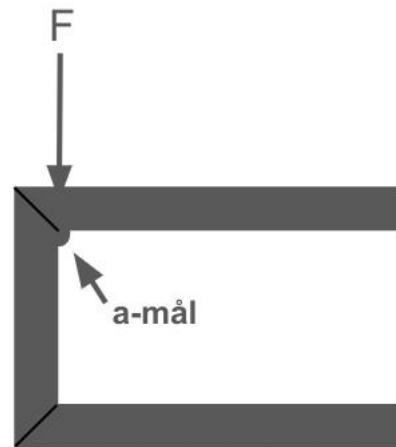
Dimensionering af svejsninger til konstruktionen beregnes med udgangspunkt i svejsningen øverst i stellet. Det antages ligesom i *beregning af nedbøjning* at massen af det stellet skal bære er 10 kg. Dette betyder at kræften der påvirkes med er 98,2 N, idet massen ganges med tyngdeaccelerationen. Nedenstående figur viser visuelt hvordan svejsningen påvirkes, vist med orange pile, når bjælken presses i retningen vist med blå pil, se figur 32. Sidestående er modellen for beregning af trykkraften på den indvendige svejsning, som er indtegnet og markeret, se figur 33.

Visualisering af træk/tryk



Figur 32: Visualisering af træk/tryk i stelkonstruktionen

Stelkonstruktion set fra siden



Figur 33: Visualisering af beregning

$$\sigma = c \cdot \frac{F}{a \cdot L}$$

Figur 34: Formel for beregning af a-mål

For at beregne dette a-mål anvendes formlen til beregning af a-mål, se figur 34, hvor σ er trykspændingen, c er sømfaktoren, F er kraftpåførslen i [N], a er a-målet og L er længden af svejsningen:

Da trykspændingen σ og sømfaktoren c er konstanter, samt det faktum at vi kender længden af svejsningen på 30mm og at kræften der påvirkes med er 98,2 N kan a-målet beregnes som vist nedenstående, her er beregningen løst vha. CAS-værktøjet Mathcad:

$$\sigma := 125 \text{ N} \quad c := 1.4 \quad F := 98.2 \text{ N} \quad L := 30 \text{ mm}$$

$$\sigma = c \cdot \frac{F}{a \cdot L} \xrightarrow{\text{solve}, a} \frac{float, 3}{mm} 0.0367$$

Det beregnede a-mål i dette eksempel bliver 0.0367 mm, hvilket er meget småt, idet konstruktionen nærmest ikke skal bære noget i relation til dets dimension. Derfor bliver vi nødt til at følge reglen om at alle a-mål som minimum skal være 3 mm. Havde eksemplet taget udgangspunkt i en større last, ville a-målet også være større.

Bestemmelse af konstanter til beregningerne

Sømfaktor

Sømfaktoren bestemmes ud fra den følgende figur opgivet fra Learn i dokumentet om svejse beregninger.

Når der svejses i maskinværkstedet på vores skole arbejder vi efter en sømkasse B, og da rapportens eksempel tager udgangspunkt i et kantsøm er det ved hjælp af tabel til sømfaktor, se figur 35, muligt at konkludere at den sømfaktor der skal bruges i vores beregning af a-målet er 1,4.

Sømfaktor C

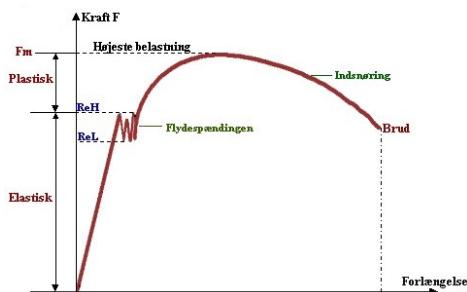
Sømtype	Belastnings type	Sømkasse		
		A	B	C
Stumpsøm	træk, bøjning tryk, forskydning	1,5	1,2	1,0
		1,2	1,0	1,0
Kantsøm	træk, bøjning tryk forskydning	1,8	1,5	1,4
		1,7	1,4	1,4
		1,2	1,0	1,0

Tabel 119

Figur 35: Figur over sømfaktoren

Trykspænding

Trykspændingen handler om hvor meget kraft man kan påvirke materialet med før det mister sin form. Dette er vist med nedenstående figur, som viser hvordan kraften påvirker forlængelsen. Det er muligt at se hvordan materialet i første omgang er elastisk før det rammer flydespændingspunktet og begynder at flyde. Herefter kan man igen strække materialet en smule før det begynder at indsnøre kort før den brækker. Dette er forskelligt fra materialet, men når vi arbejder med konstanten i mekatronik-faget fastsættes værdien til 125N.



Figur 36: Figur der viser trykspændingen

Elasticitetsmodul for stål

Denne konstant er fundet ved opslag i statik og styrkelærer bogen på side 231 og findes til værdien:

$$E = 0.21 * 10^6 \text{ MPa}$$

Inertimomentet for RHS-profil

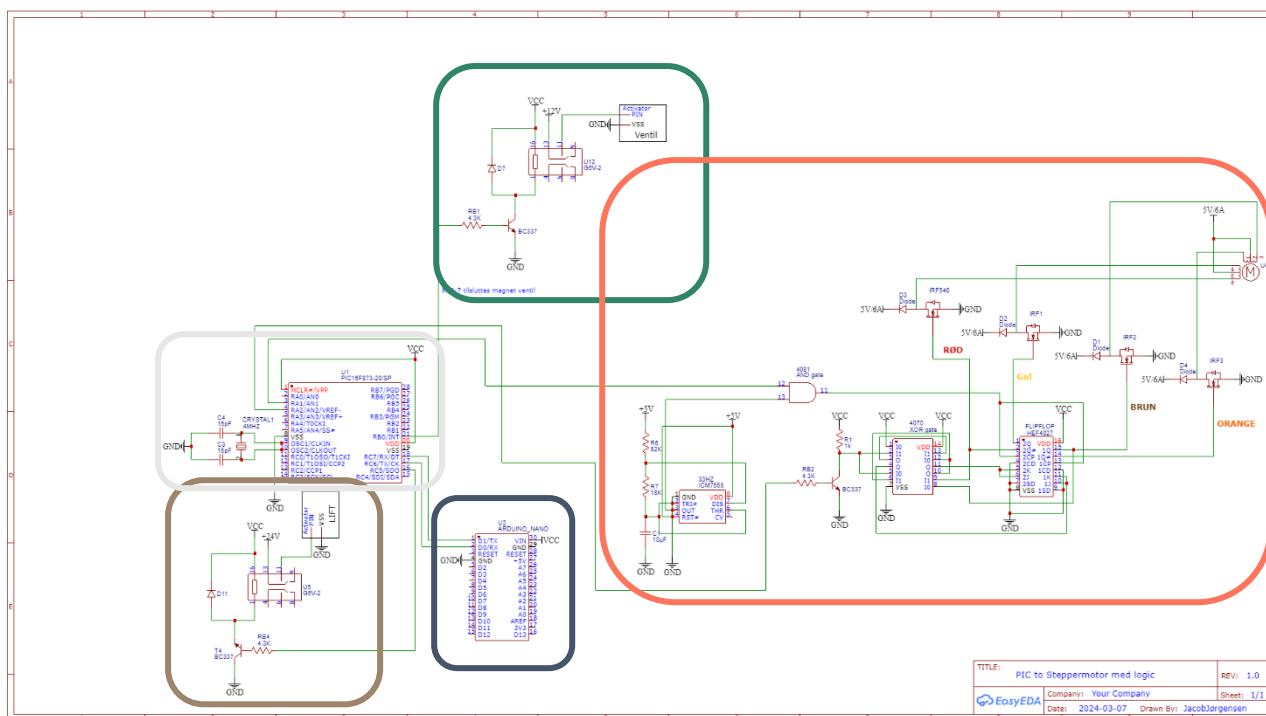
Denne konstant er fundet ved opslag i statik og styrkelærer bogen på side 266 og findes til værdien:

$$I = 0.0272 \cdot 10^6 \text{ mm}^4$$

Teknisk afsnit for den elektriske del

Den efterfølgende gennemgang af den elektroniske del er struktureret således, at hver enkelt del behandles særskilt. Dette gøres for at sikre en dybdegående forståelse af hver dels funktion og integration i det overordnede system. Indledningsvist vil styringenheden blive analyseret med henblik på at etablere en grundlæggende forståelse af kommunikationen mellem systemets forskellige elementer. Efterfølgende vil den overordnede kodestruktur, herunder hovedløkkken, blive belyst for at give et overordnet overblik. Dernæst vil en systematisk gennemgang af de enkelte kredsløbskomponenter finde sted. For hver komponent vil der blive redegjort for de anvendte komponenter, den tilhørende kode, samt en præsentation af de gennemførte tests.

Det samlede kredsløb består af følgende hovedkomponenter: Brugergrænseflade, **styring af roterende del**, **styring af magnetventiler** samt **styring af den pneumatiske del**. Alle disse komponenter er integreret og styret af en **PIC16F873-mikrocontroller**. Figur 37 inddeler el diagrammet i de 4 hovedkomponenter for at give yderligere overblik.



Figur 37: El diagram over hele kredsløbet - inddelt i hovedkomponenter. (Se bilag 21 for fuld størrelse uden inddeling)

Systemets overordnede funktionalitet er bygget op omkring en Arduino-styreenhed, der er forbundet som illustreret i Figur 37. Denne Arduino-enhed muliggør brugervalg af den ønskede drik, hvilket transmitteres via USART-serielkommunikation til en PIC16F873-mikrocontroller. Mikrocontrolleren bearbejder denne inputdata og integrerer på baggrund af dette med systemets resterende hovedkomponenter.

Når en drik skal dispenseres, aktiveres først et løfte System, der sørger for at løfte drikken tæt på den pågældende flaskes ventil. Dette styres af mikrocontrolleren via den pneumatiske del af systemet, hvor udsendelse af enten logisk 0 eller 1 på pin RC5 igangsætter den nødvendige funktion. For at sikre, at den korrekte drik står over dispenseringsåbningen, benytter mikrocontrolleren digitale signaler via pin RA1 og RA2 til at styre den roterende del. Her er RA1 aktivering af stepper motor og RA2 styrer retningen. Når den

rette ingrediens er placeret, udsender mikrocontrolleren et signal via en af PORTB-pinsne for at aktivere den relevante af de 8 magnetventiler, der dispenserer selve drikken.

Komponenter og værktøjer

Afsnittet fokuserer på at give overblik over komponenter, værktøjer og software som er blevet anvendt i udarbejdelsen af produktets el tekniske del. I Tabel 6 dannes overblik over anvendte komponenter.

Tabel 6: Stykliste over de elektroniske komponenter

Komponent	Antal
Mosfet - IRF540	4
Transistor - BC337	7
StepperMotor - NEMA-23	1
Timer - ICM7555	1
XOR gate - 4070	1
AND gate - 4081	1
FlipFlop - HEF4027	1
Relæ - G5V-2	6
Diode	10
Arduino UNO	1
Knap	1
Potentiometer - 10Kohm	1
LCDskærm	1
MagnetVentil	5
Kondensator 15 pF	2
Krystal oscilator	1

Herudover er følgende værktøjer benyttet:

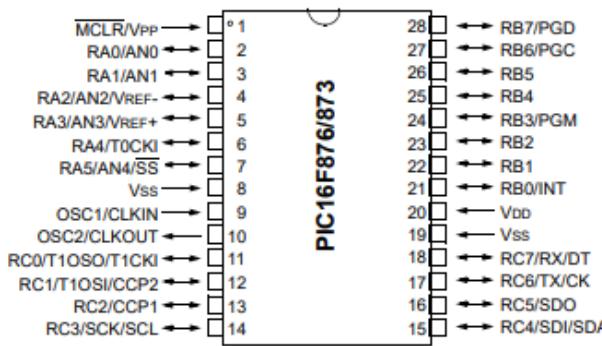
- Multimeter
- Automatisk afisoleringsværktøj
- Pincet
- Strømforsyning
- Mikrochip ICD 2 programmerings modul

Software benyttet er primært MPLAB X IDE til at skrive og simulere koden, dog er den ældre version: MPLAB V8.92 også benyttet da dette er den eneste version der stadig kan kommunikere med IDC 2 programmeringsmodulet.

Styringsenheden - PIC16F873

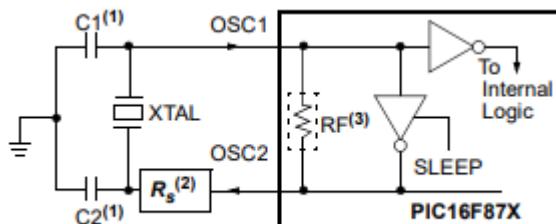
Centralt i det overordnede kredsløb er PIC16F873-mikrocontrolleren, der varetager den overordnede styring og koordinering af alle systemets komponenter. Valget af netop denne PIC-variant er velbeggrundet, idet den som illustreret i figur 38 besidder et væsentligt højere antal programmerbare I/O-ben (22) sammenlignet med alternativet PIC16F84A, der kun tilbyder 13 programmerbare ben. Ydermere muliggør PIC16F873-enheden seriellkommunikation, hvilket er essentielt for interfacet til Arduinoen.

PDIP, SOIC



Figur 38: Pinlayout for PIC16F873. (Datablad bilag 5)

Korrekt opsætning af denne komponent er essentiel, og det er derfor vigtigt at benytte dets datablad (bilag 5). Forkert opsætning vil resultere i, at koden ikke vil fungere efter hensigten. Af figur 38 kan det ses, hvor 5V (VDD) og jord (VSS) skal forbindes, samt hvor oscillatoren skal sidde. Opsætningen af oscillatoren kræver dog yderligere uddybning, som findes i databladet, se figur 39 og 40.



Figur 39: Opsætning af krystaloscillator (Datablad bilag 5)

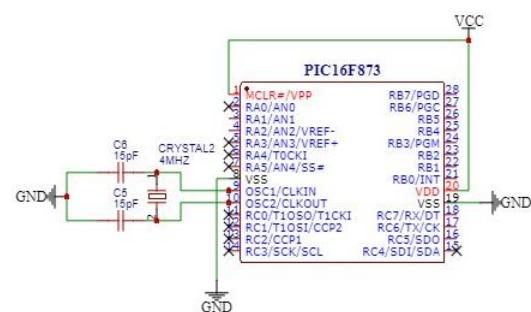
Af figur 39 fremgår det hvordan opsætningen af krystaloscillatoren og de tilhørende kondensatorer ser ud. Systemet opererer ved en klokfrekvens på 4 MHz, det kan derfor ses af figur 40 at der skal anvendes en 4 MHz krystaloscillator hvortil 2 kondensatorer med en værdi på 15pF skal anvendes. Endvidere beskriver figuren at oscillatortypen skal konfigureres til enten XT eller HS.

Derudover kan det fra databladet (bilag 5) i tabel 1-1 PINOUT DESCRIPTION ses at MCLR (Master Clear) er aktiv lav, denne skal derved sættes til VDD form at undgå at PIC'en konstant nulstiller. På baggrund af disse forudsætninger tilsluttes PIC'en som illustreret på Figur 41.

TABLE 12-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

Osc Type	Crystal Freq.	Cap. Range C1	Cap. Range C2
LP	32 kHz	33 pF	33 pF
	200 kHz	15 pF	15 pF
XT	200 kHz	47-68 pF	47-68 pF
	1 MHz	15 pF	15 pF
	4 MHz	15 pF	15 pF
	8 MHz	15-33 pF	15-33 pF
HS	4 MHz	15 pF	15 pF
	8 MHz	15-33 pF	15-33 pF
	20 MHz	15-33 pF	15-33 pF

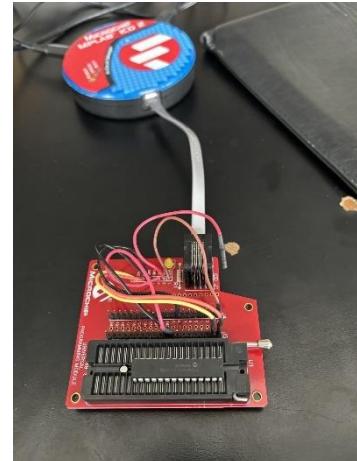
Figur 40: Krystaloscillator oversigt (Datablad bilag 5)



Fig

Programmering af PIC

For at brænde koden over på PIC16F873-mikrocontrolleren, er det nødvendigt at benytte en dedikeret programmer-enhed udviklet af Microchip, ved navn ICD 2. Denne programmeringsenhed besidder sin egen strømforsyning og tilsluttes computeren via et USB-kabel til overførsel af den ønskede programkode. Som det fremgår af figur 42, består programmeringsmodulet af 7 individuelle ledninger, der kræver korrekt forbindelse til PIC16F873-mikrocontrolleren for at sikre fejlfri overførsel af koden. Ved konsultation af komponentens datablad kan den præcise tilslutning af de enkelte signaler fastlægges, se figur 43.



Figur 42: Microchip ICD2

PIN DESCRIPTIONS (DURING PROGRAMMING): PIC16F87X

Pin Name	During Programming		
	Function	Pin Type	Pin Description
RB3	PGM	I	Low voltage ICSP programming input if LVP configuration bit equals 1
RB6	CLOCK	I	Clock input
RB7	DATA	I/O	Data input/output
MCLR	VTEST MODE	P*	Program Mode Select
VDD	VDD	P	Power Supply
VSS	Vss	P	Ground

Legend: I = Input, O = Output, P = Power

* In the PIC16F87X, the programming high voltage is internally generated. To activate the Programming mode, high voltage needs to be applied to the MCLR input. Since the MCLR is used for a level source, this means that MCLR does not draw any significant current.

Figur 43: PIN DESCRIPTIONS (DURING PROGRAMMING): PIC16F87X (PIC16F87X EEPROM Memory Programming Specification, 2002)

Specifikt skal VSS forbindes til VSS, VDD til VDD, mens MCLR-benet på PIC'en skal tilsluttes "Program Mode Select"-terminalen (VPP) på programmeringsmodulet. De resterende to ledninger, betegnet PGC (Clock) og PGD (Data) på modulet, skal forbindes til henholdsvis RB6 og RB7 på PIC16F873-mikrocontrolleren, som illustreret i figur 42.

Kodekonstruktionen

I det følgende afsnit vil programmets fulde koden gennemgås. Formålet er at give en indsigt i den overordnede sammensætning af programmets forskellige kodesegmenter samt en forklaring af de underliggende principper og koncepter, der er anvendt i assembly-sproget. Der vil blive redegjort for de forskellige instruktionssæt, der er anvendt i koden, samt hvordan initialisering og konfiguration af specifikke bitsæt udføres. Endvidere vil der blive givet en detaljeret beskrivelse af, hvordan programmets forskellige kodedele sammenarbejder og fungerer som en helhed. Afslutningsvis vil der blive givet en kort introduktion til debuggingsprocessen i MPLAB X IDE.

Assembly kode

I afsnittet vil initialisering af PIC'en, de forskellige instruktionssæt og implementeringen af delay funktionen forklaries.

Initialisering af PIC'en

I dette afsnit beskrives initialisering af koden. Først skal den specifikke type af PIC-mikrocontroller defineres linje 8 kodeudsnit 1, samt skal man inkludere den nødvendige .inc-fil for at kunne håndtere denne enhedstype korrekt ift. de forhåndsdefinerede funktioner.

```
8:      list      p=PIC16F873  
9:      include    p16f873.inc
```

Kodeudsnit 1: Definering af enhed og import af .inc fil

Opsætning af konfigurations bits

Konfigurations bitsne bruges til at definere mikrocontrollerens overordnede opførsel og funktionalitet. Disse bits er placeret i et særligt konfigurationsord, der ligger i programmeringshukommelsen på adressen 2007h, hvilket er uden for det tilgængelige programmeringsområde for brugerkoden. Konfigurationsordet kan derfor kun tilgås under den faktiske programmering af mikrocontrolleren. (Bilag 5, figur 98 og 9)

I den specifikke konfigureringsstreg, der anvendes i dette program, defineres en række vigtige systemparametre:

- CP1, CP0 (Code Protection): Disse bits bestemmer, hvilke områder af flashhukommelsen der skal beskyttes mod læsning eller kopiering. I den valgte konfiguration er kodehukommelsen ubeskyttet (CP1=0, CP0=0), hvilket tillader fuld adgang til at læse og kopiere programkoden.
- DEBUG (In-Circuit Debugger): Bit-værdien 0 (OFF) aktiverer in-circuit debuggerfunktionen, hvorved RB6 og RB7 pins tildeles til brug for debuggeren. Dette er nyttigt under udvikling og fejlretning af programmet. I koden er dette deaktiveret.
- LVP (Low Voltage Programming): En værdi på 0 (OFF) deaktiverer lav volt programmering via RB3/PGM-pinnen, hvilket gør RB3 til normal I/O port, hvortil høj volt programmering er på krævet.
- PWRTE (Power-up Timer): Værdien 1 (ON) deaktiverer power-up timeren, hvilket gør, at mikrocontrolleren starter øjeblikkeligt efter strømtilslutning.
- WDTE (Watchdog Timer): Værdien 0 (OFF) deaktiverer watchdog timeren, der ellers ville genstarte mikrocontrolleren periodisk.
- FOSC (Oscillator Selection): Sættes til HS (High Speeed) der muligør krystal/resonator-oscillator med en værdi på 4-20 MHz som klokkefrekvens for mikrocontrolleren.

(Configuration Word of PIC16F873, u.å.)

Variable definering

I programmet defineres en række variabler, der vil blive anvendt senere i koden, jf. kodeudsnit 2. Disse inkluderer:

- receivedData: En variabel til at lagre data modtaget via den serielle port, der kan bruges til at indekseringsformål.
- goToStart: En variabel til at holde styr på, hvor mange gange steppermotoren skal returnere til startpositionen.
- d1, d2, d3, d4: Forsinkelsesvariabler, der kan anvendes til at introducere tidsforskydninger eller pauser i programkørslen.

RAM-registerbanken i PIC16F873 mikrocontrolleren er inddelt i fire banker, hver med en størrelse på 128 bytes. De specielle funktionsregistre (SFR) er allokeret i de første 32 bytes af hver bank, jf. bilag . Efter byte 32 er der et registerområde på 96 bytes i hver bank, der frit kan bruges. Dog er bank 3 og 4 en spejling af bank 0 og 1, så det samlede brugbare RAM-område er på 192 bytes fordelt over 2 banker.

De ovennævnte variabler er placeret i det frie registerområde, startende fra byte 20h svarende til byte 32 til 25 svarende til byte 37 alle placeret i bank0.

13: receivedData	equ 20h	;Variable for index lookup table
14: goToStart	equ 21h	;variable for number of times needed to go to start
15: d1	equ 22h	;Delay variable
16: d2	equ 23h	;Delay variable
17: d3	equ 24h	;Delay variable
18: d4	equ 25h	;Delay variable

Kodeudsnit 2: Variable definering

Desuden defineres en række symbolske konstanter, der repræsenterer de enkelte portben, der er tilknyttet de forskellige magnetiske ventiler og steppermotorkontrolpins, jf. kodeudsnit 3. Dette giver en mere læsbar og vedligeholdelse venlig kode sammenlignet med at referere direkte til registre og bitpositioner.

24: DirectionPin	equ RA2 ; Pin 4 PORTA,2
25: GinValve	equ RB0 ; Pin 21
26: RomValve	equ RB1 ; Pin 22

Kodeudsnit 3: Konstant definering

Registerbankerne

I afsnittet gennemgås opsætningen af de anvendte registre, samt gives en forståelse af, hvordan man skifter imellem de forskellige banke.

Registerbankerne i PIC16F873 mikrocontrolleren muliggør styring af RAM-hukommelsen og de specielle funktionsregistre (SFR). I alt findes der fire registerbanke, nummereret fra 0 til 3, hvor hver bank indeholder 128 byte RAM. For at skifte mellem registerbankerne anvendes STATUS-registeret (adresse 03h, 83h, 103h, 183h). Dette register indeholder en række statusflags, herunder bit 5 (RP0) og bit 6 (RP1), der bruges til at vælge den aktive registerbane. (*SFR explanation for PIC16F873*, u.å.) Ved at manipulere disse to bits kan man skifte mellem de fire banker efter behov. Kodeudsnit 4 illustrerer koden der definerer de to makroer, `bank1` og `bank0`, der henholdsvis skifter til bank 1 og bank 0.

```

41: bank1 macro
42:     bsf STATUS, RP0
43:     endm
44: bank0 macro
45:     bcf STATUS, RP0
46:     endm

```

Kodeudsnit 4: Bank makroer

"bank1" makroen bruger instruktionen `bsf` (bit set file) til at sætte bit 5 (RP0) i STATUS-registeret til 1. Dette resulterer i, at bank 1 vælges som den aktive registerbank.

"bank0" makroen bruger instruktionen `bcf` (bit clear file) til at rydde bit 5 (RP0) i STATUS-registeret til 0. Dette resulterer i, at bank 0 vælges som den aktive registerbank.

Opsætning af relevante registre

I initialiseringfasen af programmet foretages konfigurationen af de essentielle specielle funktionsregistre (SFR) i mikrocontrolleren. Da flere af de relevante registre er placeret i registerbank 1, skiftes der indledningsvist til denne bank. Tre overordnede områder kræver initialisering: konfiguration af TRIS-registrene for styring af port-retninger, opsætning af ADCON1-registeret for analog-til-digital konvertering samt initialisering af den serielle USART-kommunikation.

TRIS-registre (TRISx)

Ifølge PIC16F873 dataarket, (jf. bilag 5), befinner konfigurationsregistrene TRISx sig i registerbank 1. I disse registre defineres retningen for de individuelle porte ved at skrive de ønskede værdier til de respektive TRIS-registre. Hvert enkelt bit i et TRIS-register korresponderer til den tilsvarende port-pin, hvor en logisk '1' angiver input-tilstand, og en logisk '0' angiver output-tilstand.

For at konfigurere alle pins på PORTA og PORTB som udgange indlæses en byte med den binære værdi '00000000' i både TRISA og TRISB. På PORTC skal bit 7 (RC7) fungere som input for modtagelse af serielle data, mens bit 6 (RC6) skal agere output til transmission af serielle data. Derfor indlæses den binære værdi '10000000' i TRISC for at opnå den ønskede konfiguration af disse pins. Kodeudsnit 5 illustrerer, hvordan koden overfører de korrekte bit-værdier fra arbejdsregisteret (W-registeret) til de respektive TRIS-registre.

```

53:     bank1
54:     movlw B'10000000'
55:     movwf TRISC      ; Set TRISC 7 til input (receive) og 6 til output (transmit)
56:     movlw B'00000000'
57:     movwf TRISB      ; Set all PORTB pins as output for magnetic valves
58:     movlw B'000000'    ; Set RA0 output stepper motor activation and RA1 output direction
59:     movwf TRISA

```

Kodeudsnit 5: Konfiguration af TRIS-registrene

ADCON1 - Analog-til-digital konverter

På PIC16F873 kan PortA pins fungere som både analoge og digitale porte. Denne port initialisering styres af ADCON1 registeret (A/D Control Register 1). Dette 8-bits register bestemmer konfigurationen af de analoge indgange på PortA. Ved initialisering er alle bits i ADCON1 resetter til logisk '0', hvilket indikerer, at alle PortA pins er konfigureret som analoge indgange.

For at kunne benytte de ønskede PortA pins (RA1 og RA2) som digitale udgange til styring af steppermotoren, er det påkrævet at omkonfigurere de tilsvarende bits i ADCON1 registeret. Ifølge figur X svarer de relevante bits PCFG3:PCFG0 til de fire mindste betydende bits i registeret.

Ved at indlæse den binære værdi '1110' i PCFG3:PCFG0 bitsne, omkonfigureres RA4-RA1 til at fungere som digitale porte, mens RA0 forbliver analog indgang (Se Orangehiglightet figur X). Denne værdi indlæses i ADCON1 registeret ved programmets opstartssekvens for at sikre den korrekte port konfiguration (Kodeudsnit 6).

060:	movlw B'00001110'	; Set RA1 to 4 to digital and leave RA0 as analog
061:	movwf ADCON1	

Kodeudsnit 6: Analog konfiguration

● ADCON1 (A/D converter control register 1) 9Fh

The A/D converter condition is controlled with this register.

7(0)	6(0)	5(0)	4(0)	3(0)	2(0)	1(0)	0(0)
ADFM	-	-	-	PCFG3	PCFG2	PCFG1	PCFG0

The value in the parenthesis is in the condition immediately after the turning on.

● ADFM : A/D result format select

- 1 : Right justified. 6 most significant bits of ADRESH are read as 0.
- 0 : Left justified. 6 least significant bits of ADRESL are read as 0.

● PCFG3,PCFG2,PCFG1,PCFG0 : A/D port configuration control bits

PCFG 3210	Port					Remarks
	AN4	AN3	AN2	AN1	AN0	
0000	A	A	A	A	A	
0001	A	V _{REF+}	A	A	A	
0010	A	A	A	A	A	Same as 0000
0011	A	V _{REF+}	A	A	A	Same as 0001
0100	D	A	D	A	A	
0101	D	V _{REF+}	D	A	A	
0110	D	D	D	D	D	
0111	D	D	D	D	D	Same as 0110
1000	A	V _{REF+}	V _{REF-}	A	A	
1001	A	A	A	A	A	Same as 0000
1010	A	V _{REF+}	A	A	A	Same as 0001
1011	A	V _{REF+}	V _{REF-}	A	A	Same as 1000
1100	A	V _{REF+}	V _{REF-}	A	A	Same as 1000
1101	D	V _{REF+}	V _{REF-}	A	A	
1110	D	D	D	D	A	
1111	D	V _{REF+}	V _{REF-}	D	A	

A: Analog port
D: Digital port
V_{REF+}: High reference voltage
V_{REF-}: Low reference voltage

Figur 44: Guide to use the pic dokumentation for instilling of ADCON1 registeret.

Opsætning af USART-kommunikation

For at aktivere USART-kommunikationen på PIC16F873-mikrocontrolleren, er der yderligere tre registre, der skal konfigureres, udover den tidligere TRISC-konfiguration.

Indledningsvist programmeres TXSTA-registeret (Transmit Status and Control Register) ved at aktivere transmit-funktionen via sætning af bit 5 (TXEN) samt selektere high-speed-tilstand gennem sætning af bit 2 (BRGH). Samtidig forbliver SYNC-bit'en cleared for at fastholde asynchronous mode (Kodeudsnit 7, linje 62-63). Dernæst konfigureres SPBRG-registeret (Baud Rate Generator Register) med henblik på at definere den ønskede bit-overførselshastighed.

Mode	Low-speed (BRGH=0)	High-speed (BRGH=1)
Asynchronous (SYNC=0)	Fosc/(64(X + 1))	Fosc/(16(X + 1))
Synchronous (SYNC=1)	Fosc/(4(X + 1))	

Fosc is the clock frequency of the PIC.

X is the value of SPBRG register. It is 0 to 255

Figur 45: Baudrate formel. (Asynchronous communication of PIC16F873 (USART), u.å.)

Ved reference til dokumentationens figur 45 og de angivne formler for asynchron high-speed-kommunikation (fremhævet med orange), kan den nødvendige SPBRG-værdi beregnes. Ved en systemklokfrekvens på 4 MHz og et mål om 115.200 bps baudrate, bliver den optimale SPBRG-værdi:

$$\frac{4 \cdot 10^6}{(16(x+1))} = 115200$$



Ligningen løses for x vha. WordMat.

$$x = \frac{337}{288} \approx 1,170139$$

Vi tilnærmer os en værdi på 1, hvilket vil give også en baudrate på:

$$\frac{4 \cdot 10^6}{(16(1+1))} = 125000$$

125000 er ca. 8,5% fra de 115200 som Arduinoen kommunikere med, hvilket vil virke fint i praksis.

Endelige tilpasses RCSTA-registeret (Receive Status and Control Register) ved at aktivere den serielle port gennem sætning af SPEN-bit'en (Bilag 4).

```

62:      movlw B'00100100' ; TXEN=1, BRGH=1 - TX enable, high bitrate
63:      movwf TXSTA
64:      movlw D'1'       ; Baud rate = 115200bps
65:      movwf SPBRG
66:      bank0
67:      movlw B'10000000'
68:      movwf RCSTA     ; SPEN=1 - Enables serial ports

```

Kodeudsnit 7: Initialisering af USART kommunikation.

Anvendte instruktionssæt

Dette afsnit præsenterer en grundlæggende gennemgang af de centrale instruktionssæt, der anvendes i koden for dette projekt. Teorien er baseret på databladet og en komplet liste over alle instruktionssættene, findes i bilag 13.

Instruktionssættene kan overordnet inddeltes i tre kategorier: bitoperationer, byteoperationer og kontroloperationer.

Bitoperationer omfatter:

- BSF: Sætter en bit i et filregister
- BCF: Nulstiller en bit i et filregister
- BTFSC: Tester en bit i et filregister og springer over næste instruktion, hvis bit'et er 0
- BTFSS: Tester en bit i et filregister og springer over næste instruktion, hvis bit'et er 1

Byteoperationer omfatter:

- ADDLW: Adderer en konstant 8-bit værdi til W-registret
- CLRF: Rydder et filregister
- MOVF: Flytter indholdet af et filregister til et andet register
- MOVWF: Flytter indholdet af W-registret til et filregister
- DECFSZ: Dekrementerer et filregister og springer til en adresse, hvis resultatet ikke er 0
- NOP: Ingen operation

Kontroloperationer omfatter:

- MOVLW: Flytter en konstant 8-bit værdi til W-registret
- GOTO: Springer til en bestemt adresse i programhukommelsen
- CALL: Kalder en subroutine
- RETURN: Returnerer fra en subroutine
- ORG: Angiver startadressen for programmet
- EQU: Definerer en symbolsk konstant

Implementering af delay funktionerne

For at opnå præcise tidsforsinkelser i programkoden til PIC16F873 mikrocontrolleren er der implementeret tre separate forsinkelsesrutiner: Delay_10ms, Delay_100ms og Delay_1s. Disse rutiner udnytter mikrocontrollerens klokkefrekvens og instruktionscyklus til at generere de ønskede forsinkelser.

PIC16F873 opererer med en klokkefrekvens på 4 MHz. Ifølge databladet, jf. tabel X tager hver instruktionscyklus fire klokcyklusser at udføre. Derved kan varigheden af en enkelt instruktionscyklus udregnes som:

$Clockfreq := 4 \text{ MHz}$

$$Operationfreq := \frac{Clockfreq}{4} = 1 \text{ MHz}$$

For at finde hvor lang tid en operation tager

$$\text{bruges formelen } \frac{1}{freq}$$

$$TidPrOperation := \frac{1}{Operationfreq} = 1 \mu\text{s}$$

Hver operation tager altså 1 μs

En instruktionscyklus har således en varighed på 1 μs (mikrosekund).

Tabel 7: PINOUT description OSC1 og OSC2, pic16f87x datasheet side 7 Kilde:(PIC16F87X.pdf, u.å.)

Pin Name	DIP Pin#	SOIC Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	9	9	I	ST/CMOS ⁽³⁾	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	10	10	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, the OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.

For at implementere de nødvendige forsinkelsesrutiner defineres fire separate variable, d1 til d4, som benyttes i de forskellige løkker. Disse variable tildeles adresser inden for PIC16F873s general purpose register-område på bank 0, som strækker sig fra 0x20h til 0x7Fh ifølge databladet, jf. bilag 4, figur 100.

De fire variable får tildelt følgende adresser i register-området:

d1: 0x22h

d2: 0x23h

d3: 0x24h

d4: 0x25h

Ved at alllokere de variable på denne måde sikres det, at deres værdier kan tilgås og manipuleres effektivt inden for mikrocontrollerens begrænsede register-ressourcer.

I Delay_10ms funktionen (Kodeudsnit 8) benyttes en indre løkke, som tager cirka 4 instruktionscyklusser (1 cyklus for 'nop'-instruktionen, 1 for 'decfsz' og 2 for 'goto'). Dette fremgår af bilag 13, her skal det også bemærkes at hvis Program counter er blevet modifieret, tager decfsz instruktionen også 2 cyklusser. Hver iteration af den indre løkke har derfor en varighed på omkring 4 μs . For at opnå en forsinkelse på 10 ms, er det nødvendigt med cirka 2.500 iterationer af den indre løkke ($10 \text{ ms} / 4 \mu\text{s} \approx 2.500$). Da en 8-bits variabel kun kan indeholde værdier op til 255, kan den indre løkke maksimalt itereres 255 gange. Derfor introduceres en ydre løkke, som gentager den indre løkke det fornødne antal gange. Ved at dividere 2500 med 255 tilnærmes en kvotient på 10, hvorfor den ydre løkkens iterationsværdi sættes til 10, mens den indre løkkens iterationsværdi sættes til 255.

For Delay_100ms funktionen, kræves cirka 25.000 iterationer af den indre løkke ($100 \text{ ms} / 4 \mu\text{s} \approx 25.000$). Dette opnås dog mere effektivt ved at kalde Delay_10ms-rutinen 10 gange i en ekstern løkke. En tilsvarende fremgangsmåde benyttes for Delay_1s funktionen, hvor Delay_100ms-rutinen kaldes 10 gange.

328: Delay_10ms:

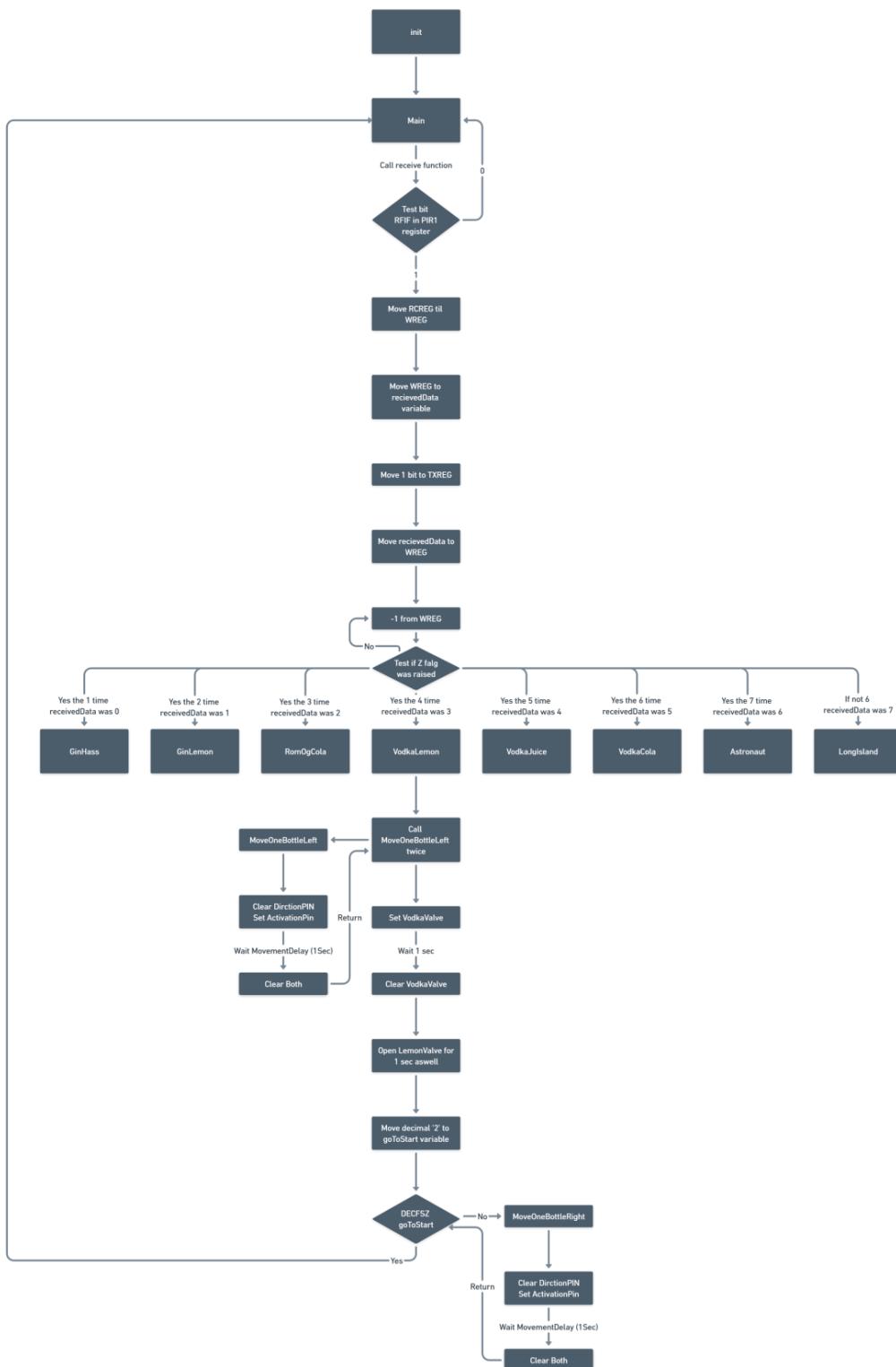
329: movlw D'10' ; Approximately 10 iterations for 10 ms delay

```
330:      movwf d1      ; Move W to d1
331: Delay_0:
332:      movlw D'255'    ; Each iteration of the inner loop takes approximately 4 µs
333:      movwf d2      ; Move W to d2
334: Delay_1:
335:      nop
336:      decfsz d2, 1    ; Decrement d2. Skip next instruction if zero
337:      goto $-2        ; Go back
338:      decfsz d1, 1    ; Decrement d1. Skip next instruction if zero
339:      goto Delay_0    ; Repeat inner loop
340:      return         ; Return from subroutine
```

Kodeudsnit 8: Delay funktionens virkemåde.

Kodens opbygning

I dette afsnit forklares den generelle kodestruktur, her vil funktionen af hovedløkken i main funktionen forklares der gør det muligt for programmet at blive ved med at køre. For at visualisere strukturen er der udarbejdet et Flowchart over PIC-koden på figur 46.



Figur 46: Flowchart over hovedløkken på PIC'en (Se bilag 20 for fuld størrelse)

Det fremgår af flowdiagrammet, at programmet indledes med en initialiseringsfase, som er beskrevet. Herefter kaldes hovedfunktionen "Main", som indeholder underliggende funktionskald.

Den første funktion, der kaldes, er "RecieveData". Denne funktion undersøger RCIF-bitets tilstand i statusregisteret PIR1 (linje 109 i kodeudsnit 9). RCIF-bitet indikerer, hvorvidt der er modtaget data via USART-forbindelsen. Hvis RCIF er 0, returneres der til "Main", og programmet fortsætter i løkken, indtil der modtages data. Hvis RCIF er 1, kopieres den modtagne data fra RCREG-registeret til den foruddefinerede variabel "recieveData" (linje 111-112 i kodeudsnit 9).

```
108: receiveData:
109:     btfss PIR1, RCIF      ; Check if data is available in the receive buffer
110:     goto Main          ; If not, continue looping
111:     movf RCREG, W       ; Read the received data from the receive buffer
112:     movwf receivedData   ; Store the received data in a variable
113:     return
```

Kodeudsnit 9: recievData funktion asm kode

Herefter kaldes funktionen "SendConfirmation", som sender byte-værdien '1' tilbage til Arduino-enheden for at bekræfte, at PIC-mikrocontrolleren har modtaget signalet.

Næste del af koden analyserer den modtagne byte-værdi for at bestemme, hvilken drik der er valgt (Kodeudsnit 10). Dette gøres ved at trække 1 fra WREG-registeret og derefter undersøge Z-flaget i STATUS-registeret (linje 82 i kodeudsnit 10). Hvis den modtagne byte-værdi i recieveData er 1, vil Z-flaget i STATUS-registeret være sat. I så fald vil programmet ikke springe over og kalde GinHass funktionen. Derimod vil det blive ved med at trække 1 fra WREG, indtil den når 0. Hvis ingen af drinkene kan få den til at gå i nul vil det derfor være den sidste drink LongIsland, som set i bilag 15 linje 103.

```
81:     movf receivedData, W  ; Move the received data into W register
82:     addlw -1            ; Subtract 1 from W register (since array index starts from 0)
83:     btfsc STATUS, Z    ; If Zero flag is set (receivedData was 1), call GinHass
84:     call GinHass
85:     addlw -1            ; Subtract 1 from W register
86:     btfsc STATUS, Z    ; If Zero flag is set (receivedData was 2), call GinLemon
87:     call GinLemon
```

Kodeudsnit 10: Drinks determination kode

Drinks funktionerne fungerer efter samme princip, som illustreret ved GinHass funktionen i kodeudsnit 10, følger et konsistent mønster. Først positioneres steppmotoren ved at kalde "MoveOneBottleLeft" et passende antal gange, indtil den når den korrekte position for den valgte drik. I tilfældet med "GinHass" er startpositionen Gin, så der behøves ikke at foretage yderligere positioneringer. Derefter aktiveres den magnetventil, der hører til den første ingrediens i drikken, ved at sætte det relevante bit i et registerkontrolleret signal (linje 150-152). Der ventes i et tidsrum svarende til den ønskede mængde, hvorefter ventilens bit nulstilles igen. Programmet fortsætter derefter ved at kalde "MoveOneBottleLeft" et antal gange, så motoren når den næste ingrediens' position (linje 154-156). Herefter aktiveres den tilhørende magnetventil, og processen gentages for alle ingredienser i den pågældende drik. Når den sidste ingrediens er dispenseret, kører programmet steppmotoren tilbage til udgangspositionen, som er Gin. Dette gøres ved at gemme den sidste drikkepositions indeks i en variabel kaldet "goToStart" (linje 165-166). Derefter opstilles en løkke, der subtraherer 1 fra WREG-registeret, så længe resultatet ikke er 0 (linje 167-169). Denne løkke kører, indtil steppmotoren når tilbage til udgangspositionen. Efter at have returneret til

udgangspositionen kalder programmet efter "Main"-funktionen, hvorved hovedløkken genoptages og er klar til at modtage en ny drikkebestilling.

```

148: GinHass:
149: ;Since Gin is the start bottle no need to turn just open valve
150: bsf PORTB, GinValve ;Open valve
151: call Delay_1s ;Dispense for 1 sec
152: bcf PORTB, GinValve ;Close valve
153: ;Go to Mango by turning 3 times and open valve
154: call MoveOneBottleLeft
155: call MoveOneBottleLeft
156: call MoveOneBottleLeft
157: bsf PORTB, MangoValve ;Open valve
158: call Delay_1s ;Dispense Mango for 1 sec
159: bcf PORTB, MangoValve ;Close valve
160: ;Open Lemon valve on the side.
161: bsf PORTB, LemonValve ;Open valve
162: call Delay_1s ;Dispense Lemon for 1 sec
163: bcf PORTB, LemonValve ;Close valve
164: ;Go to start
165: movlw D'3'
166: movwf goToStart ;Set goToStart to 3 meaning it needs to turn 3 times to the right to return
to start (Gin bottle)
167: call MoveOneBottleRight
168: decfsz goToStart, 1 ;Minus 1 from the goToStart variable and go 2 lines back if zero skip
169: goto $-2
170: ; Go back to main loop and listen for new drink
171: goto Main

```

Kodeudsnit 11: Eksempel på drinks funktion

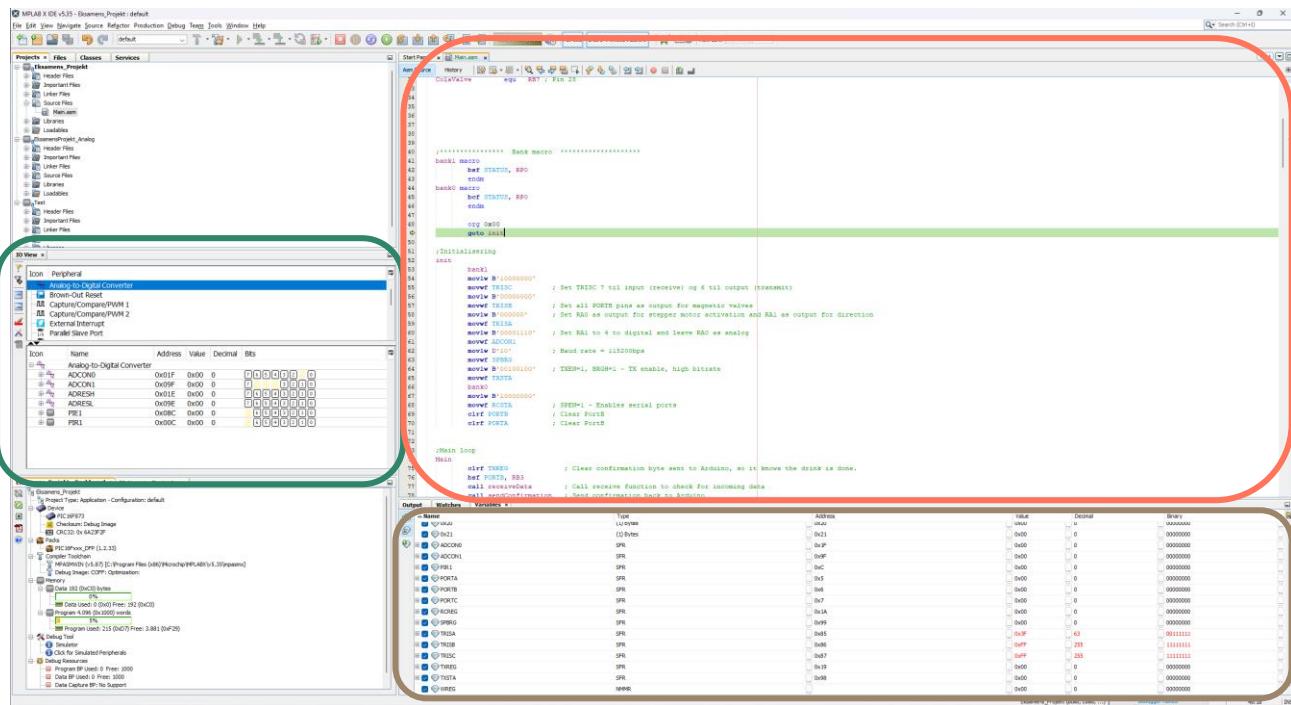
Debugging / Simulering af kode

Inden implementeringen af en mikrocontroller-baseret applikation afprøves den normalt i en integreret udviklingsmiljø (IDE) for at verificere, at koden fungerer korrekt, før den brændes over på enheden. Denne proces kaldes simulering og gør det muligt at gennemgå koden linje for linje og observere, hvordan registerværdierne ændrer sig. I dette tilfælde er MPLAB X IDE version 5.35 anvendt, da nyere versioner ikke længere understøtter MPAS-compileren, hvilket forhindrer fejlsøgning og kompilering af programmet for at verificere, at det fungerer som tiltænkt.

Hvis koden er uden fejl, kan projektet kompileres, og "Debug"-knappen kan aktiveres. Derefter vises "Build successful" i output-terminalen, hvorefter koden kan gennemgås linje for linje. I denne gennemgang fokuseres der på simulering af receiveData funktionen, som blev introduceret i et tidligere afsnit.

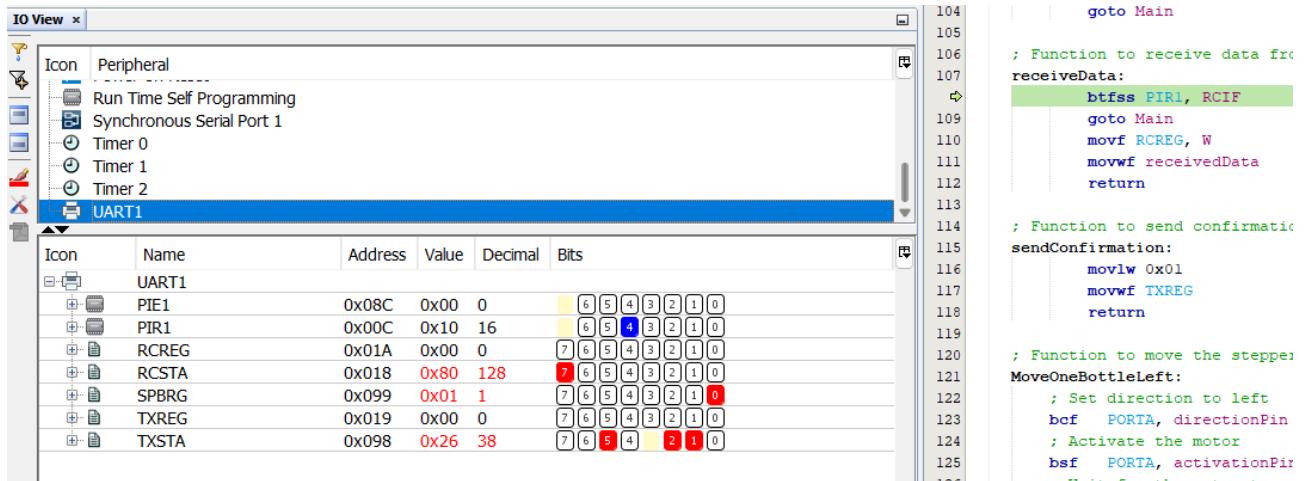
Først åbnes vinduerne "Watches" og "IO Viewer". "Watches"-vinduet bruges til at undersøge de binære værdier af registrene, mens "IO Viewer" anvendes til at manipulere registrene under fejlsøgningen.

Kodeeditoren er åben som standart, og er der man ser hvor simuleringen pt. er. Dette ses på figur 47.



Figur 47: MPLAB X Debugging

Ved at trykke på F7-tasten (Step Into) kan simuleringen gå til næste linje i koden. På figur 48 ses et eksempel, hvor man kontrollerer, om RCIF-bitten i PIR1-registeret er sat. Hvis ikke, springes der til næste linje, som fører tilbage til hovedfunktionen.



Figur 48: Kontrol af RCIF-bitten (Simulering)

Bilag 4 påviser at RCIF-bitten sider på 5 bit i PIR1 registeret, IO Vieweren viser at denne ikke er sat, og vi forventer at den går til næste linje, som illustreret på figur 49.

```

receiveData:
    btfss PIR1, RCIF
    goto Main
    movf RCREG, W
    movwf receivedData

```

Figur 49: Kontrol af btfss instruksen (Simulering)

Med samme udgangspunkt som figur 49. Kan man ved at klikke på bit 5 i "IO Viewer" sætte RCIF-bitten, og simuleringen kan derefter fortsætte til næste linje, som vist på figur 50.

Icon	Name	Address	Value	Decimal	Bits								
UART1													
PIE1		0x08C	0x00 0		<table border="1"><tr><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr></table>	6	5	4	3	2	1	0	
6	5	4	3	2	1	0							
PIR1		0x00C	0x30 48		<table border="1"><tr><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr></table>	6	5	4	3	2	1	0	
6	5	4	3	2	1	0							
RCREG		0x01A	0x00 0		<table border="1"><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr></table>	7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0						
RCSTA		0x018	0x80 128		<table border="1"><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr></table>	7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0						
SPBRG		0x099	0x01 1		<table border="1"><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr></table>	7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0						
TXREG		0x019	0x00 0		<table border="1"><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr></table>	7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0						
TXSTA		0x098	0x26 38		<table border="1"><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr></table>	7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0						

```

104        goto Main
105
106 ; Function to receive data :
107 receiveData:
108     btfss PIR1, RCIF
109     goto Main
110
111     movf RCREG, W
112     movwf receivedData
113     return
114
115 ; Function to send confirmation
116 sendConfirmation:
117     movlw 0x01
118     movwf TXREG
119     return
120
121 ; Function to move the step
122 MoveOneBottleLeft:
123     ; Set direction to left
124     bcf PORTA, directionPin
125     ; Activate the motor

```

Figur 50: Kontrol af RCIF-bitten når sat (Simulering)

Herefter flyttes værdien fra RCREG til WREG, og der simuleres, at der modtages en byte med værdien 5 ved at klikke på bit 3 og 1 i "IO Viewer", som vist på figur 51.

UART		RCREG	Value	Decimal	Bits								
PIE1	0x08C	0x00 0			<table border="1"><tr><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr></table>	6	5	4	3	2	1	0	
6	5	4	3	2	1	0							
PIR1	0x00C	0x30 48			<table border="1"><tr><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr></table>	6	5	4	3	2	1	0	
6	5	4	3	2	1	0							
RCREG	0x01A	0x05 5			<table border="1"><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr></table>	7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0						
RCSTA	0x018	0x80 128			<table border="1"><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr></table>	7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0						
SPBRG	0x099	0x01 1			<table border="1"><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr></table>	7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0						
TXREG	0x019	0x00 0			<table border="1"><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr></table>	7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0						
TXSTA	0x098	0x26 38			<table border="1"><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr></table>	7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0						

Figur 51: Kontrol af RCREG og movf instruksen (Simulering)

Klikker vi yderligere på F7-tasten kan man i "Watches" vinduet nu se, at værdien 5 er blevet overført til WREG, som forventet, se figur 52.

	Name	Address	Value	Decimal	Binary
✓	UX21	0x21	u	...	uuuuuuuu
✓	ADCON0	0x1F	0x00	0	00000000
✓	ADCON1	0x9F	0x0E	14	00001110
✓	PIR1	0xC	0x10	16	00010000
✓	PORTA	0x5	0x00	0	00000000
✓	PORTB	0x6	0x00	0	00000000
✓	PORTC	0x7	0x00	0	00000000
✓	RCREG	0x1A	0x00	0	00000000
✓	SPBRG	0x99	0x01	1	00000001
✓	Status	0x3	0x18	24	00011000
✓	TRISA	0x85	0x00	0	00000000
✓	TRISB	0x86	0x00	0	00000000
✓	TRISC	0x87	0x80	128	10000000
✓	TXREG	0x19	0x00	0	00000000
✓	TXSTA	0x98	0x26	38	00100110
✓	WREG	0x20	0x05	5	00000101

Figur 52: "Watches" vindue med verificering af WREG (Simulering)

Endelig flyttes værdien fra WREG til den definerede variable, denne variable sider på register 0x20 og vi kan derfor tilføje dette register til "Watches" vinduet, og se om dette register opdaterer når vi klikker F7.

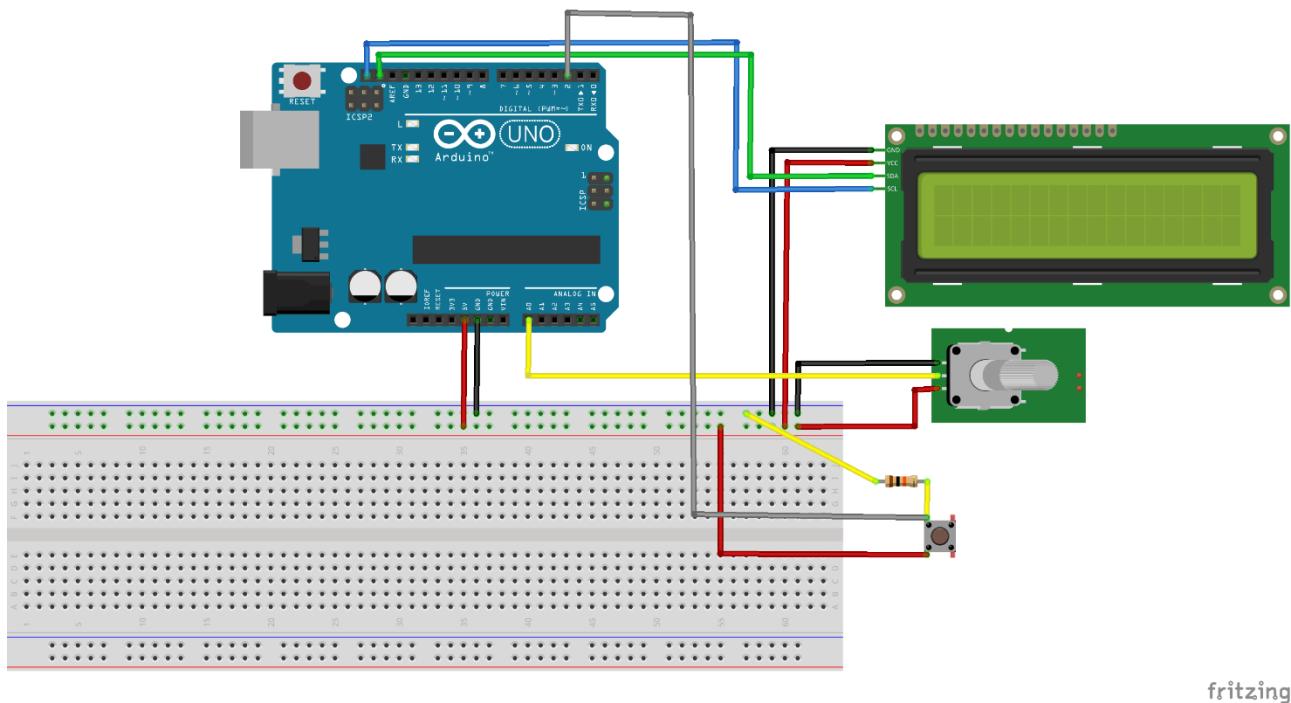
	Name	Address	Value	Decimal	Binary
	<Enter new watch>				
✓	0x20	0x20	0x05	5	00000101
✓	0x21	0x21	0x00	0	00000000
✓	ADCON0	0x1F	0x00	0	00000000
✓	ADCON1	0x9F	0x0E	14	00001110
✓	PIR1	0xC	0x10	16	00010000

Figur 53: "Watches" vindue med verificering af 0X20 registeret (Simulering)

Som vist på Figur 53 så agere receiveData funktionen præcis efter hensigten. Denne process er udført igennem hele koden for alle funktioner for at teste de virkede korrekt.

Brugergrænsefladen/USART-kommunikation:

Dette afsnit vil gå i dybden med den tekniske implementering af brugergrænsefladen, hvor brugere kan vælge den ønskede drik. Afsnittet vil fokusere på tre hovedaspekter: beskrivelsen af I2C-kommunikationen mellem LCD-skærmen og Arduinoen, funktionaliteten af potentiometeret og trykknappen, samt hvordan Arduinoen kommunikerer serielt med PIC-enheten.



Figur 54: Fritzing diagram over Brugergrænsefladen

Formålet med kredsløbet er at tillade justering af potentiometeret, hvilket resulterer i en varierende modstand. Afhængig af denne modstand vil en LCD-skærm, vise en distinkt linje. Hvis brugeren navigerer til en linje, der repræsenterer en ønsket drink, skal det være muligt at bekræfte valget ved at trykke på en knap. Efter valget skal displayet give en tilbagemelding om, at drinken laves.

Fritzing-diagrammet (figur 54) illustrerer forbindelserne for Arduino-enheden. Her er Arduinoens SDA- og SCL-pins forbundet til LCD-skærmen sammen med VDD 5V og VSS 0V. Potentiometeret er forbundet til VDD og VSS på de to yderste ben, og den analoge værdi af potentiometeret indlæses på Arduinoens A0-ben. Der er også forbundet en knap, der modtager VDD på den ene side af kløften, hvorfra værdien aflæses på den anden side og værdien indsættes i D2-benet. Knappen er forbundet med en PULLDOWN-resistor, som vil blive uddybet i det efterfølgende afsnit.

LCD-skærmens rolle i brugergrænsefladen

Kernefunktionaliteten i brugergrænsefladen udføres af en alfanumerisk LCD-skærm, som er i stand til at vise både tekst og tal. LCD-teknologien fungerer ved at styre krystallinsk væske mellem to polariserende plader. Når der anlægges en elektrisk spænding, ændrer krystallerne orientering, hvilket enten blokerer eller tillader lys at passere igennem skærmen. Den anvendte LCD-skærm har et fast layout på 16 kolonner og 2 rækker, som styres af en dedikeret controller-chip.

I2C-kommunikation mellem LCD og Arduino

Normalt kræver LCD-skærme et komplekst interface med mange forbindelser, herunder et potentiometer til at justere lysstyrkeforholdet mellem baggrund og forgrund. For at forenkle dette, benytter denne applikation en LCD-skærm, der integrerer en I2C-controller. I2C (Inter-Integrated Circuit) er en seriell kommunikationsprotokol, som muliggør dataudveksling mellem elektroniske enheder ved blot at bruge to signallinier - en dataline (SDA) og en clockline (SCL) (Santos, 2019). Denne simple forbindelse gør I2C særdeles velegnet til at tilslutte LCD-skærme og andre lavhastigheds-enheder til mikrokontrollere som Arduino.

I I2C-kommunikationen fungerer Arduinoen som master-enhed, der initierer kommunikationen og styrer clocksinalet, mens LCD-skærmen agerer som slave-enhed, der responderer på forespørgsler fra masteren. For at kunne kommunikere med LCD-slaveen skal Arduinoen kende dens unikke I2C-adresse. Denne adresse findes ved at køre et I2C-skanningsprogram, der serielt udsender forespørgsler til adresser fra 1 til 127 og registrerer eventuelle svar, hvilket afslører LCD-enhedens adresse (0x27 i dette tilfælde) Programmet findes i bilag 15.

Styring af LCD-skærmen via bibliotek

Til at styre indholdet på LCD-skærmen benyttes Arduino-biblioteket LiquidCrystal_I2C.h, som indkapsler I2C-kommunikationen. I koden initialiseres først LCD-objektet med den fundne I2C-adresse, antal kolonner og rækker (Linje 9 og 22). Herefter aktiveres LCD-skærmens baggrundsbelysning for at forbedre læsbarheden (Linje 24).

```
9: LiquidCrystal_I2C lcd(0x27, LcdColumns, LcdRows);
21: // Initialize LCD
22: lcd.init();
23: // Turn on LCD backlight
24: lcd.backlight();
```

Kodeudsnit 12: Initialisering af LCD-skærm

Igennem biblioteksfunktioner som lcd.setCursor() og lcd.print() kan Arduinoen nu adressere specifikke positioner på LCD-skærmen og vise den ønskede information, f.eks. navnet på den valgte drik. Denne funktionalitet integreres senere i koden, hvor den samspiller med potentiometeret og trykknappen for at give brugeren en intuitiv brugergrænseflade.

```
59: lcd.clear();
60: lcd.setCursor(0, 0);
61: lcd.print("Making");
62: lcd.setCursor(0, 1);
63: lcd.print(drinks[drinkIndex]);
```

Kodeudsnit 13: eksempel på lcd funktionaliteten

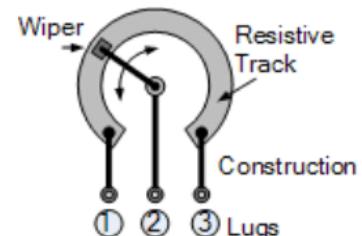
Potentiometer og trykknap

Potentiometeret og trykknappen giver brugeren mulighed for at navigere gennem de tilgængelige drikkevarer og foretage et valg.

Potentiometerets funktion

Et potentiometer er en justerbar elektrisk modstand, som fungerer ved at ændre modstandsværdien, når akslen drejes se figur 55. I denne applikation bruges potentiometeret til at navigere igennem listen af tilgængelige drikkevarer, der vises på LCD-skærmen.

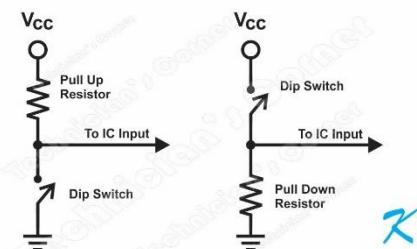
Potentiometeret forbindes til Arduinoen ved at tilslutte de to yder ben til henholdsvis 5V (VDD) og jord (GND), mens midter benet tilsluttet en analog indgang (A0 på Arduinoen). Ved at aflæse den analoge spændingsværdi på A0-porten kan Arduinoen registrere, hvad brugeren har drejet på potentiometeret, og dermed opdatere visningen på LCD-skærmen til den relevante drik.



Figur 55: Potentiometerets funktion

Trykknappens funktion

Trykknappen bruges til at vælge den drik, som brugeren har navigeret til via potentiometeret. Når brugeren trykker på knappen, registreres dette af Arduinoen via en digital indgang (D2). For at undgå ustabil indlæsning af knaptryk, forbindes D2-porten også med en pull-down-modstand (se figur 56), der trækker signalet ned til jord, når knappen ikke er trykket.



Figur 56: Pull up og pull down modstande

Integration i koden

I koden integreres potentiometerets og trykknappens funktionalitet således:

Potentiometerets analoge værdi aflæses på A0-porten og map funktionen bruges til at kortlægge værdierne fra 0 til 1023 med værdier fra 0 til 7 så programmet kan konvertere den analoge værdi om til et drink index (Kodeudsnit 14).

```
32: int sensorValue = analogRead(potPin);
35: drinkIndex = map(sensorValue, 0, 1023, 0, numDrinks);
```

Kodeudsnit 14: Potentiometer funktionalitet i koden

Når brugeren trykker på knappen (Højt signal på D2), går programmet ind i den if statment der ses i kodeudsnit 15, her sendes indekset for den valgte drik serielt til PIC-enheden, og LCD-skærmen viser en meddelelse om, at drikken bliver lavet kodeudsnit 13.

```
55: if (digitalRead(buttonPin) == HIGH) {  
56:   Serial.write(drinkIndex);
```

Kodeudsnit 15: kappens funktion i koden

Seriel kommunikation mellem Arduino og PIC

Det sidste element i brugergrænsefladen er den serielle kommunikation mellem Arduinoen og PIC-enheten. Denne kommunikation muliggør, at når brugeren vælger en drik, kan Arduinoen sende informationen videre til PIC-enheten, som står for fremstillingen af drikken.

Seriel kommunikation via USART

Arduinoen og PIC-enheten kommunikerer serielt via USART (Universal Synchronous Asynchronous Receiver Transmitter) -protokollen. USART er en udbredt seriell kommunikationsstandard, som benytter to linier - en transmitterlinie (TX) og en modtagerlinie (RX) - til at overføre data mellem de to enheder. I dette tilfælde forbindes Arduinoens TX-pin til PIC-enhedens RX-pin, og Arduinoens RX-pin til PIC-enhedens TX-pin. Denne krydsede forbindelse muliggør, at de to enheder kan sende og modtage data symmetrisk.

Dataoverførsel mellem Arduino og PIC

Når brugeren trykker på knappen for at vælge en drik, sender Arduinoen det tilhørende drinkindeks som en enkelt byte via den serielle forbindelse, linje 56 kodeudsnit 15.

PIC-enheten lytter på den serielle port og modtager dette drinkindeks, den fortolker det ved hjælp af funktionen forklaret i kodeudsnit 10. Herefter sender PIC'en en bekræftelses bit tilbage til arduinoen der viser at den har forstået det den sendte. Se kodeudsnit 16

```
117: sendConfirmation:  
118:     movlw 0x01          ; Send a confirmation byte back to Arduino  
119:     movwf TXREG  
120:     return
```

Kodeudsnit 16: Afsendelse af bekræftelses bit til Arduino

Arduinoen venter 5 sekunder, hvorefter den aflæser om den har fået tilsendt noget data fra PIC'en, hvilket den har vha PIC'en har forstået den sendte byte. Arduinoen påbegynder et while loop, som illustreret på linje 70-73 i kode udsnit 17. Her venter den indtil PIC'en clear sit TXREG, som den først gør når drinken er færdig se linje 75 bilag 5.

```
65: delay(5000);  
67: char data = Serial.read();  
70: while (data == 0x01) {  
71: // Waiting while the drink is being made  
72: data = Serial.read();  
73: }
```

Kodeudsnit 17: Arduino forståelse af bekræftelses bit

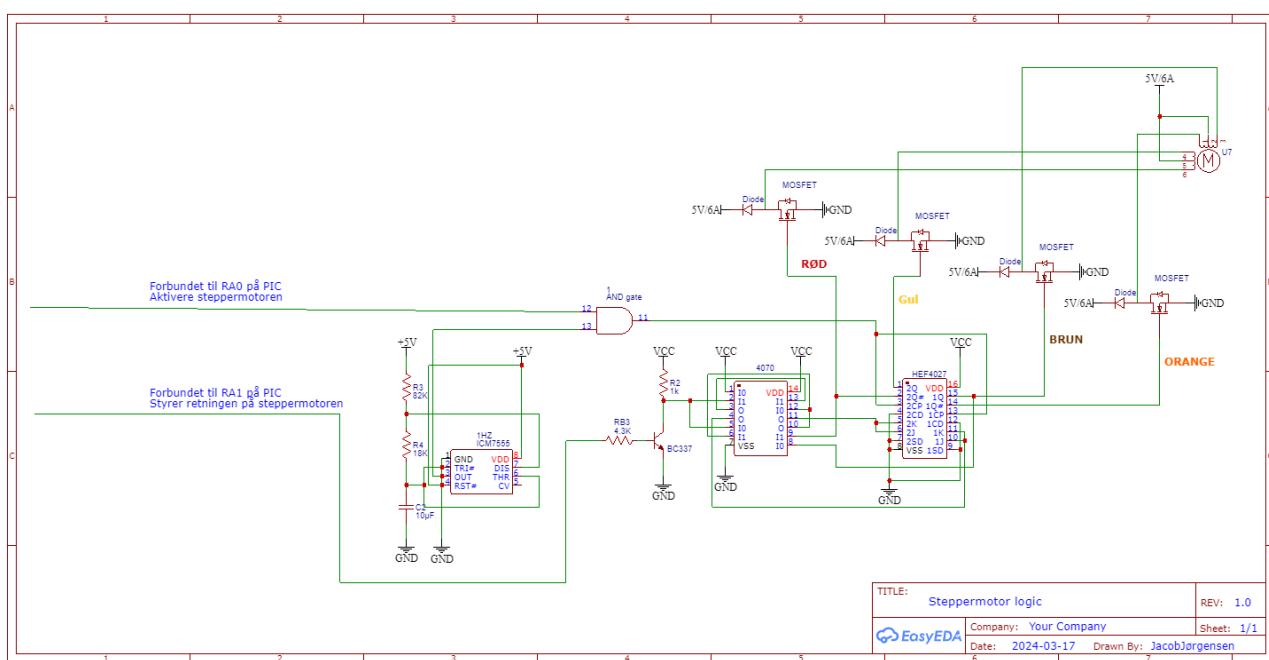
Roterende del

Denne del er ansvarlig for at dreje stepper motoren så det er den korrekte drink der står over koppen.

Kredsløbet fungerer ved at PIC16F873-mikrocontrolleren sender digitale signaler til steppermotoren via to udgange - RA1 og RA2. RA1 styrer om motoren skal rotere eller ej, mens RA2 bestemmer rotationsretningen. Jf. figur 57 for el teknisk diagram over den roterende dels komponenter.

For at generere de præcise signalsekvenser der kræves til at aktivere motorspolerne i den rette rækkefølge, anvendes en ICM7555 timer og en 4070 logiske gate. Timeren genererer en klokkefrekvens, der fødes ind i en 4070 XOR-gate. Udgangen fra XOR-gaten går derefter til en flip-flop kreds, som konverterer signalet til de endelige sekvenser til steppermotoren. Disse sekventielle signaler sendes til steppermotoren via MOSFET transistor, der fungerer som en switch. En flybagdiode er også tilkoblet for at beskytte kredsløbet mod strøm/voltagespikes fra motorens induktive belastning.

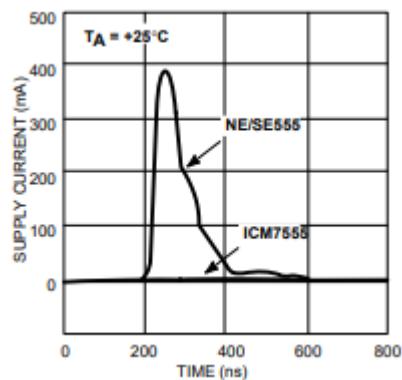
Fordelen ved denne løsning er først og fremmest, at vi sparer digitale udgange på PIC'en ved kun at bruge 2 i stedet for de 4 der normalt ville kræves til at styre steppermotoren direkte. Dette giver mulighed for at bruge de resterende udgange til andre formål. Derudover er denne implementering med separat timer og logikkredse mere modulær og skalerbar, da det gør det nemmere at justere motorhastigheden eller ændre signalsekvenserne uden at skulle omprogrammere PIC'en.



Figur 57: El teknisk diagram over logisk steppermotor kontrol (Se bilag 21 for fuld størrelse)

ICM7555

ICM7555, er valgt til dette projekt på grund af dens adskillige fordele over den mere traditionelle bipolare 555 timer. Ifølge databladet, se bilag 7, har ICM7555 flere fordele som blandt andet et ekstremt lavt strømforbrug, høj frekvensydelse op til 500 kHz samt et bredt forsyningsspændingsområde. Derudover undgår den det problem med currentspikes på udgangen, som ses i de bipolare 555-enheder, jf figur 58. Disse egenskaber gør ICM7555 til et oplagt valg til at generere de nøjagtige signalsekvenser, der kræves for at drive steppermotoren.



Figur 58: Outputtet forsyningsstrøm ved outputovergang

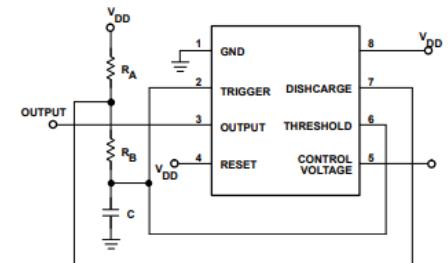
I kredsløbet bruges ICM7555'erns astabile drift, hvor man har 3 eksterne komponenter, 2 modstande og 1 kondensator. Her kan man styre timerens frekvens og duty cycle ved at tilpasse komponenternes værdier. I databladet opgives disse beregninger:

$$F = \frac{1.38}{(R_A + 2R_B)C} \quad D = \frac{R_A + R_B}{R_A + 2R_B}$$

Forbindelsen mellem komponenterne i astabil drift findes på figur 59. For at finde værdierne af de 3 komponenter vælges der en værdi for R_B på $18k\Omega$, og en kondensator værdi på $1\mu\text{F}$. Herefter kan vi opstille en ligning så vi kan finde R_A idet der ønskes en frekvens på 38Hz.

Omskrivningen af fomlen for at isolere R_A :

$$\begin{aligned} \frac{1.38}{F} &= (R_A + 2R_B)C \\ \Updownarrow \\ \frac{1.38}{F} &= R_A + 2R_B \\ \Updownarrow \\ \frac{1.38}{F} - 2R_B &= R_A \\ \frac{1.38}{\frac{38\text{Hz}}{1\mu\text{F}}} - 2 \cdot 18k\Omega &\approx 315,7895 \end{aligned}$$



Figur 59: Forbindelse af ICM7555 i Astabile drift

Det er i stedet for den beregnede værdi valgt at bruge en 120Ω modstand i stedet, da dette giver en dutycycle på ca. 50%. Herefter kan vi beregne frekvensen ved brug af formlen fundet i databladet:

Kendte værdier:
 $R_A := 120 \Omega$ $R_B := 18000 \Omega$ $C := 1 \mu F$

Formel for beregning af duty cycle ved ICM7555

$$\text{Duty} := \frac{R_A + R_B}{R_A + 2 R_B} = 0.502 \quad \text{Hvor } R_A \text{ er første modstand og } R_B \text{ er anden modstand.}$$

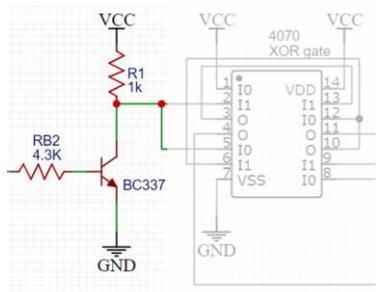
Formel for beregning af frekvens ved ICM7555

$$f := \frac{1.38}{(R_A + 2 R_B) \cdot C} = \frac{(3.821 \cdot 10)}{s} \quad \text{Hvor } R_A \text{ er første modstand, } R_B \text{ er anden modstand og } C \text{ er kondensator og } 1.38 \text{ er en konstant.}$$

Det kan dermed konstateres at med den tilnærmede værdi på 120Ω i stedet for 315Ω som vi fandt i første beregning får en frekvensen på 38,21Hz. Disse værdier vil testes senere i dette afsnit.

BC337

For at styre retningen af stepper motoren, er der valgt en BC337 NPN-transistor. BC337 er en almindeligt anvendt NPN-transistor, der er kendt for sin robusthed og alsidighed. Denne transistor er forbundet med to modstande, som illustreret på figur 60. For en mere dybdegående forståelse af denne komponent, henvises der til afsnittet om magnetventil.

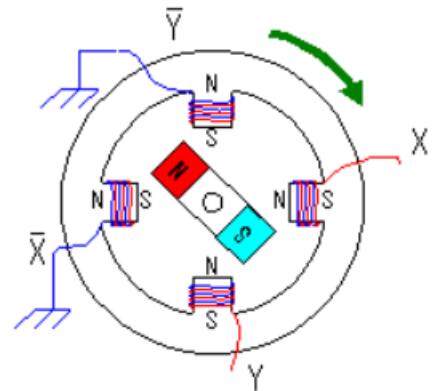


Figur 60: Forbindelse af BC337 til kontrol af stepper motor retning

Stepper Motor

Steppermotoren er en type elektrisk motor, hvor rotoren bevæger sig i diskrete trin, fremfor en kontinuerlig rotation. Som vist i figur 61, er motoren opbygget med fire spoler, der er forbundet til fire indgangsledninger. Ved at ændre polariteten af disse ledninger kan man få motoren til at dreje i en ønsket retning.

Princippet bag styringen af steppermotoren er følgende: Der skal altid være to spoler, der er aktive og har modsatte polariteter. Ved at skifte, hvilke to spoler der er aktive, og i hvilken rækkefølge, kan man få motoren til at dreje i den ønskede retning. Det fremgår af figur 59, at det kun er en af de to Spolesæt der skal opdatere af gangen. Fx ser vi med den klokvis rotation at det er X der skal ændre værdi, hvorefter det så bliver Y bagefter. Denne rækkefølge fortsættes for at opnå den ønskede rotation. Hvis man ønsker at dreje motoren i modsat retning, skal rækkefølgen af de aktive spoler blot ændres i den modsatte retning. På den måde kan man opnå både frem- og tilbagekørsel af steppermotoren ved at styre, hvilke spoler der er aktive, og i hvilken rækkefølge.



Figur 61: Opbygning af stepper motor

Clockwise control

X, \bar{X} , Y and \bar{Y} are controlled in the following order.

X	\bar{X}	Y	\bar{Y}	Step angle
0	1	0	1	0°
1	0	0	1	90°
1	0	1	0	180°
0	1	1	0	270°

"0" means grounding.

Figur 61: Modsat uret kontrol af Spolesæt

Counterclockwise control

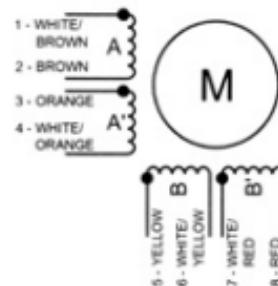
X, \bar{X} , Y and \bar{Y} are controlled in the following order.

X	\bar{X}	Y	\bar{Y}	Step angle
0	1	0	1	0°
0	1	1	0	-90°
1	0	1	0	-180°
1	0	0	1	-270°

"0" means grounding.

Figur 62: Med uret kontrol af Spolesæt

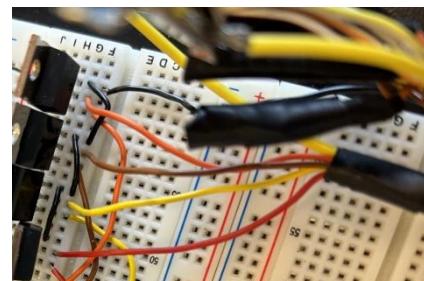
CONFIGURATION #6



Figur 63:
Ledningskonfiguration på
NEMA-23

Den valgte steppermotor er en NEMA 23-model, hvis specifikationer i henhold til producentens datablad omfatter en maksimal spænding på 75V, en strøm på 2,1 Ampere og 200 trin per omdrejning. Hvert individuelle trin svarer til en rotation på 1,8 grader. De efterfølgende afsnit vil omhandle den logiske styring af motorens drift.

Desuden kan ledningskonfigurationen for den valgte steppermotor udledes fra producentens datablad. Motoren har en 8-lednings konfiguration, som illustreret i figur 63. Herfra kan man se, hvordan motorens ledninger skal forbunes. VDD-ledningen (Stripped) skal tilsluttes strømforsyningen, mens de øvrige ledninger skal kobles til en logisk styreenhed, f.eks. en flipflop-kreds, for at sikre korrekt faseskift. Det er således essentielt, at de parvist farvekodede ledninger (brun/orange og rød/gul) aldrig har samme logiske værdi, da dette ville medføre fejlagtig drift af motoren. Figur 64 viser, hvordan stepper motorens 4 ledninger er forbundet med MOSFET'enes Drain ben.



Figur 63: Stepper motorens forbindelse til MOSFET

4070 Xor-gate

4070 Xor-gate er en digital logikport, der implementerer eksklusiv-ELLER-funktionen. Den er udstyret med fire uafhængige gates, hver med to indgange. Xor-gaten er designet til at give et højt output, hvis antallet af høje indgange er ulige. Dette betyder, at hvis begge indgange er lave (0) eller høje (1), vil outputtet være lavt (0). Hvis derimod kun én af indgangene er høj (1), vil outputtet være højt (1). Dette gør Xor-gaten ideel til opgaver, der kræver differentiering mellem forskellige kombinationer af indgangssignaler.

XOR		
x	y	F
0	0	0
0	1	1
1	0	1
1	1	0



Figur 64: XOR Logik

HEF4027 Dual JK Flip-Flop

HEF4027 er en dual JK flip-flop, der indeholder to uafhængige flip-flops, hver med direkte sæt (SD) og nulstil (RD) indgange. En flip-flop er en enhed, der har to stabile tilstande og kan bruges til at gemme tilstandsoplysninger. JK flip-flop er en forbedring af SR flip-flop, hvor S og R er erstattet med J (sæt) og K (nulstil) for at undgå den ugyldige tilstand, der opstår i SR flip-flop. Hvis J og K er lave, vil outputtet forblive uændret, uanset ændringer i clockpulsen (CP). Hvis J og K er høje, vil outputtet skifte tilstand ved hver clockpuls. Dette gør JK flip-flop ideel til sekventielle kredsløb, hvor bestemte outputtilstande skal opnås baseret på en given række inputtilstande.

Disse to komponenter kan kombineres på en intelligent måde for at styre en steppermotor, som beskrevet i det efterfølgende afsnit.

Hardware kontrol af Steppermotor ved brug af Flip-flop og XOR-gate.

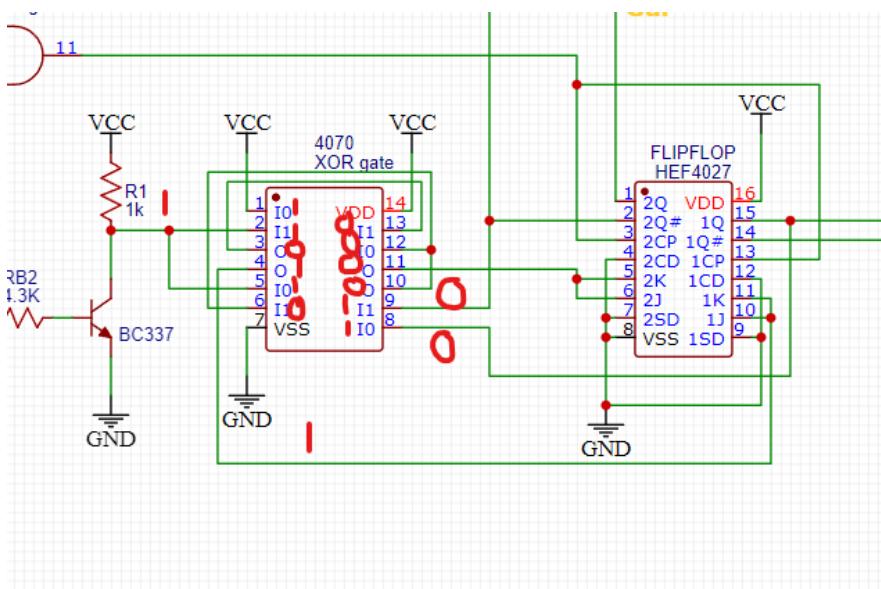
Til logisk styring af Steppermotoren er 4070 Xor gate og HEF4027 Dual JK Flip-Flop anvendt. Her bruges Xorgaten til logisk styring af retningen på steppermotoren. Det er forsøgt at indtægen hvordan Xorgaten styre hvilken rækkefølge FlipFloppen skal aktiveres i. Hvis vi ser på flipfloppens datanblad kan vi se at hvis J og K er sat til Lav vil en ændring i signalet på CP ikke ændre på outputtet. Da vi i det ovenstående afsnit fik etableret at kun en af steppermotorens spolesæt skal opdateres afgange kan denne funktion anvendes ved at sætte J og K pins til høj på det spolsæt der ikke skal skifte og J og K til lav på det spolsæt der skal ændre

porlariitet. Dette er opnået ved hjælp af HEF 4070 Xor-gaten. Her kan man se på figur 67 hvordan logikken ser ud når det er den venstre spole der skal opdatere. Den status der er indtegnet på billedet indikerer at venstre side lige er blevet opdateret fra at outputtet 1 på not Q benet til at outputte 0 på not Q benet. Derved kommer der 0 fra den ene flipflop og 0 fra den anden flipflop dette gør at outputtet fra denne XOR gate i bunden til venstre bliver 0, når den er 0 og transistoren ikke er aktiveret vil de 2 øverste xor-gates outputte 0 hvilket resulterer i at den nederst til venstre XOR gate vil aktivere og opdatere FlipFlop'en til højre da J og K benene så sættes til høj. Denne opdatering af XOR-gaten sker altså i samme hastighed som ændringen af flipflopens output. Som er styret af frekvensen der er outputtet af Steppermotoren.

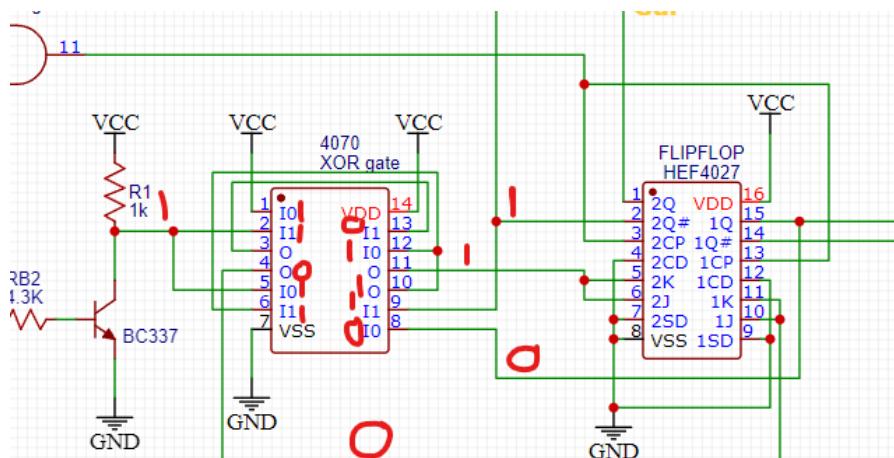
For logisk styring af steppermotoren, anvendes en 4070 Xor-gate og en HEF4027 Dual JK Flip-Flop. Xor-gaten bruges til at styre opdateringsrækkefølgende afhængit af om Steppermotoren skal køre med eller mod uret. FlipFlop'en bruges til at styre opdateringen af spolsættet. De logiske værdier der indgår til XOR-gaten er forsøgt tegnet på figur 66 og 67.

Hvis vi konsulterer databladet for HEF4027 (jf. bilag 9), kan vi se, at hvis J og K er sat til lav, vil en ændring i signalet på CP ikke ændre outputtet. Da vi tidligere har etableret, at kun et af steppermotorens spolesæt skal opdateres ad gangen, kan denne funktion anvendes ved at sætte J og K pins til lav på det spolesæt, der ikke skal skifte, og J og K til høj på det spolesæt, der skal ændre polaritet. Dette er opnået ved hjælp af HEF 4070 Xor-gaten.

Lad os tage et eksempel, hvor det er den venstre spole, der skal opdateres (jf. figur 65). Den status, der er indtegnet på billedet, indikerer, at venstre side netop er blevet opdateret fra at outputte 1 på not Q benet til at outputte 0 på not Q benet. Derved kommer der 0 fra den ene flip-flop og 0 fra den anden flip-flop. Dette resulterer i, at outputtet fra denne XOR-gate i bunden til højre bliver 0. Når den er 0, og transistoren ikke er aktiveret, vil de to øverste xor-gates outputte 0, hvilket resulterer i, at den nederste til venstre XOR-gate vil aktivere og opdatere Flip-Flop til højre, da J og K benene så sættes til høj. Denne opdatering af XOR-gaten sker i samme hastighed som ændringen af flip-flopens output, som er styret af frekvensen, der er outputtet af ICM7555 timeren.



Figur 65: Logisk udtryk i kredsløbet når begge udgange er ens.



Figur 66: Logisk udtryk i kredsløbet når udgangene er forskellige.

Således vil højre side opdatere, når q på højre side har samme logiske output som not Q på venstre side, og venstre flip-flop vil opdatere, når de begge har forskellige værdier (jf. Figur 66). Disse to stadier, illustreret på figur 65 og 66, vil således blive gentaget uendeligt. Hvorefter det bare er i omvendt rækkefølge, flip-flop bliver opdateret i, hvis transistoren er aktiveret, og der kommer 0V på pin 2 og 5.

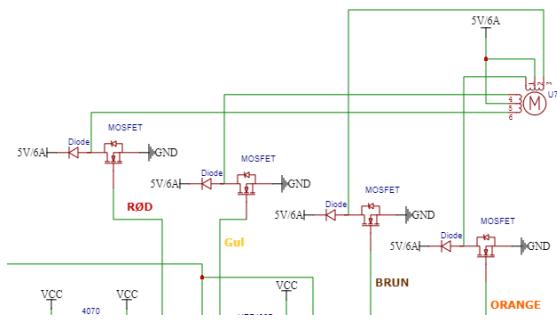
MOSFET - IRF540N

IRF540N er en type MOSFET (Metal-Oxide-Semiconductor Field-Effect Transistor), der ofte bruges som en switch i forskellige elektroniske applikationer, herunder styring af steppermotorer.

En MOSFET fungerer ved at anvende en spænding til gate-terminalen, hvilket skaber et elektrisk felt. Dette felt styrer "kanalen", som er en vej for strømmen mellem source og drain terminalerne. Når der er en tilstrækkelig spænding på gate-terminalen (kaldet tærskelspændingen), tillader MOSFET strøm at flyde fra drain til source. Når spændingen på gate-terminalen er under tærskelspændingen, stopper strømmen med at flyde. Dette gør det muligt for MOSFET'en at fungere som en switch.

IRF540N MOSFET er valgt til dette formål af flere grunde:

- **Høj strømkapacitet:** IRF540N kan håndtere høje strømme (op til 33A), hvilket gør den ideel til styring af steppermotorer, der ofte kræver høj strøm for at fungere effektivt.
- **Høj spaendingstolerance:** Denne MOSFET kan håndtere spændinger op til 100V på drain-source kanalen, hvilket giver en bred vifte af anvendelser.
- **Lav $R_{DS(on)}$:** IRF540N har en lav on-state modstand ($R_{DS(on)}$) på omkring 49mΩ, hvilket betyder, at den har minimal effektab, når den er i aktiveret. Dette gør den energieffektiv og ideel til applikationer, der kræver langvarig eller kontinuerlig drift.



Figur 67: Forbindelse af MOSFET

Figur 68 illustrerer, hvorledes MOSFET'erne er forbundet i kredsløbet. Gate-bennet på MOSFET'erne er forbundet til udgangen af flipflopene, hvilket muliggør elektronisk styring af transistorernes on/off-tilstand. Drain på MOSFET'erne er tilsluttet steppermotorens drivledninger, mens Source-terminalen er forbundet til jord. Når Gate-bennet aktiveres, sænkes modstanden mellem Drain og Source, hvorved strøm kan flyde til motoren og drive den.

Desuden er der placeret en såkaldt flyback-diode på drain-bennet af hver MOSFET. Funktionen af disse dioder vil blive nærmere beskrevet i det efterfølgende afsnit.

Flybackdiode

Når man styrer en steppermotor, er det nødvendigt at kunne håndtere de induktive spændinger, der opstår, når motorens spole bliver slukket. Disse induktive spændinger kan forårsage skader på kredsløbet, hvis de ikke håndteres korrekt. Flybackdioder spiller en central rolle i at beskytte kredsløbet mod disse potentielte skadelige voltagespikes. En flybackdiode er en speciel type diode, der er placeret parallelt med motorspolen. Dens funktion er at give en vej for den induktive energi, når spolen slukkes. Når strømmen gennem spolen pludselig afbrydes, vil den induktive last forsøge at oprettholde strømmen, hvilket resulterer i en høj voltagespike. Strømmen ledes igennem flybackdioden i denne situation og giver et kredsløb for den induktive energi, så den kan dissiperes gradvist i stedet for at forårsage skader. Uden flybackdioden ville de høje induktive spændinger potentielt kunne beskadige transistorene i kredsløbet.

Kodegennemgang:

Den overordnede logik bag styringen af steppermotoren er realiseret gennem de tidligere beskrevne hardwarekomponenter. Selve programkoden er imidlertid forholdsvis simpel og baserer sig primært på brug af instruktionssættet til at sætte og clear bits (BSF og BCF). Skiftet mellem de forskellige tilstande, der styrer rotationen af steppermotoren, implementeres ved at kalde to separate funktioner - MoveOneBottleRight (illustreret i kodeudsnit 18) og MoveOneBottleLeft.

Disse funktioner aktiveres fra hovedløkken, når der skal drejes henholdsvis højre og venstre. Når en af funktionerne kaldes, sættes bit'en for retningspin (defineret som RA2 i initialiseringsfasen) og derefter bit'en for motor aktivering (RA1). Herved sendes et højt signal, som aktiverer steppermotoren i 1 sekund, svarende til den tidsforsinkelse, som MovementDelay-funktionen er programmeret til. Herefter clear bit'ene igen, hvorved steppermotoren stopper, og funktionen returnerer til hovedprogramsløkken.

```
MoveOneBottleRight:
```

```
; Set direction to right
bsf PORTA, directionPin
; Activate the motor
bsf PORTA, activationPin
; Wait for the motor to move one position
call MovementDelay
; Deactivate the motor
bcf PORTA, activationPin
bcf PORTA, directionPin
return
```

Kodeudsnit 18: MoveOneBottleRicht funktionen

Test og måling:

Test af timerfrekvens

For at verificere, at timerfrekvensen fra ICM7555 er i overensstemmelse med den beregnede værdi fra tidligere afsnit, blev outputpinden (pin 3) på ICM7555 målt. Som illustreret i figur 68, stemte den observerede frekvens ikke overens da vi kunne måde en frekvens på 33 Hz, dette er dog ikke noget problem for kredsløbet.



Figur 68: Multimeter ICM7555 Frekvens

Validering af FLIPFLOP-signaloutput vha. LED'er

For at verificere, at den korrekte outputsekvens genereres fra FLIPFLOP-kredsløbet, blev der tilsluttet LED'er i stedet for stepper motoren, som vist i den vedhæftede videofil, se figur 69. Videooptagelsen bekræfter, at LED-sekvensen matcher den beskrevne sekvens for stepper motoren, som blev gennemgået tidligere.

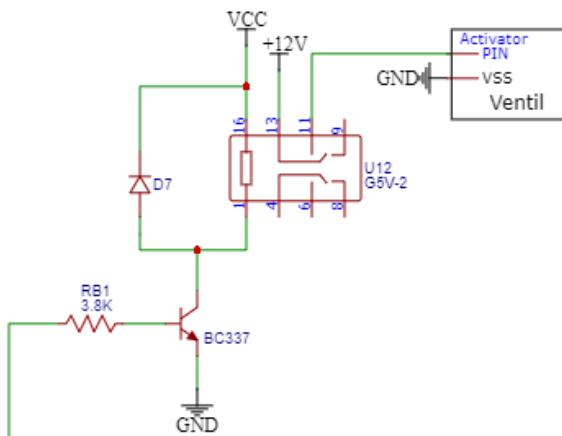


Figur 69: Video -
Validering af FlipFlop

Magnetventil

Dette afsnit behandler de komponenter, der anvendes til at dosere de alkoholiske ingredienser, som er nødvendige for hver drink.

Systemet består, som illustreret i Figur 70, af 8 identiske systemer, der er forbundet fra PIC'ens 8 PORTB-ben til en BC337-transistor, som fungerer som en switch til at aktivere et G5V-2-relæ. Relæet muliggør levering af de 12 volt, der kræves for at betjene magnetventilerne.



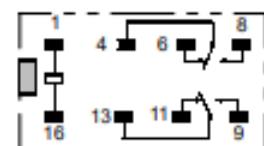
Figur 70: El teknisk diagram over tilslutningen af magnetventilerne.

Ifølge produktinformationen på magnetventilerne kræver de 12 volt spænding, hvor den ene terminal er en aktiveringsterminal og den anden skal forbindes til jord. Når der er signal til enheden, vil den åbne for væskeegenemløbet. Det er derfor nødvendigt at tilslutte et relæ i stedet for blot at forbinde til forsyningsspændingen.

Det anvendte relæ er et G5V-2-relæ. Relæer er komponenter, der fungerer som elektrisk styrede kontakter. De består af en elektromagnet, som kan tiltrække et anker inde i kredsen, illustreret i Figur 70A. Når ankeret trækkes til, oprettes en forbindelse mellem ben 13 og 11 i stedet for 13 og 9. Relæer bruges til at styre højspændingskredse med et lavspændingssignal. I dette projekt bruges de til at muliggøre en 12V-spænding til magnetventilerne, uden at dette forstyrrer styrekredsen.

Relæer aktiveres med forskellige strømværdier. Det relæ, der anvendes i projektet, aktiveres ifølge databladet, se bilag 6, ved 100 mA ved 5V. Der skal altså være en strøm på 100 mA ved ben 1 på relæet for at aktivere det. Dette tages i betragtning, når der skal beregnes på BC337-transistoren. Derudover forbindes der en flybackdiode parallelt med spolen inde i relæet, grundet samme princip som stepper motoren.

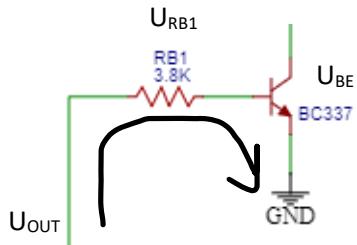
Terminal Arrangement/
Internal Connections
(Bottom View)



Figur 70A: Oversigt over relæ

BC337-transistoren bruges som switch til relæet i stedet for BC546, da BC337 har en kollektorstrøm på 800 mA, hvilket er betydeligt over det, som relæet kræver. BC546 har kun 100 mA kollektorstrøm, og BC337 anvendes derfor for at være på den sikre side.

Værdien af modstanden R_B , som illustreret i figur 71, på transistorens baseben, skal beregnes. Denne modstand styrer størrelsen af strømmen gennem transistoren, og skal derfor dimensioneres, så der kun løber 100 mA gennem kollektoren. Ved at anvende Kirchhoffs spændingslov for det lukkede kredsløb vist i figur 71, kan følgende ligning opstilles:



Figur 71: Kredsløbs analyse til BC337

Som illustreret i figur 71, kan spændingsfaldet opdeles i tre dele. Først har vi U_{out} , som er 5V spændingen fra PIC'en. Dernæst er der et spændingsfald over modstanden, givet ved $R_B \cdot I_B$. Endelig er der et spændingsfald fra base til emitter på BC337-transistoren, betegnet U_{BE} , hvor værdien ifølge databladet (Bilag 8) er 1,2V. Dette giver følgende ligning:

$$U_{out} - R_B \cdot I_B - U_{BE} = 0$$

Strømmen I_B kan udtrykkes ved hjælp af forstærkningsgraden β , som er en karakteristisk værdi for transistoren mellem 100 og 630 ifølge databladet. For beregningsformål vælges $\beta = 100$. Dermed kan I_B erstattes med:

$$I_B = \frac{I_C}{\beta}$$

Den oprindelige ligning kan nu omskrives for at isolere R_B :

$$R_B = \frac{U_{out} - U_{be}}{\frac{I_C}{\beta}}$$

Da alle værdier kendes, kan de indskrives i formlen hvortil man får R_B :

$$\frac{5V - 1,2V}{\frac{100mA}{100}} \approx 3.8k\Omega$$

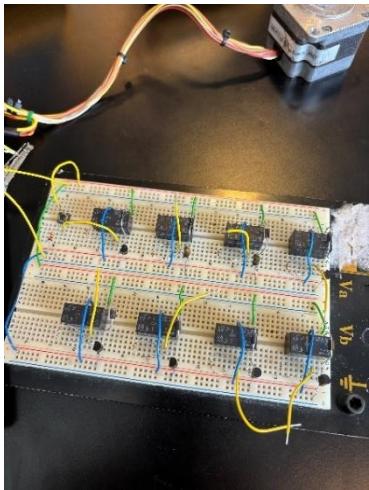
Modstanden skal altså have en værdi på $3.8k\Omega$ for at kunne aktivere relæet.

Kode

Koden til denne del er allerede gennemgået i Kodeudsnit 11. Illustrer, hvordan det relevante bit sættes, hvorefter man venter en periode, og clear det igen for at lukke ventilen.

Validering af signaloutput fra BC337

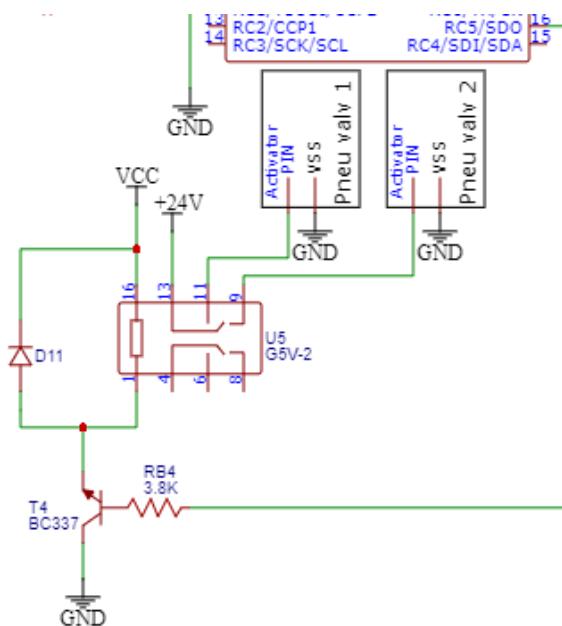
For at vurdere funktionaliteten af BC337-transistoren blev kollektorpinden undersøgt. Når der ikke var strøm på basisben, var kollektorspændingen relativt til jord målt til 5V, hvilket bekræfter den forventede adfærd. Når der var strøm på basisben, var kollektorspændingen 0V, som forventet. Derudover blev relæforbindelsen testet ved hjælp af en trykknap som vist i figur 72. Hvert Relæ registrerede et klik ved tryk på knappen, hvilket indikerer korrekt funktion.



Figur 72: Test af relæ

Pneumatik

Pneumatikmodulet fungerer på præcis samme måde som magnetventilerne, den eneste forskel er, at det kræver en forsyningsspænding på 24V til betjene. Der tilsluttes derfor en ekstra strømforsyning til produktet, som kan levere 24V. Som det også fremgår af figur 73, tilsluttes relæet på samme vis som beskrevet i det ovenstående afsnit, og beregningen af modstandsværdien R_B1 er tilsvarende. Den pneumatisk ventil har dog to aktiveringsterminaler, hvor den ene muliggør løft til den ene ende af stemplet, og den anden til den modsatte ende. For at kunne bevæge stemplet frem og tilbage, og dermed løfte koppen op og ned, skal terminalerne aktiveres skiftevis. Af denne årsag forbindes de to aktiveringsterminaler til samme side af relæet, hvilket muliggør, at den ene altid vil være aktiveret, og samtidig sparar et ekstra relæ.



Figur 73: El teknisk diagram over tilslutningerne til 24V relæ

Koden til dette modul er relativt simpel. Programmet sætter bit RC5, når drinken er valgt, hvilket får liften til at køre op. Når drinken er færdig blandet, cleares denne bit igen, og liften kører ned.

Reelle implementering

På trods af at alle simuleringer af koden fungerede som forventet og den korrekte hardwareopsætning var implementeret, opførte systemet sig ikke efter hensigten. Alle målinger viste, at PIC'en modtog de korrekte værdier via dens porte fra Arduinoen, men ikke desto mindre aktiverede PIC'en tilfældige ben, som ikke var indkodet i programmet. Gruppen forsøgte sig først med at implementere analog kommunikation mellem enhederne, men dette fungerede heller ikke, på trods af at simuleringerne viste den forventede opførsel. Det blev derfor besluttet at undgå PIC'en og i stedet implementere al koden på Arduinoen for at opnå en funktionsdygtig prototype. Dette medfører også at koden bliver meget simplificeret, i det Arduinoen er kodet i highend programmeringssproget C++, dertil behøves USART eller analog kommunikation imellem enheder ikke og projektet simplificeres yderligere.

Koden på Arduinoen og ændringerne i projektet beskrives i dette afsnit.

Forsøg på analog kommunikation

Gruppen forsøgte at anvende analog kommunikation, hvor Arduinoen skulle sende et pulsebreddemoduleret (PWM) signal til PIC'en, som så skulle aflæse spændingsforskellen og derved afkode, hvilken drink der skulle blandes. I denne implementering skulle koden på PIC'en ændres, så registeropsætningen blev modificeret, og der skulle implementeres en ny struktur til modtagelse og fortolkning af det modtagne signal samt kode til at kalde den korrekte drinkfunktion baseret på den modtagne værdi. Disse ændringer er vedlagt i bilag 18 for at dokumentere, hvor tiden er blevet anvendt. Udover de softwaremæssige ændringer blev der også indført et RC-filter for at filtrere støjen og konvertere PWM-signalet til et stabilt DC-signal, som illustreret i Figur 74. Koden på Arduinoen og RC-filteret fungerede som forventet, men PIC'en opførte sig ikke som programmeret på trods af, at det fungerede i simuleringen. Som tidligere beskrevet, fungerede denne implementering heller ikke, og gruppen besluttede derfor at implementere løsningen uden brug af PIC'en.



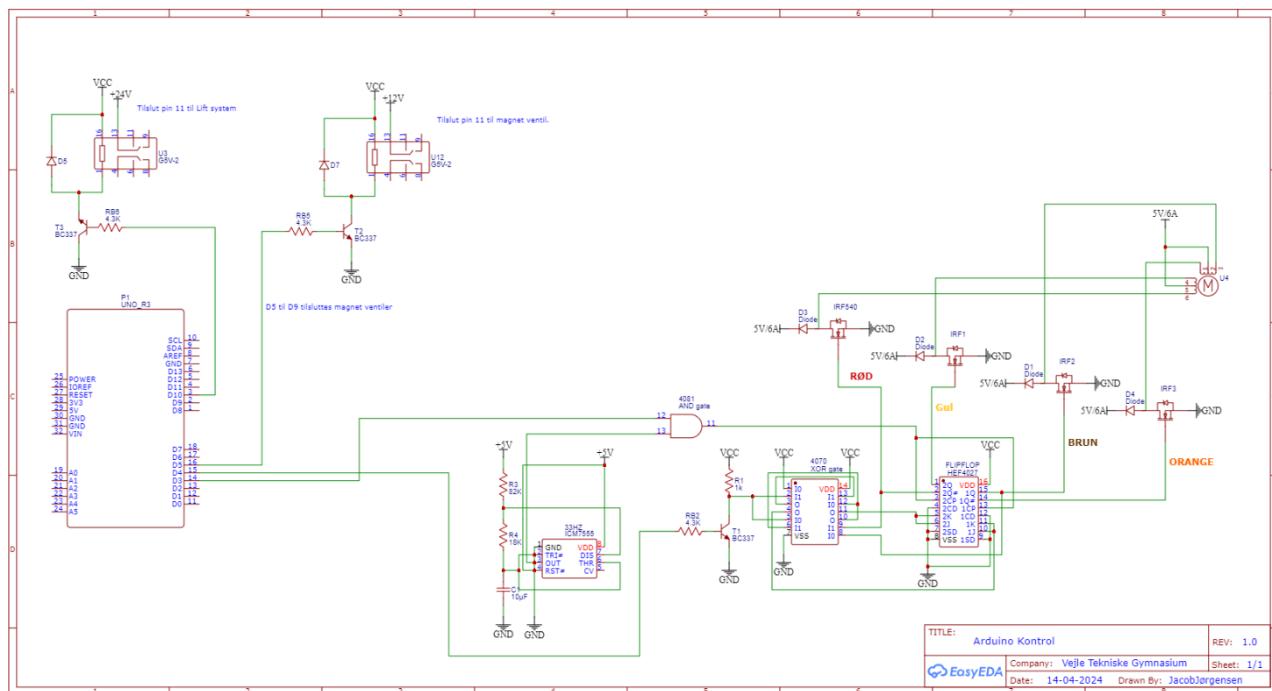
Figur 74: Signal på AN0 ved brug af RC filter til konvertering fra PWM til DC

Endelig løsning med Arduino

I dette afsnit gennemgås først de hardwaremæssige ændringer, der er foretaget for at opnå en funktionsdygtig prototype, hvorefter den nye Arduinokode gennemgås ved hjælp af figur 75.

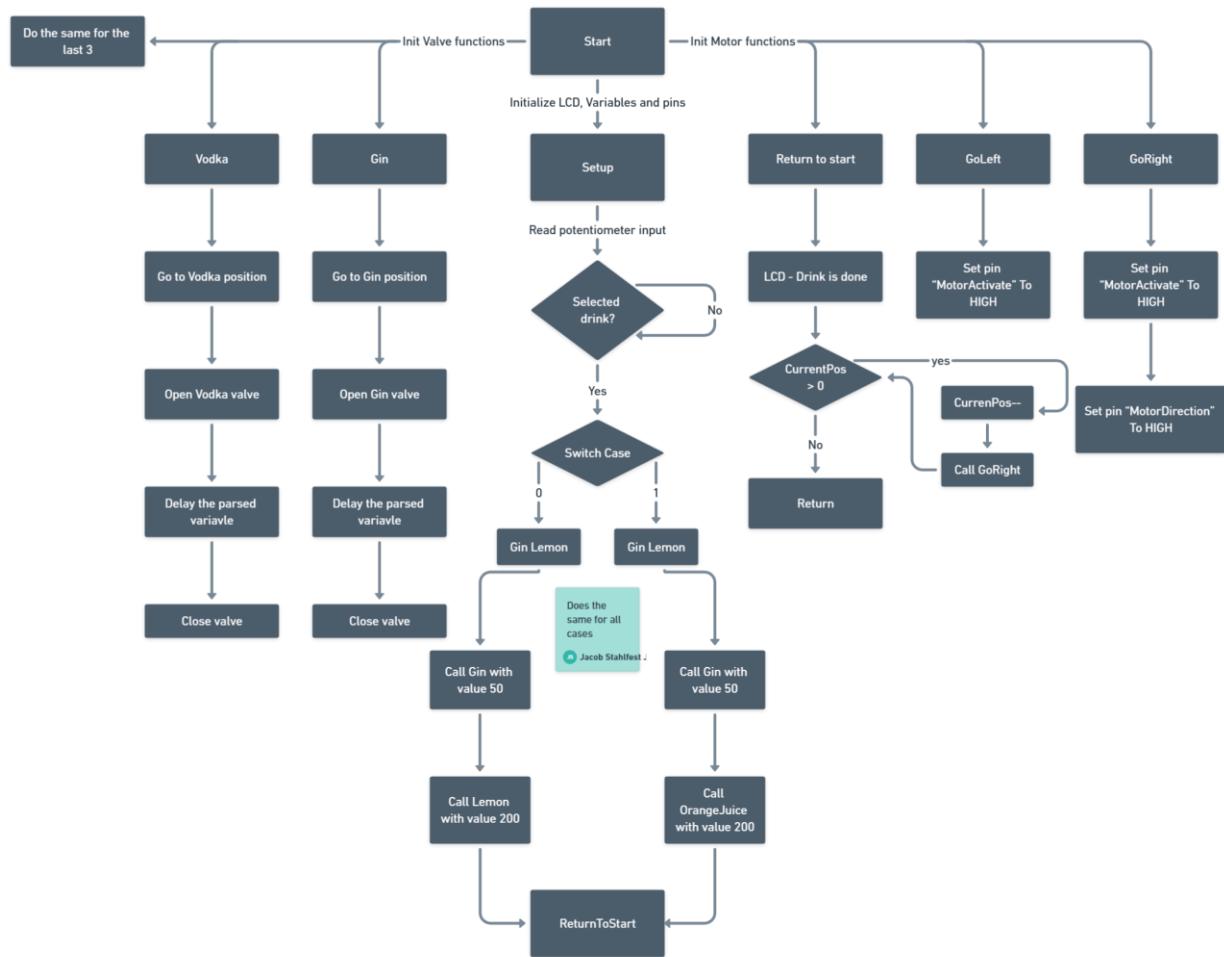
Den endelige løsning anvender det samme princip som den oprindelige idé, men alt er nu styret inden for ét loop i stedet for at køre to forskellige loops og informere to enheder om, hvor i loopet de befinder sig. Som det ses af figur 75, er der ikke foretaget nogle ændringer til hovedkomponenterne i produktet.

Motorstyringen og relæstyringen er den samme, det eneste, der er opdateret, er de digitale udgange på Arduinoen, så de styrer de korrekte enheder. Her er D5 til D9 tilsluttet de 5 magnetventiler, der anvendes, og D10 er forbundet til liftrelæet. Motorstyringen sker nu ved at sende et højt signal på D3, hvilket aktiverer steppermotoren, og ved et højt signal på D4 ændres motorens rotationsretning. Forbindelserne til brugergrænsefladen er uændrede.



Figur 75: Et teknisk diagram over tilslutningerne i kredsløbet uden PIC (Se bilag 21 for fuld størrelse)

De største ændringer er kodemæssige, hvor koden er vokset fra 85 linjer til 310 linjer. For at danne et overblik over den nye kodestruktur, er der udarbejdet et flowchart over koden på Arduinoen, jf. Figur 76.



Figur 76: Flow chart over Arduino kode uden PIC (Se bilag 20 for fuld størrelse)

Ud fra flowchartet visualiseres det, at starten af koden blot er ren initialisering af forskellige funktioner. Her initialiseres tre forskellige typer af funktioner. De første er GoLeft og GoRight, som fungerer ligesom i PIC-koden, se kodeudsnit 19. Her sættes blot pinnene til høj, indtil delay-funktionen har ventet i TimeBetweenValves-variablen, som er blevet testet til at være 1140 ms. Da det tog steppermotoren 5,7 sekunder at køre en hel omdrejning. Derefter deaktiveres de, og man returnerer.

```

119: void GoRight(){
121:     digitalWrite(MotorActivate, HIGH);
122:     digitalWrite(MotorDirection, HIGH);
123:     delay(TimeBetweenValves);
124:     digitalWrite(MotorActivate, LOW);
125:     digitalWrite(MotorDirection, LOW);
126: }

```

Kodeudsnit 19: GoRight funktionen

Herefter er der drink-funktionerne, hvor koden er omstruktureret, så når hovedløkken har brug for en ingrediens, kaldes navnet på den pågældende drink-funktion, og antallet af ml af den nødvendige ingrediens parses som en variabel. Hvis vi tager udgangspunkt i Gin-funktionen, kan vi på kodeudsnit 20 linje 55 se, at den tillader, at en variabel ml parses ind i funktionen. Herefter initieres et for-loop, der kører,

så længe dets betingelse ikke er opfyldt, altså så længe den nuværende position er mindre end GinPos. I løkken kaldes GoLeft-funktionen, og den nuværende position gemmes i en variabel CurrentPos, som inkrementeres med 1. Når for-loopet afbrydes, aktiveres GinValve-benet i ml * MsToMIConstant. MsToMIConstant var en konstant, vi ville have testet for at finde den tid, det tog at dispensere 100 ml, og derefter divideret med 100 for at bestemme konstanten. Dette ville gøre, at man kunne indtaste ml i funktionen i stedet for at angive den tid, ventilen skal være åben, hvilket fremtidssikre maskinen, da det vil gøre det nemmere at udvikle nye drinks.

```

55: void Gin(int ml){
56:   Serial.println("Gin valve function called");
57:   for (; GinPos > CurrentPos; CurrentPos++){
58:     GoLeft();
59:   }
60:   digitalWrite(GinValve, HIGH);
61:   Serial.println("Pouring");
62:   delay(ml * MsToMIConstant);
63:   digitalWrite(GinValve, LOW);
64: }
```

Kodeudsnit 20: eksempel på opbygning af drink funktion

Den sidste funktion, der er blevet introduceret i den nye kode, er ReturnToStart (bilag 16, linje 128-141). Denne funktion er egentlig blot et for-loop, der kører så længe, den nuværende position er større end 1 (udgangspositionen), hvorved den kører én gang til højre.

En af de største ændringer i koden findes i hovedløkken, da der er mange flere muligheder for at programmere den, når det gøres i et højniveau-programmeringssprog som C++. Her er switch-funktionen anvendt, hvilket muliggør, at man blot sender den en variabel, og den ser på værdien af denne, hvorefter der er opstillet 8 cases. Hvis drinkIndex er 0, betyder det, at kunden gerne vil have en Gin Lemon, og switch-funktionen aktiverer case 0 (se kodeudsnit 21). I denne case kaldes først Gin-funktionen, der dispenserer 50 ml gin i glasset. Herefter ventes der et antal millisekunder, der er defineret i toppen af koden, så ventilen kan nå at dryppe af. Dernæst kaldes den næste ingrediensfunktion, og her dispenseres 200 ml Lemon. Til sidst kaldes ReturnToStart, der kører tilbage til startpositionen, hvorefter der er et break, der afbryder switch-funktionen og returnerer til hovedløkken igen.

```

208: switch(drinkIndex){
209:   case 0:
210:     // Make Gin Lemon
211:     Serial.println("Making Gin Lemon");
212:     Gin(50);
213:     delay(MsBetweenIncredences);
214:     Lemon(200);
215:     delay(MsBetweenIncredences);
216:     ReturnToStart();
217:     delay(1000);
218:     break;
```

Kodeudsnit 21: Eksempel på hvordan en af case funktionerne ser ud.

Budget

Som afslutning på fremstillingen af det færdige produkt udarbejdes et budget, som giver overblik over produktionsprisen. Budgettet laves med udgangspunkt i styklister for hhv. det maskintekniske og elektroniske afsnit, jf. tabel 5 + 6. Her findes alle oplysninger om anvendt materialer, samt antallet heraf. Er på samme måde opplistet i budgetoversigten, hvor der er fundet stk/enhedspris. På denne måde findes den subtotal, som bruges på hver enkel del, og det er derfor nemmere at få overblik over hvilke dele af produktet udgør den største mængde af totalprisen. Nedenstående figur 77 viser budgettet i dets fulde størrelse, og her er det muligt at se materialeomkostningerne fra maskindelen udgør 1620,50 kr., mens de elektroniske komponenter udgør 1660,24 kr., som i alt bliver en total omkostningspris på 3280,74 kr.

Priserne til materialerne fra maskin er fundet gennem Claus i værkstedet, som har adgang til Ahlsall som er skolens hovedleverandør. Priserne på de elektroniske komponenter er fundet fra hjemmesiden Mouser electronics. Den eneste undtagelse for disse 2 hjemmesider er magnetventilen, som er købt hos RS electronics.

Budget til mekatronik eksamensprodukt				
Materialer jf. styklister	Antal	Enhed	Pris pr. enhed	Subtotal
Maskin				
Stålplade	0,81	m ²	kr 130,00	kr 105,30
Profilrør	2,22	m	kr 25,26	kr 56,08
Møtrik M6	4	stk	kr 0,60	kr 2,40
Bolt m. gevind M6	4	stk	kr 0,81	kr 3,24
Møtrik M4	2	stk	kr 0,11	kr 0,22
Bolt m. gevind M4	2	stk	kr 0,54	kr 1,08
Pneumatik luftcylinder	1	stk	kr 465,08	kr 465,08
Pneumatik ventil	1	stk	kr 972,36	kr 972,36
Pneumatik slange	1	m	kr 14,74	kr 14,74
<i>I alt:</i>				kr 1.620,50
Elektroniske komponenter				
Magnet ventil	5	stk	kr 180,00	kr 900,00
LCD Skærm	1	stk	kr 68,43	kr 68,43
Stepper Motor	1	stk	kr 608,00	kr 608,00
Mosfet - IRF540	4	stk	kr 7,47	kr 29,88
Transistor - BC337	7	stk	kr 2,57	kr 17,99
Timer - ICM7775	1	stk	kr 23,98	kr 23,98
XOR gate - 4070	1	stk	kr 3,47	kr 3,47
AND gate - 4081	1	stk	kr 4,39	kr 4,39
FlipFlop - HEF4027	1	stk	kr 4,10	kr 4,10
Relæ - G5V-2	6	stk	kr 24,58	kr 147,48
Diode	10	stk	kr 3,26	kr 32,60
Arduino Uno	1	stk	kr 159,41	kr 159,41
Potentiometer	1	stk	kr 6,28	kr 6,28
Kondensator	2	stk	kr 5,49	kr 10,98
Krystal oscilator	1	stk	kr 11,06	kr 11,06
<i>I alt:</i>				kr 1.660,24
Total kr 3.280,74				

Figur 77: Budget for produktet

Opfølgning på krav og produkt

Kravopfølgning

Det følgende afsnit vil efter prototypen færdiggørelse opfølge på de tidligere stillede krav. Dette gøres ved først at opsummere de tidligere opstillede krav, og kommenterer på dem. I projektet er der løbende opstillet krav til de enkelte faser. Her er først opstillet krav til løsningen, og disse er allerede konkluderet på i rapporten, se tabel 1, hvorved der blev valgt den endelige løsning i form af drinkmaskinen. Efter den valget om den endelige løsning blev truffet opstilles der her krav til hvad den endelige løsning skulle kunne, se tabel 3. I tabel 8 vil der for hver af kravene opstillet til den endelige løsning konkluderes på hvordan dette overholdes.

Tabel 8: Opfølgning på krav

Krav	Overholdelse af krav
Tiden fra drinkbestilling til servering må maksimalt være 30 sekunder	Da prototypen ikke skærer væske kan det ikke endegyldigt godkendes, men den fulde cycle tager under 30 sekunder.
Brugergrænsefladen skal være intuitiv og let at betjene for personer over 18 år	Brugergrænsefladen har vist sig at være let og intuitiv at betjene, når en testgruppe har skulle anvende den.
Løsningen skal kunne håndtere minimum 50 bestillinger i træk uden driftsstop	Da prototypen ikke kan skærke en drink f, har det ikke været muligt at kunne teste dette krav.
Støjniveauet må ikke overstige 70 dB målt 1 meter fra løsningen	Der er ikke udført test med decibelmåler, så kravet kan ikke endegyldigt godkendes. Det kan dog noteres, at den er støjsvag.

Foruden kravene til den endelige løsning bruges opstilling af krav også i afsnittet omkring materialevalget. Her det ligesom med kravene til løsningsforslagene, at kravene konkluderes med det samme i rapporten, og at disse har været afgørende faktorer for at træffe et endegyldige materialevalg.

Til den el tekniske del, er der ikke sat nogle generelle krav idet løsningerne er bygget og testet løbende.

Produktopfølgning

Den endelige prototype, som kan ses på figur 78, endte med at fungerer delvist. Med delvist funktionelt menes at alle funktioner virker hver for sig og/eller at der opstår enkelte problemer med disse under drift. Dette beskrives yderligere, samt der indsættes videoer af funktionerne hvor de virker under billede af produktet.



Figur 78: Billede af færdig prototype

Magnetventil

På prototypen er der 5 magnetventiler monteret på den roterende del, hvis funktion var at åbne op for væsken i beholderen ved højt signal. Dette viser sig dog ikke at fungerer, idet der er problemer med magnetventilen. Problemet bunder i at magnetventilen simpelthen ikke åbner helt op for flowet af væsken. Det er blevet forsøgt at finde løsningen til hvorfor dette sker, men uden held. Der gættes på at det skyldes at magnetventilen kræver en del tryk fra det system den indgår i. Med det tryk den får fra prototypen drypper der blot en smule væske fra ventilen, når denne åbnes. Dette er testet, og vist med film, se figur 79.



Figur 79: Video -
Test af
magnetventil

Pneumatik

Prototypen indeholder et sakselift system, hvis funktion er at løfte koppen op, således væsken ikke falder så langt og plasker. Denne sakselift som er 3D printet er koblet op til et pneumatisk system. Dette system virker også kun delvist idet der er problemer med det ventilmkul, som vi har monteret på prototypen. Selve systemet og funktionen af sakseliften der løfter koppen op og ned, virker når den blot tilsluttes et andet ventilmkul og der køres slanger til luftcylinderen. Princippet i hvordan dette pneumatisksystem skulle virke er vist med film, se figur 80.



Figur 80: Video -
Test af pneumatik

Stepper motor

Som centrum i prototypen sidder stepper motoren som drejer og styrer hele den roterende del. Med denne stepper motor opleves der udsving i driftssikkerheden, idet den til tider ikke følger kodens anvisninger. Det opleves, at stepper motoren kun kører den ene vej rundt hele tiden. Det ønskes ikke, da ledningerne som er forbundet til magnetventilerne så vil drejes sammen, krølles og smadres. For at undgå dette sker er koden programmeret således at den ved udskænkning drejer med uret og maskimalt en omgang (hvis bagerste flaske bruges), for så at dreje mod uret tilbage til nul-positionen. Denne metode er vist fungerende videoen fra figur 81. Samtidigt vedhæftes et eksempel, hvor stepper motoren drejer den forkerte vej, se figur 82.



Figur 81: Video -
Test af stepper
motor (funktionel)



Figur 82: Video - Test
af stepper motor
(ikke funktionel)

Hvorfor der kun er 5 varianter af væske?

I afsnittet *afgrænsning af produkt* beskrives der, som også illustreret i figur 8, at maskinen skal kunne have 5 pladser til flasker samt 3 til sodavand, som skal sidde på ydersiden af produktet. Som det fremgår af produktbilledet, har den altså ikke et ophæng til flasker på ydersiden, men kun de 5 pladser der er inde i maskinen. Dette er et valg der er truffet af 2 årsager, dels fordi skolen ikke havde 8 magnetventiler til rådighed, samt det at udseendet på maskinen blev flottere og mere symmetrisk af at undlade et ophæng på ydersiden af produktet. Det medfører altså at de mulige drinks minimeres i forhold til det som angives på figur 8.

Konklusion

Alt i alt har dette projekt mundet ud i en delvis funktionel prototype af en drinksmaskine der efter brugerens input, kan blande præcis den drink de ønsker mellem 5 væsketyper. Inden drinken blandes løftes koppen op til væskeudtaget med et pneumatisk sakseliftsystem. Dette input igangsættes fra den meget minimalistiske brugergrænseflade som giver brugeren mulighed for at scroll gennem de mulige drinks og slutteligt bekræfte sit valg med et tryk på knappen.

Projektets færdige prototype er ikke fuldt funktionsdygtig, men er i sine isolerede funktioner mulig at anvende efter ønsket funktion. Dette er blevet vist i rapportens afsnit *Opfølgning på krav*, samt der er argumenteres for årsagen til hvorfor prototypen ikke virker fuldstændigt. Foruden prototypen har projektet givet et fornuftigt indblik i hvordan det er at arbejde i både det maskin- og el-tekniske værkstedet, samt ikke mindst hvordan man skal få disse to til at spille sammen. Det har medført, at det har været muligt at udforme en rapport som på bedste vis illustrerer de færdigheder og den viden vi gennem året har tilegnet os.

Perspektivering

Fremtidsudsigten for vores produkt, er først og fremmest at få lavet en fuldt funktionsdygtig prototype som 'proof of concept'. Herefter vil næste step være at få lavet en markedsanalyse, der skal sikre at der faktisk er kunder til et endeligt produkt. Finder denne markedsanalyse ud af at der er et marked, vil processen med at få udviklet det endelige produkt begynde. Her skal laves om i konstruktionen sådan at den passer med de materialeovervejelser der er beskrevet i afsnittet *materialevalg*. Dette indebærer at plastdeler skal støbes, samt enkelte aluminiumsdele fabrikeres. Ydermere skal hele det elektroniske system optimeres sådan at det først og fremmest virker, men også fylder mindre. På den måde kan vi lave et produkt der er bedre og mere bekvemmeligt for slutbrugeren at have i sit køkken.

Vurdering

Vurderingen af gruppearbejdets samarbejde har været positivt. Vi har i gruppen evnet at arbejde opdelt i værkstederne, hvor Jacob primært har været i el, mens Max og Magnus har været i maskin. På trods af at vi har arbejdet meget opdelt, har vi været gode til at inddrage hinanden i hvor langt vi er kommet, samt inddrage hinanden i vigtige beslutninger.

På bagkant skulle vi have været bedre til at informerer hinanden endnu mere om hvordan vi har gjort det forskellige arbejds metoder, og hvordan tingene virker. Det positive ved det, er at vi har fået fremstillet en prototype, som er tæt på at fungere, og at vi nu frem til det mundtlige forsvar af projektet kan bruge tid på at sætte hinanden ind i det arbejde vi har lavet.

Litteraturliste

30292D.pdf. (u.å.). Hentet 19. april 2024, fra

<https://ww1.microchip.com/downloads/en/DeviceDoc/30292D.pdf>

Asynchronous communication of PIC16F873 (USART). (u.å.). Hentet 18. april 2024, fra

http://www.piclist.com/images/www/hobby_elec/e_pic7_7.htm

Configuration Word of PIC16F873. (u.å.). Hentet 3. april 2024, fra

http://www.piclist.com/images/www/hobby_elec/e_pic7_6.htm

PIC16F87X EEPROM Memory Programming Specification. (2002).

Santos, S. (2019, februar 1). *I2C LCD with ESP32 on Arduino IDE - ESP8266 compatible | Random Nerd Tutorials.* <https://randomnerdtutorials.com/esp32-esp8266-i2c-lcd-arduino-ide/>

SFR explanation for PIC16F873(2). (u.å.). Hentet 3. april 2024, fra

http://www.piclist.com/images/www/hobby_elec/e_pic7_1_2.htm#1

Figur Liste

<i>Figur 1: Skitse af løsningsforslag 1</i>	8
<i>Figur 2: Skitse af løsningsforslag 2</i>	9
<i>Figur 3: Skitse af det special designet glas til løsningsforslag 3.</i>	10
<i>Figur 4: Skitse af løsningsforslag 3</i>	10
<i>Figur 5: Skitse til løsningsforslag 4</i>	11
<i>Figur 6: Skitse til løsningsforslag 5</i>	12
<i>Figur 7: Blok diagram der illustrerer de el tekniske forbindelser</i>	14
<i>Figur 8: Oversigt over mulige drinks baseret på valgte ingredienser til drinksmaskinen</i>	15
<i>Figur 9: Samlingstegning af CAD-konstruktionen hentet fra rapportafsnittet 'Den mekaniske del'</i>	17
<i>Figur 10: Visualisation af stellets udformning</i>	19
<i>Figur 11: Arbejdstegning over steldel type 1</i>	20
<i>Figur 12: Flowdiagram over fremstilling af steldel</i>	21
<i>Figur 13: Arbejdstegning til den roterende del</i>	22
<i>Figur 14: Flowdiagram over fremstilling af den roterende del</i>	23
<i>Figur 15: Arbejdstegning til enclosuren</i>	24
<i>Figur 16: Flowdiagram over fremstilling af enclosure</i>	25
<i>Figur 17: Visualisering af brugergrænsefladen</i>	26
<i>Figur 18: Flowdiagram over samlingsproceduren for produktets enkeltdeler</i>	27
<i>Figur 19: Billede af pladeklipper</i>	29
<i>Figur 20: Billede af svejsestation</i>	30
<i>Figur 21: Billede af båndsav</i>	30
<i>Figur 22: Billede af søjleboremaskine</i>	31
<i>Figur 23: Tabel med oversigt over borehastigheder</i>	31
<i>Figur 24: Billede af pladebukker</i>	31
<i>Figur 25: Eksporterter af tegning</i>	32
<i>Figur 26: Valg af fil-type (STL)</i>	32
<i>Figur 27: Importeret STL-fil i prusa slicer</i>	33
<i>Figur 28: Menu til printerindstillinger</i>	33
<i>Figur 29: Visualisering af konstruktion</i>	34
<i>Figur 30: Beregningsmodel for nedbøjning</i>	34

<i>Figur 31: Formel for nedbøjning</i>	34
<i>Figur 32: Visualisering af træk/tryk i stelkonstruktionen</i>	36
<i>Figur 33: Visualisering af beregning</i>	36
<i>Figur 34: Formel for beregning af a-mål</i>	36
<i>Figur 35: Figur over sømfaktoren</i>	37
<i>Figur 36: Figur der viser trykspændingen</i>	37
<i>Figur 37: El diagram over hele kredsløbet - inddelt i hovedkomponenter. (Se bilag 21 for fuld størrelse uden inddeling)</i>	39
<i>Figur 38: Pinlayout for PIC16F873. (Datablad bilag 5)</i>	41
<i>Figur 39: Opsætning af krystaloscillator (Datablad bilag 5)</i>	41
<i>Figur 40: Krystaloscillator oversigt (Datablad bilag 5)</i>	41
<i>Figur 41: Opsætning af PIC'en</i>	41
<i>Figur 42: Microchip ICD2</i>	42
<i>Figur 43: PIN DESCRIPTIONS (DURING PROGRAMMING): PIC16F87X (PIC16F87X EEPROM Memory Programming Specification, 2002)</i>	42
<i>Figur 44: Guide to use the pic dokumentation for instilling af ADCON1 registeret.</i>	46
<i>Figur 45: Baudrate formel. (Asynchronous communication of PIC16F873 (USART), u.å.)</i>	47
<i>Figur 46: Flowchart over hovedløkken på PIC'en (Se bilag 20 for fuld størrelse)</i>	51
<i>Figur 47: MPLAB X Debugging</i>	54
<i>Figur 48: Kontrol af RCIF-bitten (Simulering)</i>	54
<i>Figur 49: Kontrol af btfss instruksen (Simulering)</i>	55
<i>Figur 50: Kontrol af RCIF-bitten når sat (Simulering)</i>	55
<i>Figur 51: Kontrol af RCREG og movf instruksen (Simulering)</i>	55
<i>Figur 52: "Watches" vindue med verificering af WREG (Simulering)</i>	56
<i>Figur 53: "Watches" vindue med verificering af 0X20 registeret (Simulering)</i>	56
<i>Figur 54: Fritzing diagram over Brugergrænsefladen</i>	57
<i>Figur 55: Potentiometerets funktion</i>	59
<i>Figur 56: Pull up og pull down modstande</i>	59
<i>Figur 57: El teknisk diagram over logisk stepper motor kontrol (Se bilag 21 for fuld størrelse)</i>	62
<i>Figur 58: Outputtet forsyningsstrøm ved outputovergang</i>	63
<i>Figur 59: Forbindelse af ICM7555 i Astabile drift</i>	63
<i>Figur 60: Forbindelse af BC337 til kontrol af stepper motor retning</i>	64

<i>Figur 61: Opbygning af stepper motor-----</i>	65
<i>Figur 62: Med uret kontrol af Spolesæt -----</i>	65
<i>Figur 63: Stepper motorens forbindelse til MOSFET -----</i>	66
<i>Figur 64: XOR Logik-----</i>	66
<i>Figur 65: Logisk udtryk i kredsløbet når begge udgange er ens.-----</i>	67
<i>Figur 66: Logisk udtryk i kredsløbet når udgangene er forskellige.-----</i>	68
<i>Figur 67: Forbindelse af MOSFET-----</i>	69
<i>Figur 68: Multimeter ICM7555 Frekvens-----</i>	70
<i>Figur 69: Video - Validering af FlipFlop-----</i>	70
<i>Figur 70A: Oversigt over relæ -----</i>	71
<i>Figur 71: Kredsløbs analyse til BC337-----</i>	72
<i>Figur 72: Test af relæ -----</i>	73
<i>Figur 73: El teknisk diagram over tilslutningerne til 24V relæ-----</i>	74
<i>Figur 74: Signal på AN0 ved brug af RC filter til konvertering fra PWM til DC-----</i>	75
<i>Figur 75: El teknisk diagram over tilslutningerne i kredsløbet uden PIC (Se bilag 21 for fuld størrelse) -----</i>	76
<i>Figur 76: Flow chart over Arduino kode uden PIC (Se bilag 20 for fuld størrelse)-----</i>	77
<i>Figur 77: Budget for produktet-----</i>	79
<i>Figur 78: Billede af færdig prototype -----</i>	81
<i>Figur 79: Video - Test af magnetventil-----</i>	82
<i>Figur 80: Video - Test af pneumatik -----</i>	82
<i>Figur 81: Video - Test af stepper motor (funktionel) -----</i>	82
<i>Figur 82: Video - Test af stepper motor (ikke funktionel) -----</i>	82
<i>Figur 83: Arbejdstegning af steldel type 2-----</i>	93
<i>Figur 84: Arbejdstegning steldel type 3 -----</i>	94
<i>Figur 85: Arbejdstegning steldel type 4 -----</i>	95
<i>Figur 86: Overblik over Special funktion register for pic16f873. Fundet i databladet side 14.-----</i>	96
<i>Figur 87: TXSTA register opsætning -----</i>	97
<i>Figur 88: PIE1-register opsætning-----</i>	98
<i>Figur 89: PIR1-register opsætning -----</i>	99
<i>Figur 90: RCREG-register opsætning -----</i>	101
<i>Figur 91: Funktion af TXREG og RCREG -----</i>	101
<i>Figur 92: PORTn-register opsætning -----</i>	102

<i>Figur 93: STATUS-register opsætning</i>	103
<i>Figur 94: ADCON1-register opsætning</i>	104
<i>Figur 96: Pin konfiguration af PIC16F873</i>	105
<i>Figur 95: Tabeloversigt over kondensator værdier ift. klokfrekvens</i>	105
<i>Figur 97: Opsætning af crystal for PIC mikrokontrollere</i>	105
<i>Figur 98: Teori omkring konfigurations ordet</i>	105
<i>Figur 99: Teori om konfigurations bitsne</i>	106
<i>Figur 100: PINOUT beskrivelse af PIC16F873</i>	107
<i>Figur 101: GV5-2 egenskaber</i>	108
<i>Figur 102: Ben konfiguration af ICM7555</i>	109
<i>Figur 103: Astable Operation</i>	109
<i>Figur 104: sandhedstabel og ratings</i>	109
<i>Figur 105: beregninger til Frekvens og dutycycle</i>	110
<i>Figur 106: Ben konfiguration af BC337</i>	111
<i>Figur 107: El egenskaber for BC337</i>	111
<i>Figur 108: Ben konfiguration af HEF4070</i>	112
<i>Figur 109: Funktionsdiagram af HEF4070</i>	112
<i>Figur 110: Egenskaber ift. HEF4070</i>	113
<i>Figur 111: Funktionstabell over HEF4027</i>	113
<i>Figur 112: el egenskaber for IRF540N</i>	114
<i>Figur 113: Vigtigste egenskaber</i>	114
<i>Figur 114: Instruktions sæt i assembler kode, ift. PIC16F873</i>	115
<i>Figur 115: Ledningskonfiguration af Nema-23 Stepper Motor</i>	116
<i>Figur 116: Egenskaber for Stepper motor</i>	116
<i>Figur 117: Tidsplan udarbejdet i trello.</i>	138
<i>Figur 118: Flowchart over hovedløkken på PIC'en (Fuld størrelse)</i>	139
<i>Figur 119: Flowchart over den endelige arduino kode (Fuld størrelse)</i>	140
<i>Figur 120: Diagram over arduino kontrol (Fuld størrelse)</i>	142
<i>Figur 121: Diagram over Stepper motor logik (Fuld størrelse)</i>	143
<i>Figur 122: Diagram over det fulde kredsløb (Fuldstørrelse)</i>	144

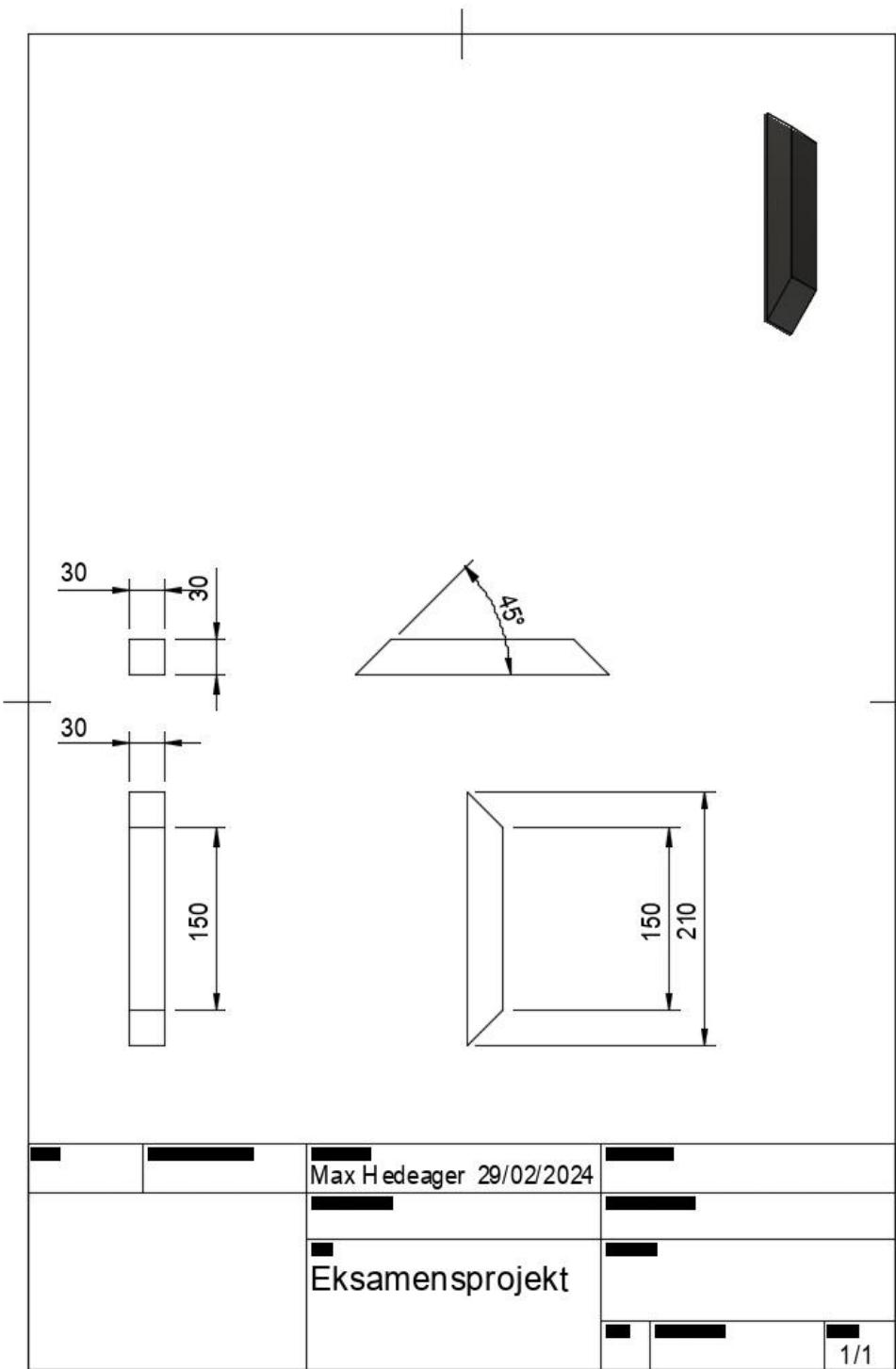
Tabel Liste

<i>Tabel 1: Tabel over opstillede krav til løsning med relevans og beskrivelse af kravets test.....</i>	7
<i>Tabel 2: Tabel over løsningsforslagenes overholdelse af de opstillede krav.....</i>	13
<i>Tabel 3: Krav til det endelige produkt.....</i>	16
<i>Tabel 4: Tabel over materialekrav</i>	18
<i>Tabel 5: Stykliste over materialer i maskin</i>	28
<i>Tabel 6: Stykliste over de elektroniske komponenter.....</i>	40
<i>Tabel 7: PINOUT description OSC1 og OSC2, pic16f87x datasheet side 7 Kilde:(PIC16F87X.pdf, u.å.).....</i>	49
<i>Tabel 8: Opfølgning på krav</i>	80

Bilag Liste

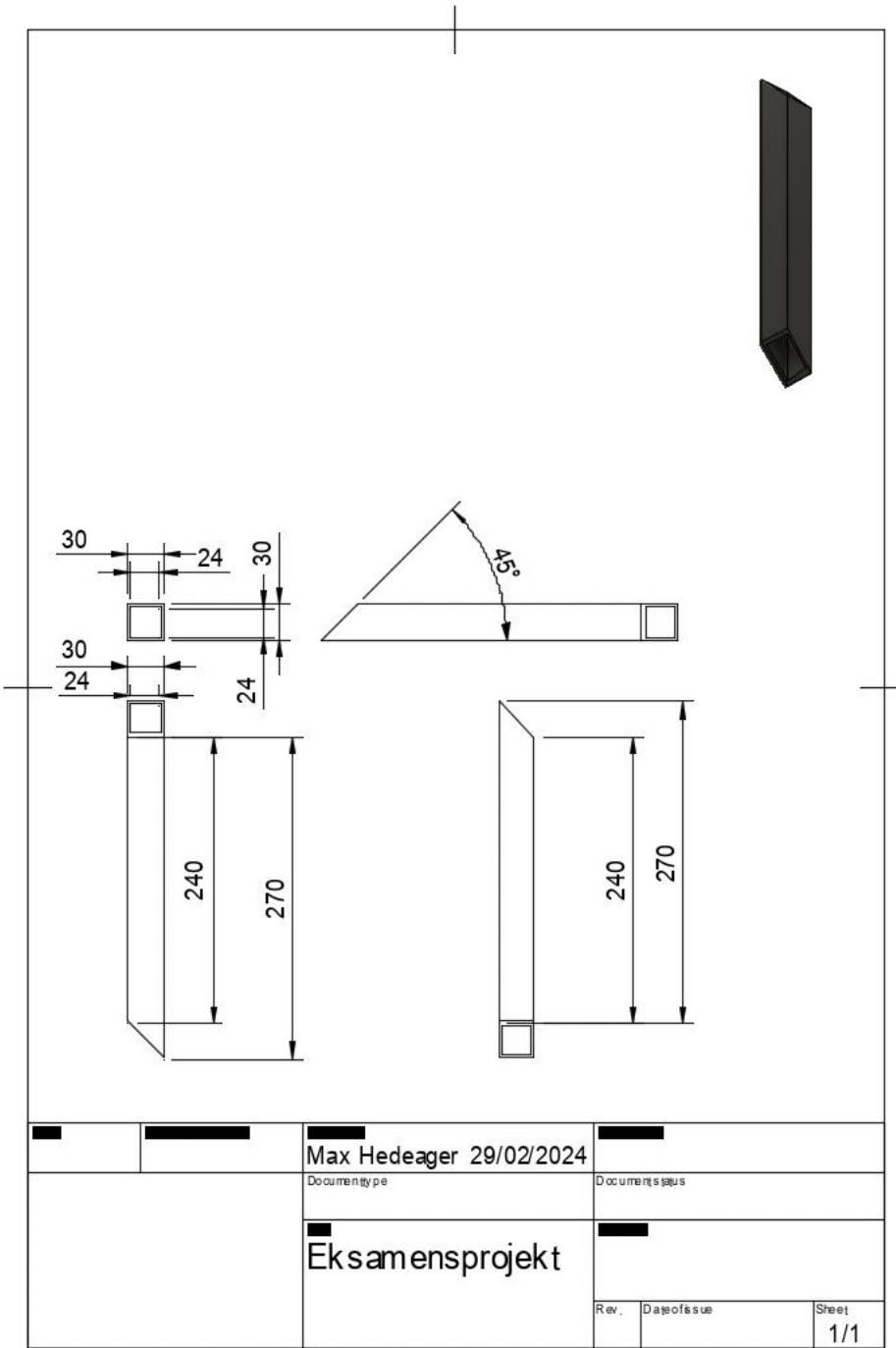
Bilag 1 – Arbejdstegninger af stel type 2	93
Bilag 2 – Arbejdstegninger af stel type 3	94
Bilag 3 – Arbejdstegninger af stel type 4	95
Bilag 4 - PIC-registre	96
Bilag 5 - PIC16F873	105
Bilag 6 - GV5-2 Datablad	108
Bilag 7 - ICM7555 DataBlad	109
Bilag 8 - BC377	111
Bilag 9 - HEF4070	112
Bilag 10 - HEF4027 FlipFlop	113
Bilag 11 - IRF540N	114
Bilag 12 - Instruktions sæt	115
Bilag 13 - Nema-23 Stepper Motor	1177
Bilag 14 - Arduino I2C skanner	117
Bilag 15 – PIC-kode	118
Bilag 16 - Arduino kode uden PIC	127
Bilag 17 - Arduino kode med PIC	134
Bilag 18 - Kodeændringer ved analog kommunikation	136
Bilag 19 - Tidsplan	138
Bilag 20 - Flow Charts i fuld størrelse	139
Bilag 21 - Diagrammer i Fuld størrelse	141

Bilag 1 – Arbejdstegninger af stel type 2



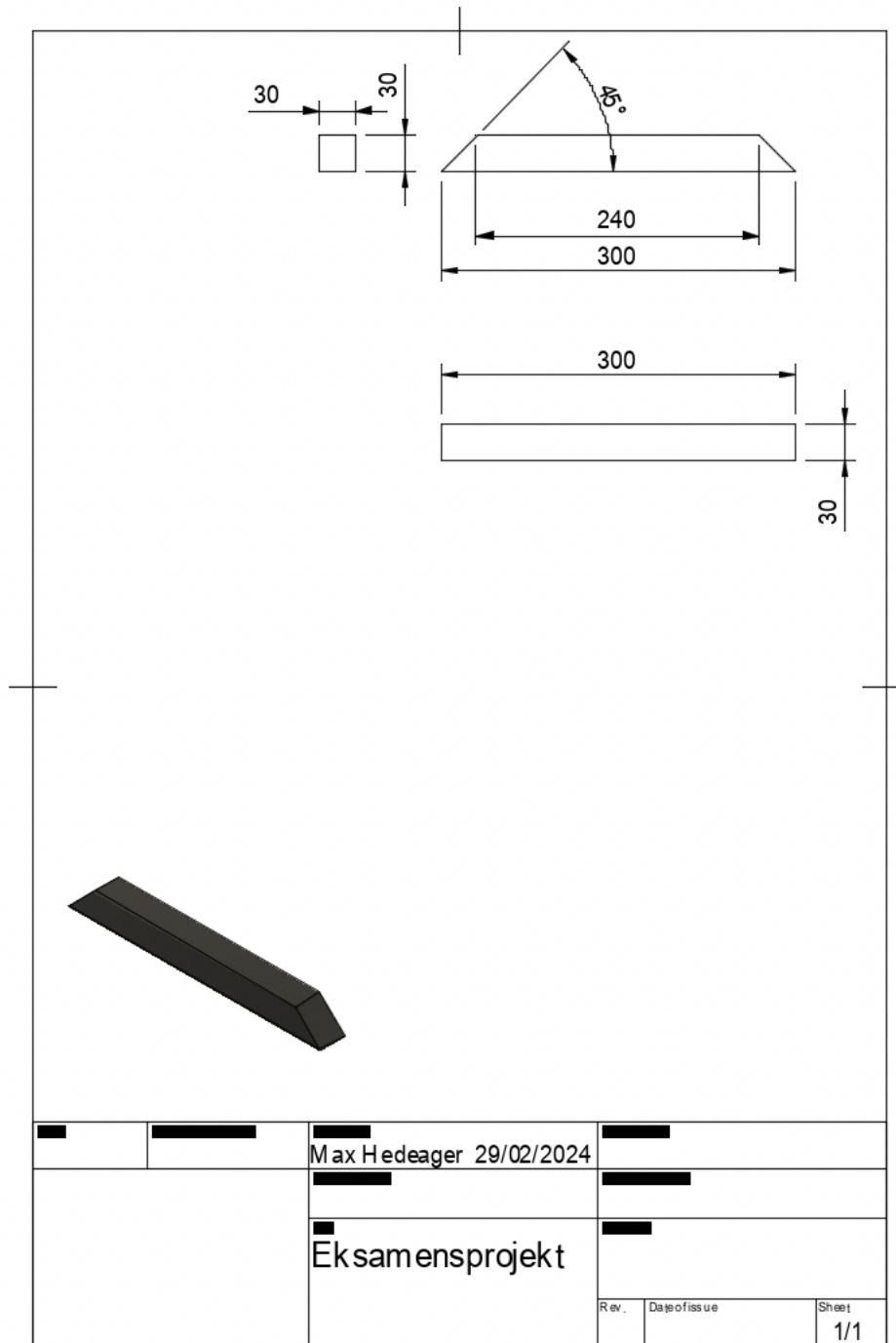
Figur 83: Arbejdstegning af steldel type 2

Bilag 2 – Arbejdstegninger af stel type 3



Figur 84: Arbejdstegning steldel type 3

Bilag 3 – Arbejdstegninger af stel type 4



Figur 85: Arbejdstegning steldel type 4

Bilag 4 - PIC-registre

FIGURE 2-4: PIC16F874/873 REGISTER FILE MAP

File Address	File Address	File Address	File Address
Indirect addr.(*) 00h	Indirect addr.(*) 80h	Indirect addr.(*) 100h	Indirect addr.(*) 180h
TMR0 01h	OPTION_REG 81h	TMR0 101h	OPTION_REG 181h
PCL 02h	PCL 82h	PCL 102h	PCL 182h
STATUS 03h	STATUS 83h	STATUS 103h	STATUS 183h
FSR 04h	FSR 84h	FSR 104h	FSR 184h
PORTA 05h	TRISA 85h	PORTB 105h	TRISB 185h
PORTB 06h	TRISB 86h	PORTB 106h	TRISB 186h
PORTC 07h	TRISC 87h	PORTB 107h	TRISB 187h
PORTD ⁽¹⁾ 08h	TRISD ⁽¹⁾ 88h	PORTB 108h	TRISB 188h
PORTE ⁽¹⁾ 09h	TRISE ⁽¹⁾ 89h	PORTB 109h	TRISB 189h
PCLATH 0Ah	PCLATH 8Ah	PCLATH 10Ah	PCLATH 18Ah
INTCON 0Bh	INTCON 8Bh	INTCON 10Bh	INTCON 18Bh
PIR1 0Ch	PIE1 8Ch	EEDATA 10Ch	EECON1 18Ch
PIR2 0Dh	PIE2 8Dh	EEADR 10Dh	EECON2 18Dh
TMR1L 0Eh	PCON 8Eh	EEDATH 10Eh	Reserved ⁽²⁾ 18Eh
TMR1H 0Fh		EEADRH 10Fh	Reserved ⁽²⁾ 18Fh
T1CON 10h		90h	110h
TMR2 11h	SSPCON2 91h	91h	
T2CON 12h	PR2 92h	92h	
SSPBUF 13h	SSPADD 93h	93h	
SSPSTAT 14h	SSPSTAT 94h	94h	
CCPR1L 15h		95h	
CCPR1H 16h		96h	
CCP1CON 17h		97h	
RCSTA 18h	TXSTA 98h	98h	
TXREG 19h	SPBRG 99h	99h	
RCREG 1Ah		9Ah	
CCPR2L 1Bh		9Bh	
CCPR2H 1Ch		9Ch	
CCP2CON 1Dh		9Dh	
ADRESH 1Eh	ADRESL 9Eh	9Eh	
ADCON0 1Fh	ADCON1 9Fh	9Fh	
General Purpose Register 96 Bytes	General Purpose Register 96 Bytes	accesses 20h-7Fh	accesses A0h - FFh
Bank 0 7Fh	Bank 1 FFh	Bank 2	Bank 3
		120h	1A0h
		16Fh	1EFh
		170h	1F0h
		17Fh	1FFh

Unimplemented data memory locations, read as '0'.

* Not a physical register.

Note 1: These registers are not implemented on the PIC16F873.

Note 2: These registers are reserved, maintain these registers clear.

Figur 86: Overblik over Special funktion register for pic16f873. Fundet i databladet side 14.

TXSTA (Transmit Status and Control Register) 98h

This register controls USART transmit function.

7(0)	6(0)	5(0)	4(0)	3(0)	2(0)	1(1)	0(0)
CSRC	TX9	TXEN	SYNC	-	BRGH	TRMT	TX9D

The value in the parenthesis is in the condition immediately after the turning on.

CSRC : Clock Source Select bit

Asynchronous mode
Don't care

Synchronous mode

- 1 : Master mode (Clock generated internally from BRG)
- 0 : Slave mode (Clock from external source)

TX9 : 9-bit Transmit Enable bit

- 1 : Selects 9-bit transmission
- 0 : Selects 8-bit transmission

TXEN : Transmit Enable bit

- 1 : Transmit enable
- 0 : Transmit disable

SREN/CREN of [RCSTA register](#) overrides TXEN in SYNC mode

SYNC : USART Mode Select bit

Asynchronous mode
1 : Synchronous mode
0 : Asynchronous mode

BRGH : High Baud Rate Select bit

Asynchronous mode
1 : High speed
0 : Low speed

Synchronous mode

Unused in this mode

TRMT : Transmit Shift Register Status bit (Read only)

- 1 : TSR empty
- 0 : TSR full

TX9D : 9th bit of transmit data. Can be parity bit.

Figur 87: TXSTA register opsætning

🟡 PIE1 (Peripheral Interrupt Enable register) 8Ch

This register contains the enable bits for various peripheral interrupts.

7(0)	6(0)	5(0)	4(0)	3(0)	2(0)	1(0)	0(0)	
PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	The value in the parenthesis is in the condition immediately after the turning on.

● PSPIE : Parallel Slave Port Read/Write Interrupt Enable bit (Not used)

● ADIE : A/D Converter Interrupt Enable bit

- 1 : Enables the A/D converter interrupt
- 0 : Disables the A/D converter interrupt

● RCIE : USART Receive Interrupt Enable bit

- 1 : Enable the USART receive interrupt
- 0 : Disable the USART receive interrupt

● TXIE : USART Transmit Interrupt Enable bit

- 1 : Enable the USART transmit interrupt
- 0 : Disable the USART transmit interrupt

● SSPIE : Synchronous Serial Port(SSP) Interrupt Enable bit

- 1 : Enable the SSP interrupt
- 0 : Disable the SSP interrupt

● CCP1IE : CCP1 Interrupt Enable bit

- 1 : Enable the CCP1 interrupt
- 0 : Disable the CCP1 interrupt

● TMR2IE : TMR2 to PR2 Match Interrupt Enable bit

- 1 : Enable the TMR2 to PR2 match interrupt
- 0 : Disable the TMR2 to PR2 match interrupt

● TMR1IE : TMR1 Overflow Interrupt Enable bit

- 1 : Enable the TMR1 overflow interrupt
- 0 : Disable the TMR1 overflow interrupt

Figur 88: PIE1-register opsætning

🟡 PIR1 (Peripheral Interrupt register) 0Ch

This register contains various peripheral interrupts.

7(0)	6(0)	5(0)	4(0)	3(0)	2(0)	1(0)	0(0)
PSP1F	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF

The value in the parenthesis is in the condition immediately after the turning on.

● PSPIF : Parallel Slave Port Read/Write Interrupt Flag (Not used)

● ADIF : A/D Converter Interrupt Flag

- 1 : A/D conversion completed
- 0 : A/D conversion is not complete

● RCIF : USART Receive Interrupt Flag

- 1 : USART receive buffer is full
- 0 : USART receive buffer is empty

● TXIF : USART Transmit Interrupt Flag

- 1 : USART transmit buffer is empty
- 0 : USART transmit buffer is full

● SSPIF : Synchronous Serial Port(SSP) Interrupt Flag

- 1 : SSP interrupt condition has occurred (must be cleared in software)
- 0 : No SSP interrupt condition occurred

● CCP1IF : CCP1 Interrupt Flag

Capture Mode

- 1 : TMR1 register capture occurred (must be cleared in software)
- 0 : No TMR1 register capture occurred

Compare Mode

- 1 : TMR1 register compare match occurred (must be cleared in software)
- 0 : No TMR1 register compare match occurred

PWM Mode

Unused in this mode

● TMR2IF : TMR2 to PR2 Match Interrupt Flag

- 1 : TMR2 to PR2 match occurred (must be cleared in software)
- 0 : No TMR2 to PR2 match occurred

● TMR1IF : TMR1 Overflow Interrupt Flag

- 1 : TMR1 register overflowed (must be cleared in software)
- 0 : TMR1 register did not overflow

Figur 89: PIR1-register opsætning

🟡 RCSTA (Receive Status and Control Register) 18h

This register controls USART receive function.

7(0)	6(0)	5(0)	4(0)	3(0)	2(0)	1(0)	0(x)
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D

The value in the parenthesis is in the condition immediately after the turning on.

● SPEN : Serial Port Enable bit

- 1 : Serial port enabled
(Configures RC7/RX/DT and RC6/TX/CK pins as serial port pins)
- 0 : Serial port disabled

● RX9 : 9-bit Receive Enable bit

- 1 : Selects 9-bit reception
- 0 : Selects 8-bit reception

● SREN : Single Receive Enable bit

Asynchronous mode
Don't care

Synchronous mode - master
1 : Enables single receive This bit is cleared after reception is complete.
0 : Disables single receive

Synchronous mode - slave
Unused in this mode

● CREN : Continuous Receive Enable bit

Asynchronous mode
1 : Enables continuous receive
0 : Disables continuous receive

Synchronous mode
1 : Enables continuous receive until enable bit CREN is cleared
(CREN is overrides SREN)
0 : Disables continuous receive

● ADDEN : Address Detect Enable bit

Asynchronous mode 9-bit (RX9 = 1)
1 : Enables address detection
enable interrupt and load of the receive buffer when RSR<8> is set
0 : Disables address detection
all bytes are received, and ninth bit can be used as parity bit

● FERR : Framing Error bit (Read only)

- 1 : Framing error
(Can be updated by reading RCREG register and receive next valid byte)
- 0 : No framing error

● OERR : Overrun Error bit (Read only)

- 1 : Overrun error
(Can be cleared by clearing bit CREN)
- 0 : No overrun error

● RX9D : 9th bit of receive data (Read only)

Figur 90: RCREG-register opsætning

🟡 TXREG (Transmit buffer) 19h

Transmit data is set to this buffer.

🟡 RCREG (Receive buffer) 1Ah

Received data is stored to this buffer.

Figur 91: Funktion af TXREG og RCREG



Figur 92: PORTn-register opsætning

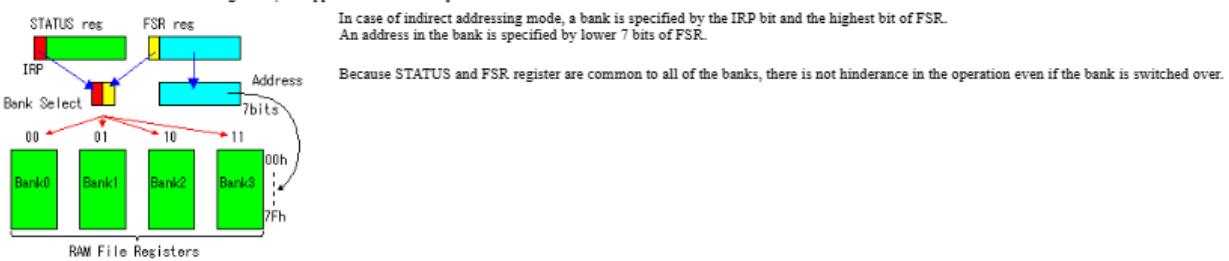
● STATUS (Status register) 03h,83h,103h,183h

By this register, the bank specification of the RAM file register, the time-out condition of the watchdog timer, the power down condition, the flag of the calculation result and so on, are managed.

7(0) 6(0) 5(0) 4(1) 3(1) 2(x) 1(x) 0(x)
 IRP | RP1 | RP0 | TO | PD | Z | DC | C

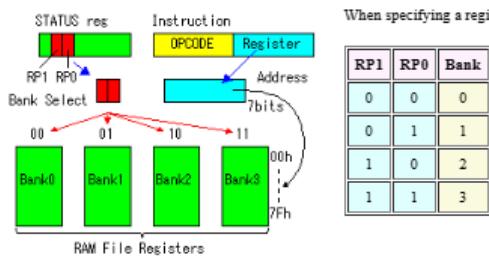
The value in the parenthesis is in the condition immediately after the turning on.

● IRP : In case of indirect addressing mode, the upper bit of the bank specification is set.



● RP1,RP0 : The bank specification is set by these bits in case of direct addressing mode.

When specifying a register directly by an instruction, the bank is specified by RP0 and RP1.



● TO(inv) : The time-out condition of the watchdog timer is set. It isn't possible to do writing.

- 1 : After power-on, CLRWDT instruction, or SLEEP instruction
- 0 : Watchdog time-out occurred

● PD(inv) : A power down(power saving) condition is displayed. It isn't possible to do writing.

- 1 : After power-on or by the CLRWDT instruction
- 0 : By execution of the SLEEP instruction

● Z : Zero bit

- 1 : The result of an arithmetic or logic operation is zero
- 0 : The result of an arithmetic or logic operation is not zero

● DC : Digit carry/borrow(invert) bit (for ADDWF and ADDLW)

- 1 : A carry-out from the 4th low order bit of the result occurred
- 0 : No carry-out from the 4th low order bit of the result

● C : Carry/borrow(invert) bit (for ADDWF and ADDLW)

- 1 : A carry-out from the most significant bit of the result occurred
- 0 : No carry-out from the most significant bit of the result occurred

Figur 93: STATUS-register opsætning

🟡 ADCON1 (A/D converter control register 1) 9Fh

The A/D converter condition is controlled with this register.

7(0)	6(0)	5(0)	4(0)	3(0)	2(0)	1(0)	0(0)
ADFM	-	-	-	PCFG3	PCFG2	PCFG1	PCFG0

The value in the parenthesis is in the condition immediately after the turning on.

● ADFM : A/D result format select

- 1 : Right justified. 6 most significant bits of ADRESH are read as 0.
- 0 : Left justified. 6 least significant bits of ADRESL are read as 0.

● PCFG3,PCFG2,PCFG1,PCFG0 : A/D port configuration control bits

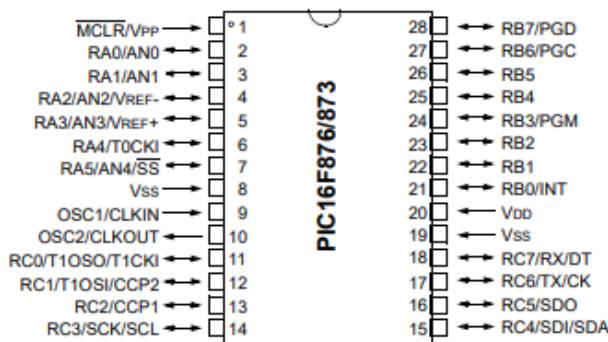
PCFG 3210	Port					Remarks
	AN4	AN3	AN2	AN1	AN0	
0000	A	A	A	A	A	
0001	A	V _{REF+}	A	A	A	
0010	A	A	A	A	A	Same as 0000
0011	A	V _{REF+}	A	A	A	Same as 0001
0100	D	A	D	A	A	
0101	D	V _{REF+}	D	A	A	
0110	D	D	D	D	D	
0111	D	D	D	D	D	Same as 0110
1000	A	V _{REF+}	V _{REF-}	A	A	
1001	A	A	A	A	A	Same as 0000
1010	A	V _{REF+}	A	A	A	Same as 0001
1011	A	V _{REF+}	V _{REF-}	A	A	Same as 1000
1100	A	V _{REF+}	V _{REF-}	A	A	Same as 1000
1101	D	V _{REF+}	V _{REF-}	A	A	
1110	D	D	D	D	A	
1111	D	V _{REF+}	V _{REF-}	D	A	

A: Analog port
D: Digital port
V_{REF+}: High reference voltage
V_{REF-}: Low reference voltage

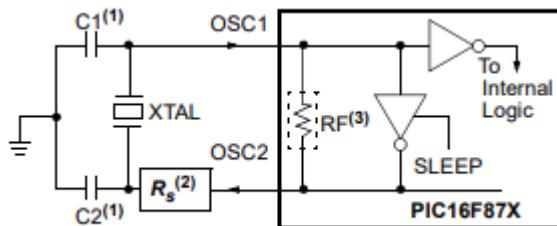
Figur 94: ADCON1-register opsætning

Bilag 5 - PIC16F873

PDIP, SOIC



Figur 96: Pin konfiguration af PIC16F873



Figur 97: Opsætning af crystal for PIC mikrokontrollere

TABLE 12-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

Osc Type	Crystal Freq.	Cap. Range C1	Cap. Range C2
LP	32 kHz	33 pF	33 pF
	200 kHz	15 pF	15 pF
XT	200 kHz	47-68 pF	47-68 pF
	1 MHz	15 pF	15 pF
	4 MHz	15 pF	15 pF
	8 MHz	15-33 pF	15-33 pF
HS	4 MHz	15 pF	15 pF
	8 MHz	15-33 pF	15-33 pF
	20 MHz	15-33 pF	15-33 pF

Figur 95: Tabeloversigt over kondensator værdier ift. klokfrekvens

12.1 Configuration Bits

The configuration bits can be programmed (read as '0'), or left unprogrammed (read as '1'), to select various device configurations. The erased, or unprogrammed value of the configuration word is 3FFFh. These bits are mapped in program memory location 2007h.

It is important to note that address 2007h is beyond the user program memory space, which can be accessed only during programming.

Figur 98: Teori omkring konfigurations ordet

REGISTER 12-1: CONFIGURATION WORD (ADDRESS 2007h)⁽¹⁾

CP1	CP0	DEBUG	—	WRT	CPD	LVP	BODEN	CP1	CP0	PWRTE	WDTE	FOSC1	FOSC0
bit13													bit0
bit 13-12, CP1:CP0: FLASH Program Memory Code Protection bits ⁽²⁾													
bit 5-4 11 = Code protection off 10 = 1F00h to 1FFFh code protected (PIC16F877, 876) 10 = 0F00h to 0FFFh code protected (PIC16F874, 873) 01 = 1000h to 1FFFh code protected (PIC16F877, 876) 01 = 0800h to 0FFFh code protected (PIC16F874, 873) 00 = 0000h to 1FFFh code protected (PIC16F877, 876) 00 = 0000h to 0FFFh code protected (PIC16F874, 873)													
bit 11 DEBUG: In-Circuit Debugger Mode 1 = In-Circuit Debugger disabled, RB6 and RB7 are general purpose I/O pins 0 = In-Circuit Debugger enabled, RB6 and RB7 are dedicated to the debugger.													
bit 10 Unimplemented: Read as '1'													
bit 9 WRT: FLASH Program Memory Write Enable 1 = Unprotected program memory may be written to by EECON control 0 = Unprotected program memory may not be written to by EECON control													
bit 8 CPD: Data EE Memory Code Protection 1 = Code protection off 0 = Data EEPROM memory code protected													
bit 7 LVP: Low Voltage In-Circuit Serial Programming Enable bit 1 = RB3/PGM pin has PGM function, low voltage programming enabled 0 = RB3 is digital I/O, HV on MCLR must be used for programming													
bit 6 BODEN: Brown-out Reset Enable bit ⁽³⁾ 1 = BOR enabled 0 = BOR disabled													
bit 3 PWRTE: Power-up Timer Enable bit ⁽³⁾ 1 = PWRT disabled 0 = PWRT enabled													
bit 2 WDTE: Watchdog Timer Enable bit 1 = WDT enabled 0 = WDT disabled													
bit 1-0 FOSC1:FOSC0: Oscillator Selection bits 11 = RC oscillator 10 = HS oscillator 01 = XT oscillator 00 = LP oscillator													

Figur 99: Teori om konfigurations bitsne

TABLE 1-1: PIC16F873 AND PIC16F876 PINOUT DESCRIPTION

Pin Name	DIP Pin#	SOIC Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	9	9	I	ST/CMOS ⁽³⁾	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	10	10	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, the OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
MCLR/VPP	1	1	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
RA0/AN0	2	2	I/O	TTL	PORTA is a bi-directional I/O port. RA0 can also be analog input0.
RA1/AN1	3	3	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	4	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	5	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	6	I/O	ST	RA4 can also be the clock input to the Timer0 module. Output is open drain type.
RA5/SS/AN4	7	7	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
RB0/INT	21	21	I/O	TTL/ST ⁽¹⁾	PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs. RB0 can also be the external interrupt pin.
RB1	22	22	I/O	TTL	
RB2	23	23	I/O	TTL	
RB3/PGM	24	24	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	25	25	I/O	TTL	Interrupt-on-change pin.
RB5	26	26	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	27	27	I/O	TTL/ST ⁽²⁾	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	28	28	I/O	TTL/ST ⁽²⁾	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.
RC0/T1OSO/T1CKI	11	11	I/O	ST	PORTC is a bi-directional I/O port. RC0 can also be the Timer1 oscillator output or Timer1 clock input.
RC1/T1OSI/CCP2	12	12	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	13	13	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	14	14	I/O	ST	RC3 can also be the synchronous serial clock input/output for both SPI and I ² C modes.
RC4/SDI/SDA	15	15	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I ² C mode).
RC5/SDO	16	16	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	17	17	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	18	18	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
Vss	8, 19	8, 19	P	—	Ground reference for logic and I/O pins.
Vdd	20	20	P	—	Positive supply for logic and I/O pins.

Figur 100: PINOUT beskrivelse af PIC16F873

Bilag 6 - GV5-2 Datablad

■ Coil Ratings

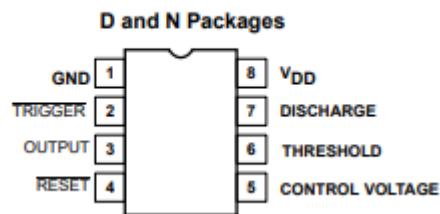
Standard Models

Rated voltage	3 VDC	5 VDC	6 VDC	9 VDC	12 VDC	24 VDC	48 VDC
Rated current	166.7 mA	100 mA	83.3 mA	55.6 mA	41.7 mA	20.8 mA	12 mA
Coil resistance	18 Ω	50 Ω	72 Ω	162 Ω	288 Ω	1,152 Ω	4,000 Ω
Coil inductance	Armature OFF	0.04	0.09	0.16	0.31	0.47	1.98
(H) (ref. value)	Armature ON	0.05	0.11	0.19	0.49	0.74	2.63
Must operate voltage	75% max. of rated voltage						
Must release voltage	5% min. of rated voltage						
Max. voltage	120% of rated voltage at 23°C						
Power consumption	Approx. 500 mW						Approx. 580 mW

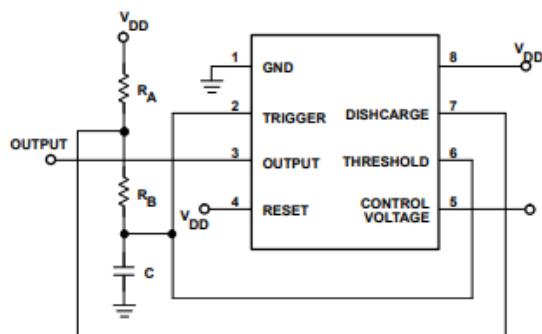
Figur 101: GV5-2 egenskaber

Bilag 7 - ICM7555 DataBlad

PIN CONFIGURATION



Figur 102: Ben konfiguration af ICM7555



Figur 103: Astable Operation

TRUTH TABLE

THRESHOLD VOLTAGE	TRIGGER VOLTAGE	RESET ¹	OUTPUT	DISCHARGE SWITCH
DON'T CARE	DON'T CARE	LOW	LOW	ON
>2/3(V+)	> 1/3(V+)	HIGH	LOW	ON
$V_{TH} < 2/3$	$V_{TR} > 1/3$	HIGH	STABLE	STABLE
DON'T CARE	<1/3(V+)	HIGH	HIGH	OFF

NOTES:

- NOTES:** 1. RESET will dominate all other inputs: TRIGGER will dominate over THRESHOLD.

ABSOLUTE MAXIMUM RATINGS¹

SYMBOL	PARAMETER	RATING	UNITS
V_{DD}	Supply voltage	+18	V
V_{TRIG}^1	Trigger input voltage		
V_{CV}	Control voltage	> -0.3 to	
V_{TH}	Threshold input voltage	$<V_{DD} + 0.3$	V
V_{RST}	RESET input voltage		
I_{OUT}	Output current	100	mA
P_{DMAX}	Maximum power dissipation, $T_A = 25^\circ\text{C}$ (still air) ²		
	N package	1160	mW
	D package	780	mW
T_{STG}	Storage temperature range	-65 to +150	$^\circ\text{C}$
T_{SOLD}	Lead temperature (Soldering 60s)	300	$^\circ\text{C}$

Figur 104: sandhedstabel og ratings

$$F = \frac{1.38}{(R_A + 2R_B) C} \quad D = \frac{R_A + R_B}{R_A + 2R_B}$$

Figur 105: beregninger til Frekvens og dutycycle

Bilag 8 - BC377



Figur 106: Ben konfiguration af BC337

Electrical Characteristics $T_a=25^\circ\text{C}$ unless otherwise noted

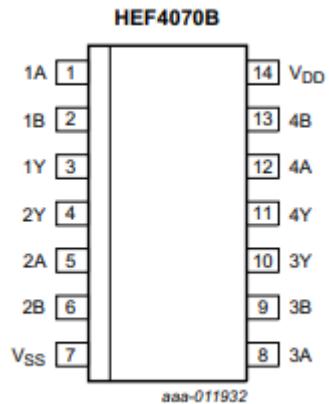
Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Units
BV_{CEO}	Collector-Emitter Breakdown Voltage : BC337	$I_C=10\text{mA}, I_B=0$	45			V
	: BC338					
BV_{CES}	Collector-Emitter Breakdown Voltage : BC337	$I_C=0.1\text{mA}, V_{BE}=0$	50			V
	: BC338					
BV_{EBO}	Emitter-Base Breakdown Voltage	$I_E=0.1\text{mA}, I_C=0$	5			V
I_{CES}	Collector Cut-off Current : BC337	$V_{CE}=45\text{V}, I_B=0$	2	100	nA	nA
	: BC338	$V_{CE}=25\text{V}, I_B=0$				
h_{FE1} h_{FE2}	DC Current Gain	$V_{CE}=1\text{V}, I_C=100\text{mA}$ $V_{CE}=1\text{V}, I_C=300\text{mA}$	100		630	
$V_{CE}(\text{sat})$	Collector-Emitter Saturation Voltage	$I_C=500\text{mA}, I_B=50\text{mA}$			0.7	V
$V_{BE}(\text{on})$	Base Emitter On Voltage	$V_{CE}=1\text{V}, I_C=300\text{mA}$			1.2	V
f_T	Current Gain Bandwidth Product	$V_{CE}=5\text{V}, I_C=10\text{mA}, f=50\text{MHz}$		100		MHz
C_{ob}	Output Capacitance	$V_{CE}=10\text{V}, I_E=0, f=1\text{MHz}$		12		pF

h_{FE} Classification

Classification	16	25	40
h_{FE1}	100 ~ 250	160 ~ 400	250 ~ 630
h_{FE2}	60-	100-	170-

Figur 107: El egenskaber for BC337

Bilag 9 - HEF4070



Figur 108: Ben konfiguration af HEF4070

5. Functional diagram

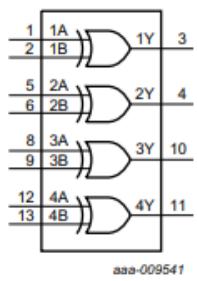


Fig 1. Functional diagram

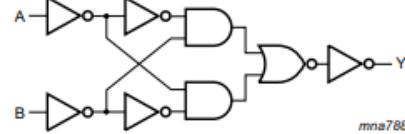


Fig 2. Logic diagram (one gate)

Figur 109: Funktionsdiagramm af HEF4070

8. Limiting values

Table 4. Limiting values

In accordance with the Absolute Maximum Rating System (IEC 60134). Voltages are referenced to $V_{SS} = 0 \text{ V}$ (ground).

Symbol	Parameter	Conditions	Min	Max	Unit
V_{DD}	supply voltage		-0.5	+18	V
I_{IK}	input clamping current	$V_I < -0.5 \text{ V}$ or $V_I > V_{DD} + 0.5 \text{ V}$	-	± 10	mA
V_I	input voltage		-0.5	$V_{DD} + 0.5$	V
I_{OK}	output clamping current	$V_O < -0.5 \text{ V}$ or $V_O > V_{DD} + 0.5 \text{ V}$	-	± 10	mA
$I_{I/O}$	input/output current		-	± 10	mA
I_{DD}	supply current		-	50	mA
T_{stg}	storage temperature		-65	+150	$^{\circ}\text{C}$
T_{amb}	ambient temperature		-40	+85	$^{\circ}\text{C}$
P_{tot}	total power dissipation	$T_{amb} = -40 \text{ }^{\circ}\text{C}$ to $+85 \text{ }^{\circ}\text{C}$			
		SO14	[1]	-	500 mW
P	power dissipation	per output	-	100	mW

Figur 110: Egenskaber ift. HEF4070

Bilag 10 - HEF4027 FlipFlop

FUNCTION TABLES

INPUTS				OUTPUTS		
S_D	C_D	CP	J	K	O	\bar{O}
H	L	X	X	X	H	L
L	H	X	X	X	L	H
H	H	X	X	X	H	H

INPUTS					OUTPUTS	
S_D	C_D	CP	J	K	O_{n+1}	\bar{O}_{n+1}
L L / L L					no change	
L L / H L			H L		H	L
L L / L H			L H		L	H
L L / H H			H H		\bar{O}_n	O_n

Figur 111: Funktionstabell over HEF4027

Bilag 11 - IRF540N

ELECTRICAL CHARACTERISTICS

$T_j = 25^\circ\text{C}$ unless otherwise specified

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
$V_{(\text{BR})\text{DSS}}$	Drain-source breakdown voltage	$V_{GS} = 0 \text{ V}; I_D = 0.25 \text{ mA}$	100	-	-	V
$V_{GS(\text{TO})}$	Gate threshold voltage	$V_{DS} = V_{GS}; I_D = 1 \text{ mA}$	89	-	-	V
$R_{DS(\text{ON})}$	Drain-source on-state resistance	$V_{GS} = 10 \text{ V}; I_D = 17 \text{ A}$	2	3	4	$\text{m}\Omega$
g_f	Forward transconductance	$V_{DS} = 25 \text{ V}; I_D = 17 \text{ A}$	1	-	-	V
I_{GSS}	Gate source leakage current	$V_{GS} = \pm 20 \text{ V}; V_{DS} = 0 \text{ V}$	-	-	6	V
I_{DSS}	Zero gate voltage drain current	$V_{DS} = 100 \text{ V}; V_{GS} = 0 \text{ V}$	-	49	77	nA
		$V_{DS} = 80 \text{ V}; V_{GS} = 0 \text{ V}; T_j = 175^\circ\text{C}$	-	132	193	μA
			8.7	15.5	-	S
			-	10	100	μA
			-	0.05	10	μA
			-	-	250	μA
$Q_{g(\text{tot})}$	Total gate charge	$I_D = 17 \text{ A}; V_{DD} = 80 \text{ V}; V_{GS} = 10 \text{ V}$	-	-	65	nC
Q_{gs}	Gate-source charge		-	-	10	nC
Q_{gd}	Gate-drain (Miller) charge		-	-	29	nC
$t_{d\text{ on}}$	Turn-on delay time	$V_{DD} = 50 \text{ V}; R_D = 2.2 \Omega;$	-	8	-	ns
t_r	Turn-on rise time	$V_{GS} = 10 \text{ V}; R_G = 5.6 \Omega$	-	39	-	ns
$t_{d\text{ off}}$	Turn-off delay time	Resistive load	-	26	-	ns
t_f	Turn-off fall time		-	24	-	ns
L_d	Internal drain inductance	Measured tab to centre of die	-	3.5	-	nH
L_d	Internal drain inductance	Measured from drain lead to centre of die (SOT78 package only)	-	4.5	-	nH
L_s	Internal source inductance	Measured from source lead to source bond pad	-	7.5	-	nH
C_{iss}	Input capacitance	$V_{GS} = 0 \text{ V}; V_{DS} = 25 \text{ V}; f = 1 \text{ MHz}$	-	890	1187	pF
C_{oss}	Output capacitance		-	139	167	pF
C_{rss}	Feedback capacitance		-	83	109	pF

Figur 112: el egenskaber for IRF540N

QUICK REFERENCE DATA

$$V_{DSS} = 100 \text{ V}$$

$$I_D = 23 \text{ A}$$

$$R_{DS(\text{ON})} \leq 77 \text{ m}\Omega$$

Figur 113: Vigtigste egenskaber

Bilag 12 - Instruktions sæt

TABLE 13-2: PIC16F87X INSTRUCTION SET

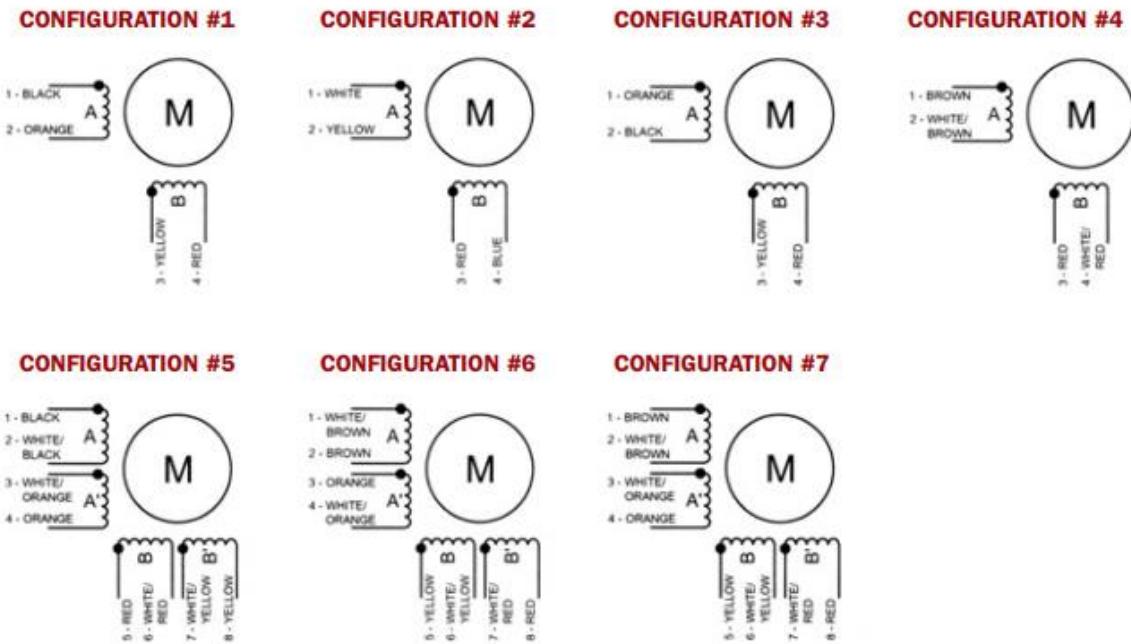
Mnemonic, Operands	Description	Cycles	14-Bit Opcode		Status Affected	Notes
			MSb	LSb		
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 lfff ffff	Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff	Z	1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff	Z	1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000 lfff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff	Z	1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff		1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff		1,2
BTFSZ	f, b	Bit Test f, Skip if Clear	1(2)	01 10bb bfff ffff		3
BTFSZ	f, b	Bit Test f, Skip if Set	1(2)	01 11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk	Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk		
CLRWD	-	Clear Watchdog Timer	1	00 0000 0110 0100	TO,PD	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk		
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk	Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk		
RETFIE	-	Return from interrupt	2	00 0000 0000 1001		
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk		
RETURN	-	Return from Subroutine	2	00 0000 0000 1000	TO,PD	
SLEEP	-	Go into standby mode	1	00 0000 0110 0011		
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk	Z	

- Note 1:** When an I/O register is modified as a function of itself (e.g., MOVF PORTB, 1), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.
- 2:** If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 module.
- 3:** If Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

Figur 114: Instruktions sæt i assembler kode, ift. PIC16F873

Bilag 13 - Nema-23 Stepper Motor

STANDARD LEADWIRE CONFIGURATION



Figur 115: Ledningskonfiguration af Nema-23 Stepper Motor

Specification	Units	S 20 2216		S 20 2221			S 20 2231	
		0100	0210	0100	0210	0300	0210	0300
Rated Phase Current ⁽¹⁾	A	1.00	2.10	1.00	2.10	3.00	2.10	3.00
Phase Resistance ⁽¹⁾	Ω	4.6	1.0	6.2	1.4	0.7	2.0	1.1
Phase Inductance ⁽¹⁾	mH	4.6	2.1	8.8	3.9	0.9	6.5	1.7
Holding Torque Unipolar	oz-in	52	—	106	—	106	—	177
	Ncm	37	—	75	—	75	—	125
Holding Torque Bipolar	oz-in	67	67	139	139	139	228	231
	Ncm	47	47	98	98	98	161	163
Detent Torque	oz-in	3.0	3.0	5.7	5.7	5.7	9.6	9.6
	Ncm	2.1	2.1	4.0	4.0	4.0	6.8	6.8
Rotor Inertia	$oz\text{-in}^2 \times 10^4$	11	11	31	31	31	48	48
	$g\text{-cm}^2$	77	77	220	220	220	340	340
Motor Weight (Mass)	lb	1.1	1.1	1.5	1.5	1.5	2.2	2.2
	kg	0.50	0.50	0.70	0.70	0.70	1.0	1.0
Maximum Voltage	V	75	75	75	75	75	75	75
Motor Length (Max)	in	1.61	1.61	2.17	2.17	2.17	3.09	3.09
	mm	41	41	55	55	55	78.5	78.5
Std. Leadwire Config. ⁽²⁾	—	6	4	7	4	7	4	7
Std. No. of Leads	—	8	4	8	4	8	4	8

Figur 116: Egenskaber for Stepper motor

Bilag 14 - Arduino I2C skanner

```
#include <Wire.h>
void setup() {
    Wire.begin();
    Serial.begin(115200);
    Serial.println("\nI2C Scanner");
}
void loop() {
    byte error, address;
    int nDevices;
    Serial.println("Scanning...");
    nDevices = 0;
    for(address = 1; address < 127; address++ ) {
        Wire.beginTransmission(address);
        error = Wire.endTransmission();
        if (error == 0) {
            Serial.print("I2C device found at address 0x");
            if (address<16) {
                Serial.print("0");
            }
            Serial.println(address,HEX);
            nDevices++;
        }
        else if (error==4) {
            Serial.print("Unknow error at address 0x");
            if (address<16) {
                Serial.print("0");
            }
            Serial.println(address,HEX);
        }
    }
    if (nDevices == 0) {
        Serial.println("No I2C devices found\n");
    }
    else {
        Serial.println("done\n");
    }
    delay(5000);
}
```

Bilag 15 – PIC-kode

```
001: ;*****
002: ;
003: ;      Drinks maskine
004: ;
005: ;      Device : PIC16F873
006: ;      Author : Jacob Jørgensen
007: ;*****
008:     list      p=PIC16F873
009:     include    p16f873.inc
010:     __config _HS_OSC & _WDT_OFF & _PWRTE_ON & _CP_OFF & _LVP_OFF & _DEBUG_OFF
011:
012:
013: receivedData      equ 20h      ;Variable for index lookup table
014: goToStart         equ 21h ;variable for number of times needed to go to start
015: d1                equ 22h      ;Delay variable
016: d2                equ 23h      ;Delay variable
017: d3                equ 24h      ;Delay variable
018: d4                equ 25h      ;Delay variable
019:
020:
021:
022: ;***** Constants *****
023: activationPin    equ RA1 ; Pin 3 PORTA,1
024: directionPin    equ RA2 ; Pin 4 PORTA,2
025: GinValve        equ RB0 ; Pin 21
026: RomValve        equ RB1 ; Pin 22
027: VodkaValve      equ RB2 ; Pin 23
028: MangoValve      equ RB3 ; Pin 24
029: RåstofValve      equ RB4 ; Pin 25
030: AppelsinValve   equ RB5 ; Pin 26
031: LemonValve      equ RB6 ; Pin 27
032: ColaValve       equ RB7 ; Pin 28
033:
034:
035:
036:
037:
038:
039:
040: ;***** Bank macro *****
041: bank1 macro
042:     bsf STATUS, RP0
043:     endm
044: bank0 macro
```

```
045:      bcf STATUS, RP0
046:      endm
047:
048:      org 0x00
049:      goto init
050:
051: ;Initialisering
052: init
053:      bank1
054:      movlw B'10000000'
055:      movwf TRISC      ; Set TRISC 7 til input (receive) og 6 til output (transmit)
056:      movlw B'00000000'
057:      movwf TRISB      ; Set all PORTB pins as output for magnetic valves
058:      movlw B'000000'    ; Set RA0 as output for stepper motor activation and RA1 as output for
direction
059:      movwf TRISA
060:      movlw B'00001110' ; Set RA1 to 4 to digital and leave RA0 as analog
061:      movwf ADCON1
062:      movlw D'10'        ; Baud rate = 115200bps
063:      movwf SPBRG
064:      movlw B'00100100'  ; TXEN=1, BRGH=1 - TX enable, high bitrate
065:      movwf TXSTA
066:      bank0
067:      movlw B'10000000'
068:      movwf RCSTA      ; SPEN=1 - Enables serial ports
069:      clrf PORTB      ; Clear PortB
070:      clrf PORTA      ; Clear PortB
071:
072:
073: ;Main loop
074: Main
075:      clrf TXREG      ; Clear confirmation byte sent to Arduino, so it knows the drink is done.
076:      bsf PORTB, RB3
077:      call receiveData ; Call receive function to check for incoming data
078:      call sendConfirmation ; Send confirmation back to Arduino
079:
080:      ; Determine which function to call based on the received data
081:      movf receivedData, W ; Move the received data into W register
082:      addlw -1           ; Subtract 1 from W register (since array index starts from 0)
083:      btfsc STATUS, Z   ; If Zero flag is set (i.e., receivedData was 1), call GinHass
084:      call GinHass
085:      addlw -1           ; Subtract 1 from W register
086:      btfsc STATUS, Z   ; If Zero flag is set (i.e., receivedData was 2), call GinLemon
087:      call GinLemon
088:      addlw -1           ; Subtract 1 from W register
089:      btfsc STATUS, Z   ; If Zero flag is set (i.e., receivedData was 3), call RomOgCola
```

```
090:    call RomOgCola
091:    addlw -1      ; Subtract 1 from W register
092:    btfsc STATUS, Z ; If Zero flag is set (i.e., receivedData was 4), call VodkaLemon
093:    call VodkaLemon
094:    addlw -1      ; Subtract 1 from W register
095:    btfsc STATUS, Z ; If Zero flag is set (i.e., receivedData was 5), call VodkaJuice
096:    call VodkaJuice
097:    addlw -1      ; Subtract 1 from W register
098:    btfsc STATUS, Z ; If Zero flag is set (i.e., receivedData was 6), call VodkaCola
099:    call VodkaCola
100:    addlw -1      ; Subtract 1 from W register
101:    btfsc STATUS, Z ; If Zero flag is set (i.e., receivedData was 7), call Astronaut
102:    call Astronaut
103:    call LongIsland           ; If not any of the above receivedData was 8, call LongIsland
104:
105:    goto Main      ; Continue looping
106:
107: ; Function to receive data from Arduino
108: receiveData:
109:    btfss PIR1, RCIF   ; Check if data is available in the receive buffer
110:    goto Main      ; If not, continue looping
111:    movf RCREG, W    ; Read the received data from the receive buffer
112:    bsf PORTA, 3
113:    movwf receivedData ; Store the received data in a variable
114:    return
115:
116: ; Function to send confirmation back to Arduino
117: sendConfirmation:
118:    movlw 0x01      ; Send a confirmation byte back to Arduino
119:    movwf TXREG
120:    return
121:
122: ; Function to move the stepper one bottle position to the left
123: MoveOneBottleLeft:
124:    ; Set direction to left
125:    bcf PORTA, directionPin
126:    ; Activate the motor
127:    bsf PORTA, activationPin
128:    ; Wait for the motor to move one position
129:    call MovementDelay
130:    ; Deactivate the motor
131:    bcf PORTA, activationPin
132:    return
133:
134: ; Function to move the stepper one bottle position to the right
135: MoveOneBottleRight:
```

```
136: ; Set direction to right
137: bsf PORTA, directionPin
138: ; Activate the motor
139: bsf PORTA, activationPin
140: ; Wait for the motor to move one position
141: call MovementDelay
142: ; Deactivate the motor
143: bcf PORTA, activationPin
144: bcf PORTA, directionPin
145: return
146: ;*****Drinks functions*****
147: ; Konfigurer dem så de kalder de rigtige funktioner i række følge.
148: GinHass:
149: ;Since Gin is the start bottle no need to turn just open valve
150: bsf PORTB, GinValve ;Open valve
151: call Delay_1s ;Dispense for 1 sec
152: bcf PORTB, GinValve ;Close valve
153: ;Go to Mango by turning 3 times and open valve
154: call MoveOneBottleLeft
155: call MoveOneBottleLeft
156: call MoveOneBottleLeft
157: bsf PORTB, MangoValve ;Open valve
158: call Delay_1s ;Dispense Mango for 1 sec NÅR VI FINDER UD AF HVOR LANG TID HVER
SKAL VÆRE ÅBEN KAN VI KØRER DEM SYNKRONT OG SLUKKE FOR DEM INDIVIDUELT
159: bcf PORTB, MangoValve ;Close valve
160: ;Open Lemon valve on the side.
161: bsf PORTB, LemonValve ;Open valve
162: call Delay_1s ;Dispense Lemon for 1 sec
163: bcf PORTB, LemonValve ;Close valve
164: ;Go to start
165: movlw D'3'
166: movwf goToStart ;Set goToStart to 3 meaing it needs to turn 3 times to the right to return to
start (Gin bottle)
167: call MoveOneBottleRight
168: decfsz goToStart, 1 ;Minus 1 from the goToStart variable and go 2 lines back if zero skip
169: goto $-2
170: ; Go back to main loop and listen for new drink
171: goto Main
172:
173: GinLemon:
174: ;Since Gin is the start bottle no need to turn just open valve
175: bsf PORTB, GinValve ;Open valve
176: call Delay_1s ;Dispense for 1 sec
177: bcf PORTB, GinValve ;Close valve
178: ;Open Lemon valve on the side.
179: bsf PORTB, LemonValve ;Open valve
```

```
180: call Delay_1s      ;Dispense Lemon for 1 sec
181: bcf PORTB, LemonValve ;Close valve
182: ; Go back to main loop and listen for new drink
183: goto Main
184:
185: RomOgCola:
186: ;Go to Rom relative to the Gin bottle meaning one turn to the left and open valve
187: call MoveOneBottleLeft
188: bsf PORTB, RomValve ;Open valve
189: call Delay_1s      ;Dispense Rom for 1 sec
190: bcf PORTB, RomValve ;Close valve
191: ;Open cola valve on the side.
192: bsf PORTB, ColaValve ;Open valve
193: call Delay_1s      ;Dispense Cola for 1 sec
194: bcf PORTB, ColaValve ;Close valve
195: ;Go to start
196: movlw D'2'
197: movwf goToStart    ;Set goToStart to 1 meaning it needs to turn 1 times to the right to return to
start
198: call MoveOneBottleRight
199: decfsz goToStart, 1 ;Minus 1 from the goToStart variable and go 2 lines back if zero skip
200: goto $-2
201: ; Go back to main loop and listen for new drink
202: goto Main
203:
204: VodkaLemon:
205: ;Go to Vodka relative to the Gin bottle meaning two turns to the left and open valve
206: call MoveOneBottleLeft
207: call MoveOneBottleLeft
208: bsf PORTB, VodkaValve ;Open valve
209: call Delay_1s      ;Dispense for 1 sec
210: bcf PORTB, VodkaValve ;Close valve
211: ;Open Lemon valve on the side.
212: bsf PORTB, LemonValve ;Open valve
213: call Delay_1s      ;Dispense Lemon for 1 sec
214: bcf PORTB, LemonValve ;Close valve
215: ;Go to start
216: movlw D'2'
217: movwf goToStart    ;Set goToStart to 2 meaning it needs to turn 2 times to the right to return to
start
218: call MoveOneBottleRight
219: decfsz goToStart, 1 ;Minus 1 from the goToStart variable and go 2 lines back if zero skip
220: goto $-2
221: ; Go back to main loop and listen for new drink
222: goto Main
223:
```

224: VodkaJuice:

225: ;Go to Vodka relative to the Gin bottle meaning two turn to the left and open valve
226: call MoveOneBottleLeft
227: call MoveOneBottleLeft
228: bsf PORTB, VodkaValve ;Open valve
229: call Delay_1s ;Dispense for 1 sec
230: bcf PORTB, VodkaValve ;Close valve
231: ;Open Appeljuice valve on the side.
232: bsf PORTB, AppelsinValve ;Open valve
233: call Delay_1s ;Dispense Lemon for 1 sec
234: bcf PORTB, AppelsinValve ;Close valve
235: ;Go to start
236: movlw D'3'
237: movwf goToStart ;Set goToStart to 2 meaning it needs to turn 2 times to the right to return to start
238: call MoveOneBottleRight
239: decfsz goToStart, 1 ;Minus 1 from the goToStart variable and go 2 lines back if zero skip
240: goto \$-2
241: ; Go back to main loop and listen for new drink
242: goto Main
243:
244:

245: VodkaCola:

246: ;Go to Vodka relative to the Gin bottle meaning two turn to the left and open valve
247: call MoveOneBottleLeft
248: call MoveOneBottleLeft
249: bsf PORTB, VodkaValve ;Open valve
250: call Delay_1s ;Dispense for 1 sec
251: bcf PORTB, VodkaValve ;Close valve
252: ;Open Lemon valve on the side.
253: bsf PORTB, ColaValve ;Open valve
254: call Delay_1s ;Dispense Lemon for 1 sec
255: bcf PORTB, ColaValve ;Close valve
256: ;Go to start
257: movlw D'2'
258: movwf goToStart ;Set goToStart to 2 meaning it needs to turn 2 times to the right to return to start
259: call MoveOneBottleRight
260: decfsz goToStart, 1 ;Minus 1 from the goToStart variable and go 2 lines back if zero skip
261: goto \$-2
262: ; Go back to main loop and listen for new drink
263: goto Main
264:
265: Astronaut:

266: ;Go to Råstoff relative to the Gin bottle meaning four turn to the left and open valve
267: call MoveOneBottleLeft

```
268: call MoveOneBottleLeft
269: call MoveOneBottleLeft
270: call MoveOneBottleLeft
271: bsf PORTB, RåstofValve ;Open valve
272: call Delay_1s ;Dispense for 1 sec
273: bcf PORTB, RåstofValve ;Close valve
274: ;Open Lemon valve on the side.
275: bsf PORTB, LemonValve ;Open valve
276: call Delay_1s ;Dispense Lemon for 1 sec
277: bcf PORTB, LemonValve ;Close valve
278: ;Go to start
279: movlw D'4'
280: movwf goToStart ;Set goToStart to 4 meaning it needs to turn 4 times to the right to return to start
281: call MoveOneBottleRight
282: decfsz goToStart, 1 ;Minus 1 from the goToStart variable and go 2 lines back if zero skip
283: goto $-2
284: ; Go back to main loop and listen for new drink
285: goto Main
286:
287: ;The LongIsland as the group calls it is all the alcohol combined
288: LongIsland:
289: ;Open for gin
290: bsf PORTB, GinValve ;Open valve
291: call Delay_1s ;Dispense for 1 sec
292: bcf PORTB, GinValve ;Close valve
293: ;Go to the next flask and open valve
294: call MoveOneBottleLeft
295: bsf PORTB, RomValve ;Open valve
296: call Delay_1s ;Dispense for 1 sec
297: bcf PORTB, RomValve ;Close valve
298: ;Go to the next flask and open valve
299: call MoveOneBottleLeft
300: bsf PORTB, VodkaValve ;Open valve
301: call Delay_1s ;Dispense for 1 sec
302: bcf PORTB, VodkaValve ;Close valve
303: ;Skip mangosirup and go to the next flask and open valve
304: call MoveOneBottleLeft
305: call MoveOneBottleLeft
306: bsf PORTB, RåstofValve ;Open valve
307: call Delay_1s ;Dispense for 1 sec
308: bcf PORTB, RåstofValve ;Close valve
309: ;Open Cola valve on the side.
310: bsf PORTB, ColaValve ;Open valve
311: call Delay_1s ;Dispense Lemon for 1 sec
312: bcf PORTB, ColaValve ;Close valve
```

```
313: ;Go to start
314: movlw D'4'
315: movwf goToStart      ;Set goToStart to 4 meaing it needs to turn 4 times to the right to return to
start
316: call MoveOneBottleRight
317: decfsz goToStart, 1  ;Minus 1 from the goToStart variable and go 2 lines back if zero skip
318: goto $-2
319: ; Go back to main loop and listen for new drink
320: goto Main
321:
322: ; Delay function
323: MovementDelay:
324:     goto Delay_1s          ; Waits 1 secound can be cahnged to waht ever by goto more
Delay funktions.
325:     return
326: ;*****Delay functions*****
327: ; Delay function for 10 ms
328: Delay_10ms:
329:     movlw D'13'          ; Approximately 13 iterations for 10 ms delay
330:     movwf d1            ; Move W to d1
331: Delay_0:
332:     movlw D'255'         ; Each iteration of the inner loop takes approximately 3 µs
333:     movwf d2            ; Move W to d2
334: Delay_1:
335:     nop
336:     decfsz d2, 1        ; Decrement d2. Skip next instruction if zero
337:     goto $-2           ; Go to next instruction
338:     decfsz d1, 1        ; Decrement d1. Skip next instruction if zero
339:     goto Delay_0        ; Repeat outer loop
340:     return             ; Return from subroutine
341:
342: ; Delay function for 100 ms
343: Delay_100ms:
344:     movlw D'10'          ; Approximately 10 iterations for 100 ms delay
345:     movwf d3            ; Move W to d3
346: Delay_2:
347:     call Delay_10ms     ; Call the 10 ms delay function
348:     decfsz d3, 1        ; Decrement d3. Skip next instruction if zero
349:     goto Delay_2        ; Repeat outer loop
350:     return             ; Return from subroutine
351:
352: ; Delay function for 1 second
353: Delay_1s:
354:     movlw D'10'          ; 10 iterations for 1 second delay
355:     movwf d4            ; Move W to d4
356: Delay_3:
```

```
357:    call  Delay_100ms ; Call the 100 ms delay function
358:    decfsz d4, 1      ; Decrement d4. Skip next instruction if zero
359:    goto  Delay_3   ; Repeat outer loop
360:    return           ; Return from subroutine
361:
362:    end
```

Bilag 16 - Arduino kode uden PIC

```
1: #include <LiquidCrystal_I2C.h>
4: // initalliasation of variables
5: int CurrentPos = 1;
6: unsigned long startTime = 0;
7: int ButtonState;
8:
9: // Set the LCD number of columns and rows
10: int lcdColumns = 16;
11: int lcdRows = 2;
12:
13: // Set LCD address, number of columns and rows
14: LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);
15:
16: // Define pin assignments
17: const int potPin = A0; // Connect potentiometer to A0
18: const int buttonPin = 2; // Connect button to D2
19: const int MotorActivate = 3;
20: const int MotorDirection = 4;
21: const int GinValve = 5;
22: const int LemonValve = 6;
23: const int VodkaValve = 7;
24: const int OrangeJuiceValve = 8;
25: const int RaastofValve = 9;
26: const int Lift = 10;
27:
28: // Set the Position of the valves
29: const int GinPos = 1;
30: const int VodkaPos = 2;
31: const int RaastofPos = 3;
32: const int LemonPos = 4;
33: const int OrangeJuicePos = 5;
34:
35: // Time variables
36: const int MsBetweenIncrediences = 2000;
37:
38:
39:
40:
41: // Machine specific variables
42: int TimeBetweenValves = 1140;
43: int MsToMIConstant = 50;
44: // SKAL TESTES!!!!!! - Hvor længe er den om at kører en hel omgang / 5
45: // Hvor længe er den om at hælde 100ml / 100
46:
```

```
47:  
48:  
49: // setting up drink indexing  
50: int drinkIndex = 0; // Index of the currently highlighted drink  
51: String drinks[] = {"Gin lemon", "Gin juice", "Jordbaer Bomb", "Filur", "Vodka lemon", "Vodka juice",  
"Astronaut", "Long island"};  
52: int numDrinks = sizeof(drinks) / sizeof(drinks[0]); // Number of drinks  
53:  
54: // function declarations - Valves  
55: void Gin(int ml){  
56: Serial.println("Gin valve function called");  
57: for (; GinPos > CurrentPos; CurrentPos++){  
58: GoLeft();  
59: }  
60: digitalWrite(GinValve, HIGH);  
61: Serial.println("Pouring");  
62: delay(ml * MsToMIConstant);  
63: digitalWrite(GinValve, LOW);  
64: }  
65:  
66: void Vodka(int ml){  
67: Serial.println("Vodka valve function called");  
68: for (; VodkaPos > CurrentPos; CurrentPos++){  
69: GoLeft();  
70: }  
71: digitalWrite(VodkaValve, HIGH);  
72: Serial.println("Pouring");  
73: delay(ml * MsToMIConstant);  
74: digitalWrite(VodkaValve, LOW);  
75: }  
76:  
77: void Raastof(int ml){  
78: Serial.println("Råstof valve function called");  
79: for (; RaastofPos > CurrentPos; CurrentPos++){  
80: GoLeft();  
81: }  
82: digitalWrite(RaastofValve, HIGH);  
83: Serial.println("Pouring");  
84: delay(ml * MsToMIConstant);  
85: digitalWrite(RaastofValve, LOW);  
86: }  
87:  
88: void Lemon(int ml){  
89: Serial.println("Lemon valve function called");  
90: for (; LemonPos > CurrentPos; CurrentPos++){  
91: GoLeft();
```

```
92: }
93: digitalWrite(LemonValve, HIGH);
94: Serial.println("Pouring");
95: delay(ml * MsToMIConstant);
96: digitalWrite(LemonValve, LOW);
97: }
98:
99: void OrangeJuice(int ml){
100: Serial.println("Orange Juice valve function called");
101: for (; OrangeJuicePos > CurrentPos; CurrentPos++){
102:   GoLeft();
103: }
104: digitalWrite(OrangeJuiceValve, HIGH);
105: Serial.println("Pouring");
106: delay(ml * MsToMIConstant);
107: digitalWrite(OrangeJuiceValve, LOW);
108: }
109:
110: //Function declaration - MotorMovement
111:
112: void GoLeft(){
113: Serial.println("Going left");
114: digitalWrite(MotorActivate, HIGH);
115: delay(TimeBetweenValves);
116: digitalWrite(MotorActivate, LOW);
117: }
118:
119: void GoRight(){
120: Serial.println("Going right");
121: digitalWrite(MotorActivate, HIGH);
122: digitalWrite(MotorDirection, HIGH);
123: delay(TimeBetweenValves);
124: digitalWrite(MotorActivate, LOW);
125: digitalWrite(MotorDirection, LOW);
126: }
127:
128: void ReturnToStart(){
129: // Display "Your drink is done" on the LCD
130: lcd.clear();
131: lcd.setCursor(0, 0);
132: lcd.print(drinks[drinkIndex]);
133: lcd.setCursor(0, 1);
134: lcd.print("Is ready");
135:
136: // Go back to start
137: for (; CurrentPos > 1; CurrentPos--){
```

```
138: GoRight();  
139: delay(300);  
140: }  
141: }  
142:  
143: // Setup serial, LCD screen and the pinmode  
144: void setup() {  
145: Serial.begin(115200);  
146: // Initialize LCD  
147: lcd.init();  
148: // Turn on LCD backlight  
149: lcd.backlight();  
150:  
151: // Set button pin as input  
152: pinMode(buttonPin, INPUT);  
153: // Set the rest as output  
154: pinMode(MotorActivate, OUTPUT);  
155: pinMode(MotorDirection, OUTPUT);  
156: pinMode(GinValve, OUTPUT);  
157: pinMode(LemonValve, OUTPUT);  
158: pinMode(VodkaValve, OUTPUT);  
159: pinMode(OrangeJuiceValve, OUTPUT);  
160: pinMode(RaastofValve, OUTPUT);  
161: pinMode(Lift, OUTPUT);  
162: }  
163:  
164: void loop() {  
165: // Read analog input from potentiometer  
166: int sensorValue = analogRead(potPin);  
167:  
168: // Map the potentiometer value to the range of drinks  
169: drinkIndex = map(sensorValue, 0, 1023, 0, numDrinks);  
170:  
171: // Ensure drinkIndex stays within the bounds  
172: drinkIndex = constrain(drinkIndex, 0, numDrinks - 1);  
173:  
174: int nextIndex = (drinkIndex + 1) % numDrinks;  
175:  
176: // Display the highlighted drink on the LCD  
177: lcd.setCursor(0, 0);  
178: lcd.print("-> ");  
179: lcd.print(drinks[drinkIndex]);  
180: lcd.print("          "); // Clear the rest of the line  
181:  
182: // Display the next drink on the LCD  
183: lcd.setCursor(0, 1);
```

```
184: lcd.print("      "); // Clear the entire line
185: lcd.setCursor(0, 1);
186: if (nextIndex != 0) {
187:   lcd.print(" ");
188:   lcd.print(drinks[nextIndex]);
189: }
190:
191: // Check if the button is pressed to select the drink
192: if (digitalRead(buttonPin) == HIGH) {
193:   Serial.print("Selected Drink: ");
194:   Serial.println(drinks[drinkIndex]);
195:
196: // Display "Making your drink" on the LCD
197: lcd.clear();
198: lcd.setCursor(0, 0);
199: lcd.print("Making");
200: lcd.setCursor(0, 1);
201: lcd.print(drinks[drinkIndex]);
202:
203: // Activate Lift, GO up.
204: digitalWrite(Lift, HIGH);
205: delay(3000);
206:
207: // Switch determines what drink to make - And calls the needed functions.
208: switch(drinkIndex){
209:   case 0:
210:     // Make Gin Lemon
211:     Serial.println("Making Gin Lemon");
212:     Gin(50);
213:     delay(MsBetweenIncredences);
214:     Lemon(200);
215:     delay(MsBetweenIncredences);
216:     ReturnToStart();
217:     delay(1000);
218:     break;
219:
220:   case 1:
221:     // Make Gin Juice
222:     Serial.println("Making Gin Juice");
223:     Gin(50);
224:     delay(MsBetweenIncredences);
225:     OrangeJuice(200);
226:     delay(MsBetweenIncredences);
227:     ReturnToStart();
228:     break;
229:
```

```
230: case 2:  
231: // Make Jordbær Bombe (Strawberry Bomb) https://raastoff.dk/drink/strawberry-bomb/  
232: Serial.println("Making Jordbær boombe");  
233: Raastof(20);  
234: delay(MsBetweenIncrediences);  
235: Vodka(20);  
236: delay(MsBetweenIncrediences);  
237: Lemon(100);  
238: delay(MsBetweenIncrediences);  
239: ReturnToStart();  
240: break;  
241:  
242: case 3:  
243: // Make Filur https://raastoff.dk/drink/frozen-filur/  
244: Serial.println("Making Filur");  
245: Raastof(60);  
246: delay(MsBetweenIncrediences);  
247: OrangeJuice(80);  
248: delay(MsBetweenIncrediences);  
249: ReturnToStart();  
250: break;  
251:  
252: case 4:  
253: // Make Vodka Lemon  
254: Serial.println("Making Vodka Lemon");  
255: Vodka(100);  
256: delay(MsBetweenIncrediences);  
257: Lemon(100);  
258: delay(MsBetweenIncrediences);  
259: ReturnToStart();  
260: break;  
261:  
262: case 5:  
263: // Make Vodka Juice  
264: Serial.println("Making VODka Juice");  
265: Vodka(100);  
266: delay(MsBetweenIncrediences);  
267: OrangeJuice(100);  
268: delay(MsBetweenIncrediences);  
269: ReturnToStart();  
270: break;  
271:  
272: case 6:  
273: // Make Astronaut https://raastoff.dk/drink/astronaut-da/  
274: Serial.println("Making Astronaut");  
275: Raastof(60);
```

```
276:     delay(MsBetweenIncrediences);
277:     Lemon(150);
278:     delay(MsBetweenIncrediences);
279:     ReturnToStart();
280:     break;
281:
282: case 7:
283: // Make Long Island
284: Serial.println("Making Gin LongIsland");
285: Gin(20);
286: delay(MsBetweenIncrediences);
287: Vodka(30);
288: delay(MsBetweenIncrediences);
289: Raastof(20);
290: delay(MsBetweenIncrediences);
291: Lemon(60);
292: delay(MsBetweenIncrediences);
293: OrangeJuice(60);
294: delay(MsBetweenIncrediences);
295: ReturnToStart();
296: break;
297: }
298: // deactivate Lift, GO down
299: digitalWrite(Lift, LOW);
300: // Display "Your drink is done" on the LCD
301: lcd.clear();
302: lcd.setCursor(0, 0);
303: lcd.print(drinks[drinkIndex]);
304: lcd.setCursor(0, 1);
305: lcd.print("Is ready");
306: delay(5000);
307:
308: lcd.clear(); // Clear LCD for next iteration
309: }
310: }
```

Bilag 17 - Arduino kode med PIC

```
#include <LiquidCrystal_I2C.h>
// Set the LCD number of columns and rows
int lcdColumns = 16;
int lcdRows = 2;

// Set LCD address, number of columns and rows
LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);

// Define pins
const int potPin = A0; // Connect potentiometer to A0
const int buttonPin = 2; // Connect button to D2

int drinkIndex = 0; // Index of the currently highlighted drink
String drinks[] = {"Gin Hass", "Gin lemon", "Rom og cola", "Vodka lemon", "Vodka juice", "Vodka cola",
"Astronaut", "Long island"};
int numDrinks = sizeof(drinks) / sizeof(drinks[0]);

void setup() {
    Serial.begin(115200);
    // Initialize LCD
    lcd.init();
    // Turn on LCD backlight
    lcd.backlight();

    // Set button pin as input
    pinMode(buttonPin, INPUT);
}

void loop() {
    // Read analog input from potentiometer
    int sensorValue = analogRead(potPin);

    // Map the potentiometer value to the range of drinks
    drinkIndex = map(sensorValue, 0, 1023, 0, numDrinks);

    // Ensure drinkIndex stays within the bounds
    drinkIndex = constrain(drinkIndex, 0, numDrinks - 1);

    int nextIndex = (drinkIndex + 1) % numDrinks;

    // Display the highlighted drink on the LCD
    lcd.setCursor(0, 0);
    lcd.print("->");
    lcd.print(drinks[drinkIndex]);
```

```
// Display the next drink on the LCD
lcd.setCursor(0, 1);
if (nextIndex != 0) {
    lcd.print(" ");
    lcd.print(drinks[nextIndex]);
}

// Check if the button is pressed to select the drink
if (digitalRead(buttonPin) == HIGH) {
    // Send drinkIndex to PIC via USART
    Serial.write(drinkIndex);

    // Display "Making your drink" on the LCD
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Making");
    lcd.setCursor(0, 1);
    lcd.print(drinks[drinkIndex]);

    delay(5000);
    // Wait for the PIC to send confirmation that the drink is done
    char data = Serial.read();

    // Wait for the the PIC to clear the confirmation meaning the drink is in progress
    while (data == 0x01) {
        // Waiting while the drink is being made
        data = Serial.read();
    }
    // Display "Your drink is done" on the LCD
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(drinks[drinkIndex]);
    lcd.setCursor(0, 1);
    lcd.print("Is ready");
    delay(10000); // Display for 10 sec before returning to main loop
}

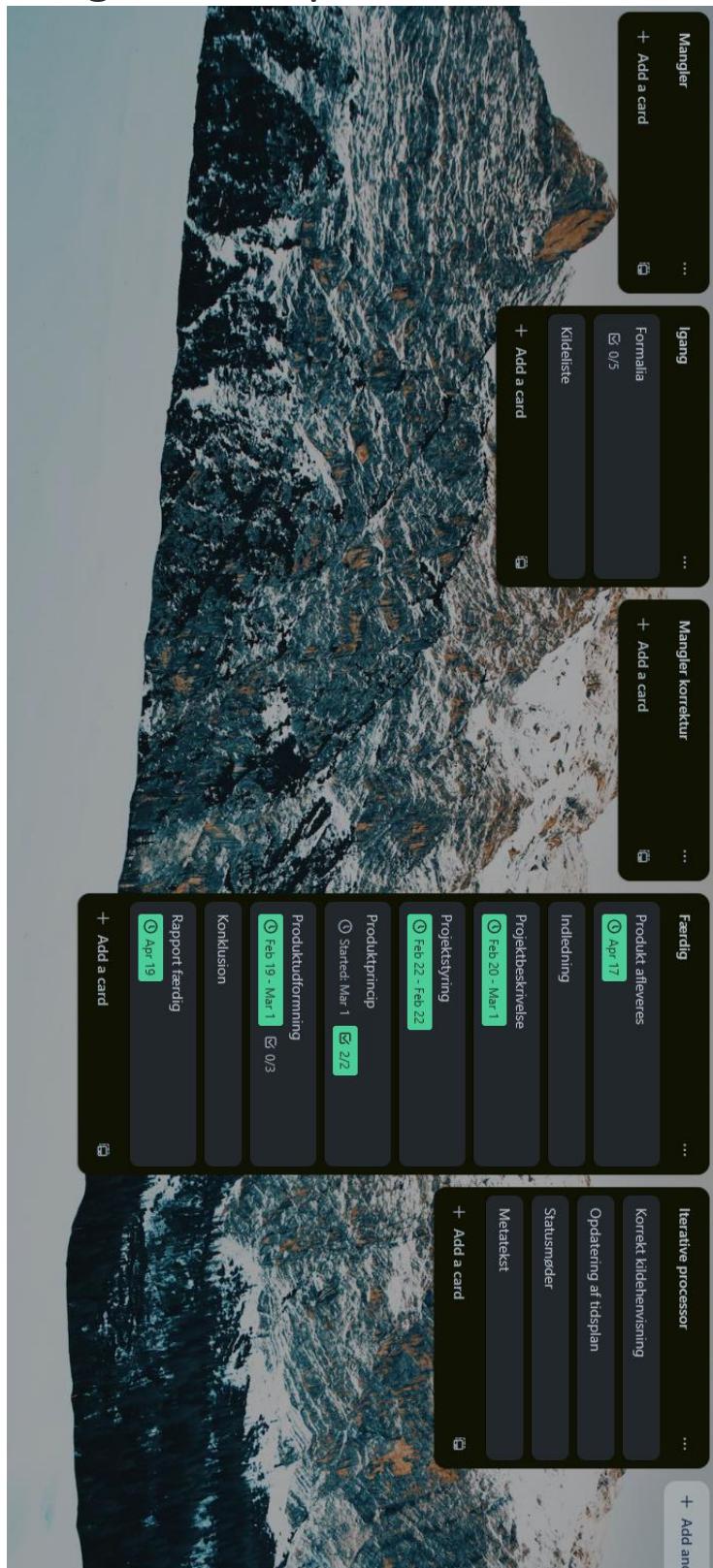
delay(100); // Adjust delay as needed
lcd.clear(); // Clear LCD for next iteration
}
```

Bilag 18 - Kodeændringer ved analog kommunikation

```
48: ;Initialisering
49: init
50: bank1
51: movlw B'00000000'
52: movwf TRISC      ; Set all PORTC pins as output for lift
53: movlw B'00000000'
54: movwf TRISB      ; Set all PORTB pins as output for magnetic valves
55: movlw B'0000001'   ; Set RA0 as output for stepper motor activation and RA1 as output for direction
56: movwf TRISA
57: movlw B'10001111' ; Set RA1 and 4 to digital and leave RA0 as analog. set RA3 and RA2 to refrence
voltage. Set ADFM to 1 for Right justified.
58: movwf ADCON1
59: bank0
60: clrf PORTB      ; Clear PortB
61: clrf PORTA      ; Clear PortA
62: clrf PORTC
63:
64: ;Main loop
65:
66:
67: ; Main loop
68: Main:
69: ; Read arduino value
70: bcf PORTC, 2
71: call ReadAnalogValue
72: movf AnalogValue, W ; Move the analog value to W register
73: sublw D'3'          ; Subtract 3 from the analog value
74: btfsc STATUS, C    ; Check if result wrapped around adn activated the C flag.
75: goto Main
76: bsf PORTC, 2
77: call determineDrink ; Determine selected drink
78:
79:
82: ; Subroutine to read potentiometer value
83: ReadAnalogValue:
84: movlw B'00000101' ; Select channel AN0 and enable ADC and Start conversion
85: movwf ADCON0
86:
87: WaitADConversion:
88: btfsc ADCON0, GO  ; Wait for conversion to finish
89: goto WaitADConversion
90:
91: bank1
92: movf ADRESL, W    ; Read result
93: bank0
94: movwf AnalogValue
95:
96:
```

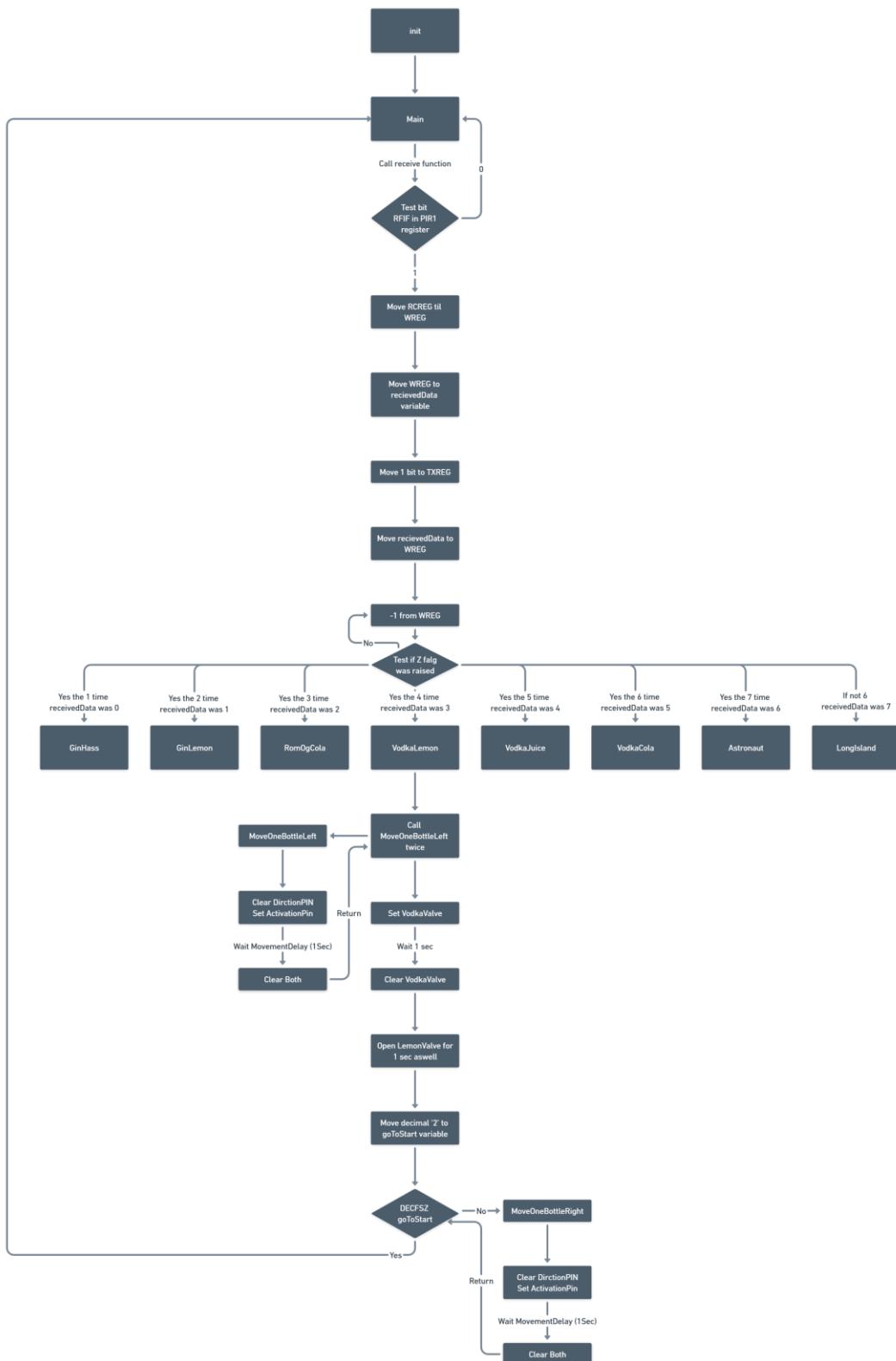
```
97:    return
98:
99: ; Subroutine to determine selected drink
100: determineDrink:
101:    movf AnalogValue, W ; Move the analog value to W register
102:    sublw D'32'          ; Subtract 32 from the analog value
103:    btfsc STATUS, C      ; Check if result wrapped around and activated the C flag.
104:    goto GinHass         ; If within threshold, call GinHass
105:
106:    ; Compare with threshold for GinLemon
107:    movf AnalogValue, W ; Move the analog value to W register again
108:    sublw D'64'          ; Subtract 64 from the analog value
109:    btfsc STATUS, C      ; Check if result wrapped around and activated the C flag.
110:    goto GinLemon        ; If within threshold, call GinLemon
111:
112:    ; Do the same for the rest
113:    movf AnalogValue, W
114:    sublw D'96'
115:    btfsc STATUS, C
116:    goto RomOgCola
117:
118:    movf AnalogValue, W
119:    sublw D'128'
120:    btfsc STATUS, C
121:    goto VodkaLemon
122:
123:    movf AnalogValue, W
124:    sublw D'128'
125:    btfsc STATUS, C
126:    goto VodkaLemon
127:
128:    movf AnalogValue, W
129:    sublw D'160'
130:    btfsc STATUS, C
131:    goto VodkaJuice
132:
133:    movf AnalogValue, W
134:    sublw D'192'
135:    btfsc STATUS, C
136:    goto VodkaCola
137:
138:    movf AnalogValue, W
139:    sublw D'224'
140:    btfsc STATUS, C
141:    goto Astronaut
142:
143:    movf AnalogValue, W
144:    sublw D'255'
145:    btfsc STATUS, C
146:    goto LongIsland
```

Bilag 19 - Tidsplan

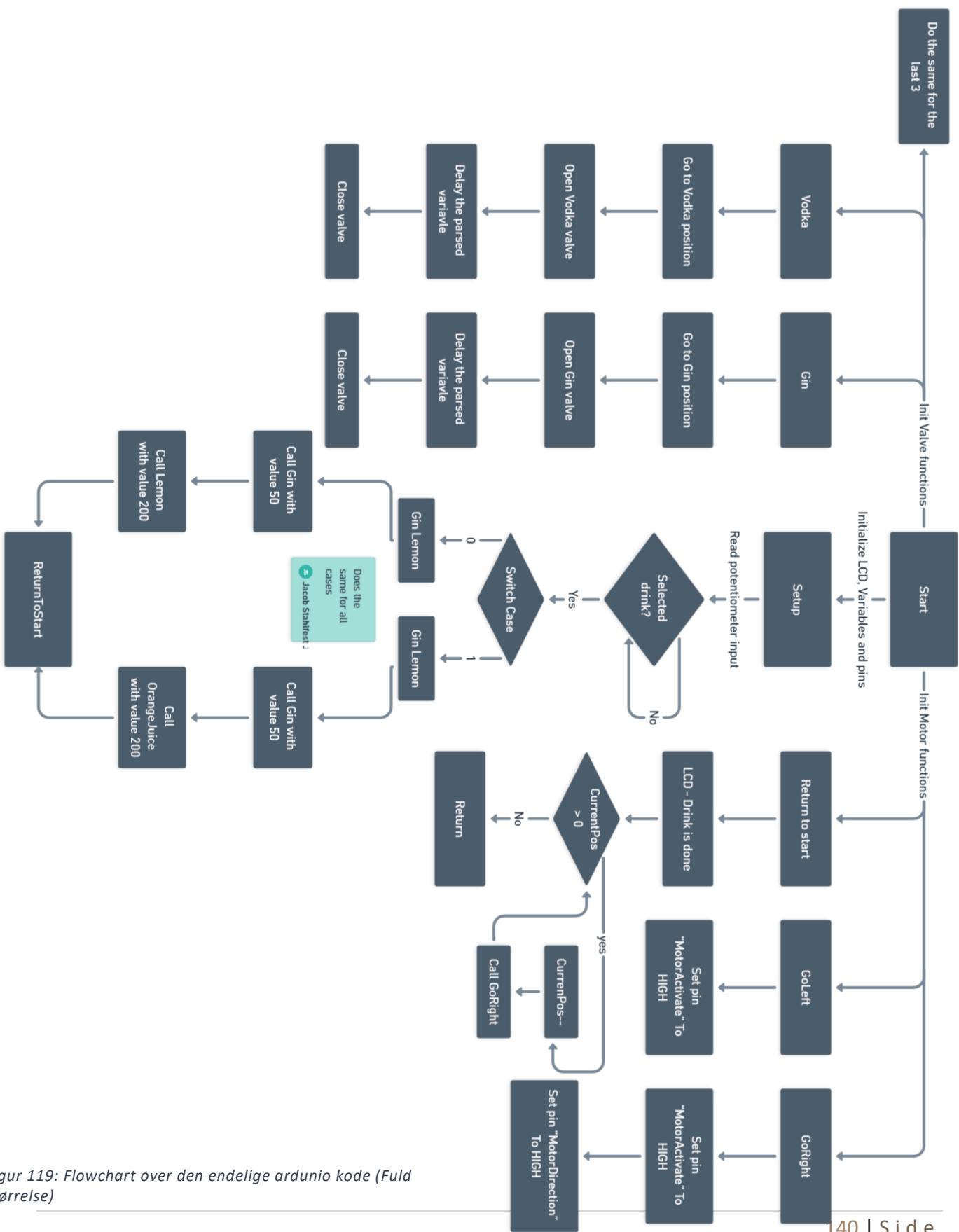


Figur 117: Tidsplan udarbejdet i trello.

Bilag 20 - Flow Charts i fuld størrelse



Figur 118: Flowchart over hovedløkken på PIC'en (Fuld størrelse)

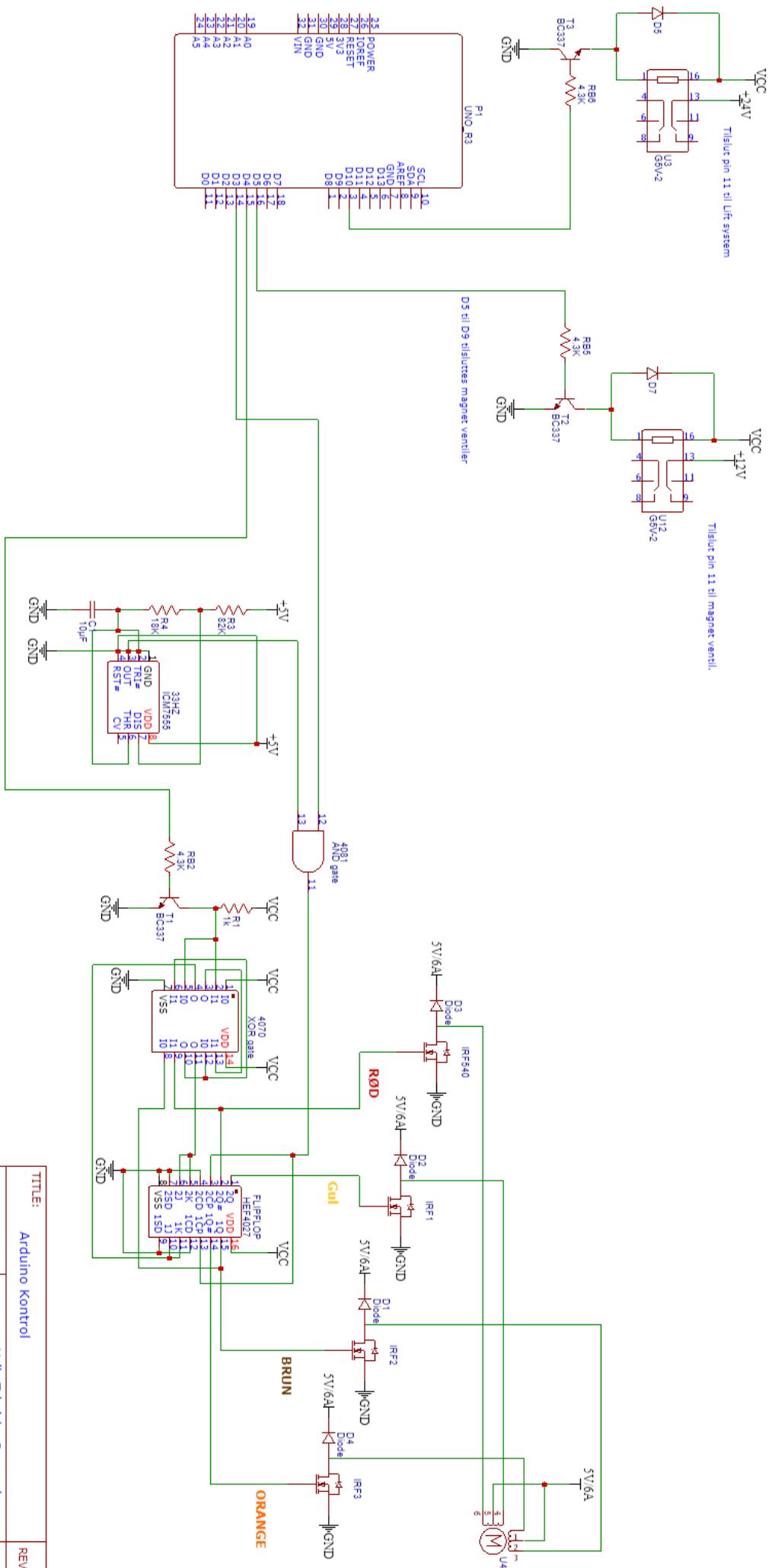


Figur 119: Flowchart over den endelige arduinokode (Fuld størrelse)

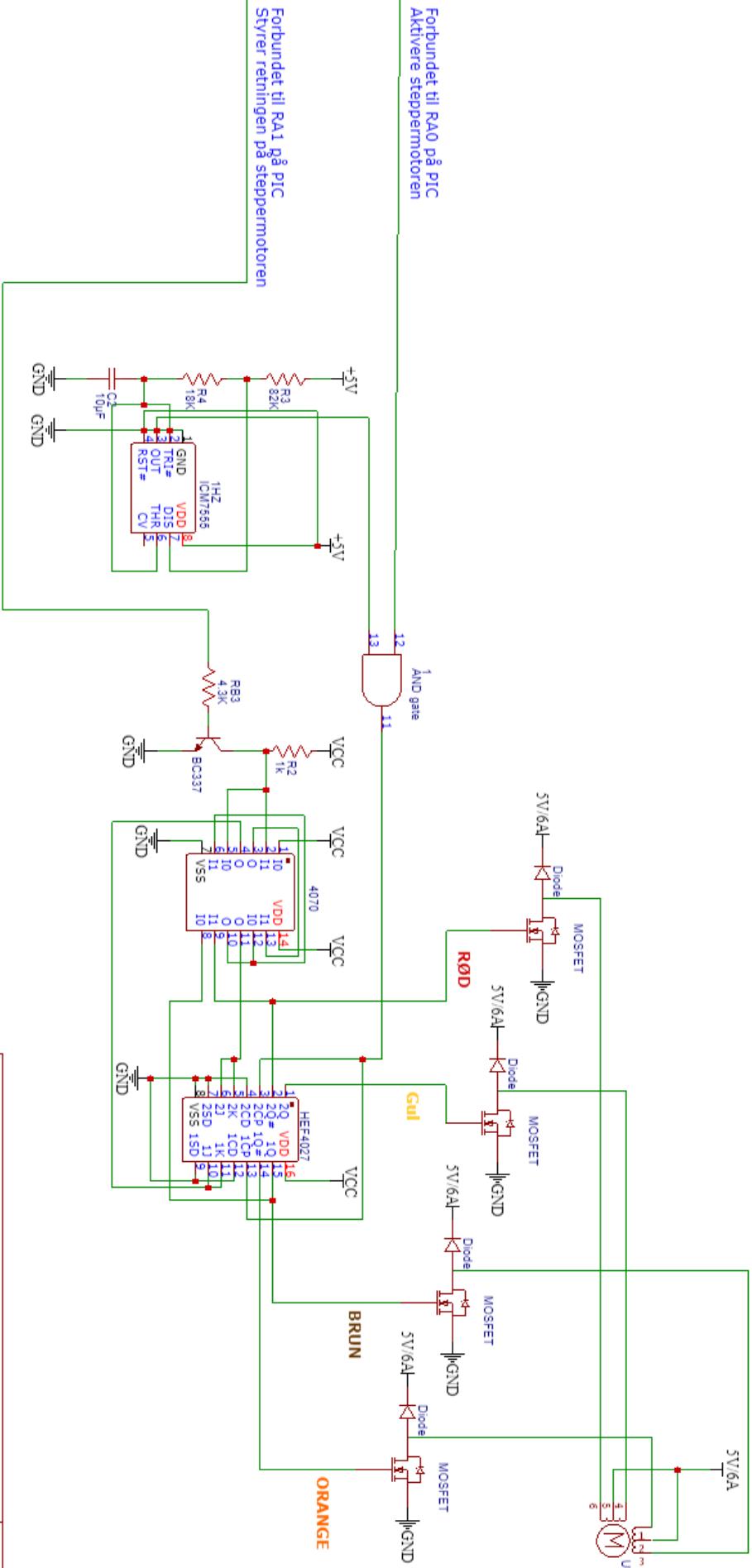
Bilag 21 - Diagrammer i Fuld størrelse

På de efterfølgende 3 sider er de el tekniske diagrammer indsat i fuld størrelse.

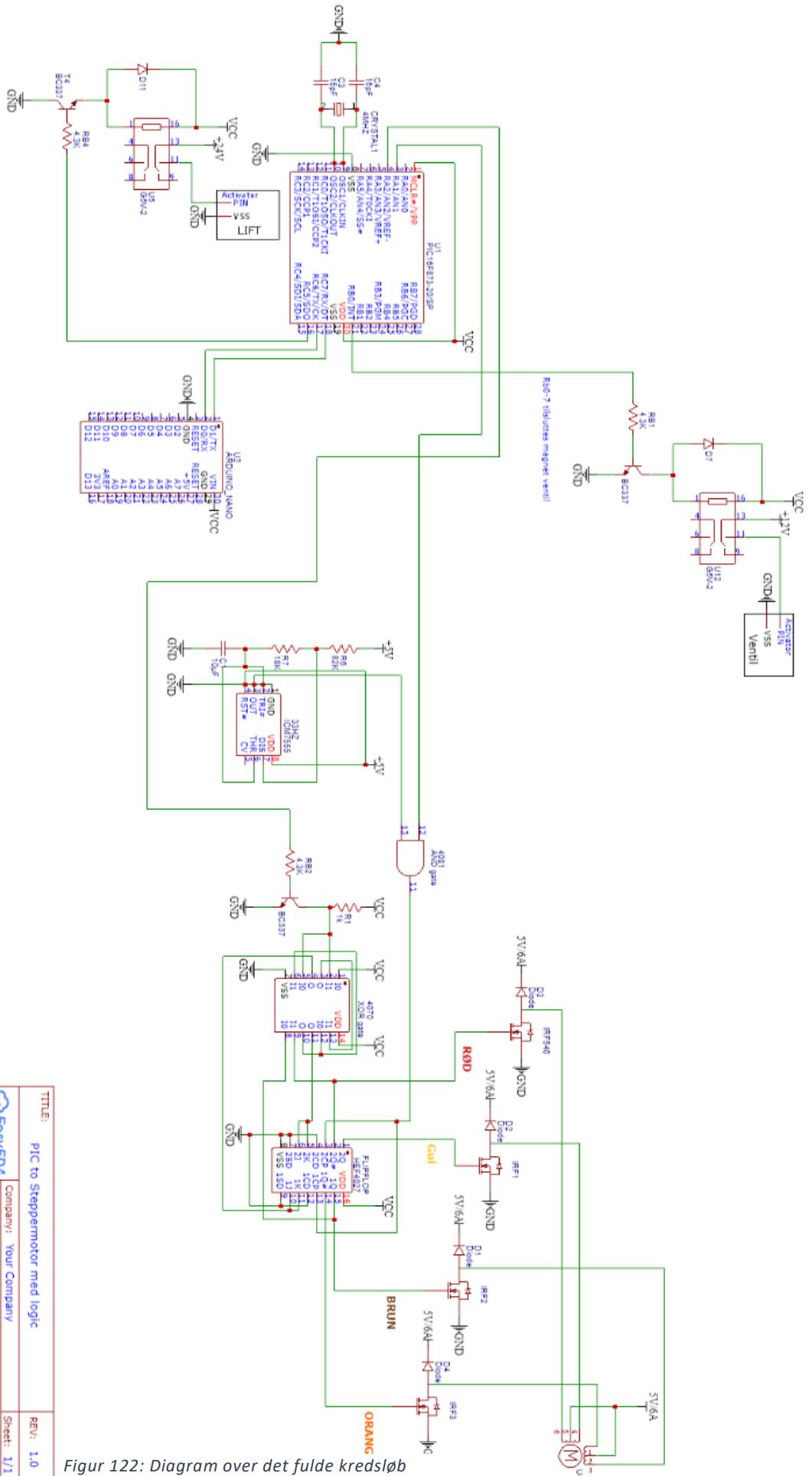
Magn



Figur 120: Diagram over arduino kontrol (Fuld størrelse)



Figur 121: Diagram over Stepper motor logik (Fuld)



Figur 122: Diagram over det fulde kredsløb