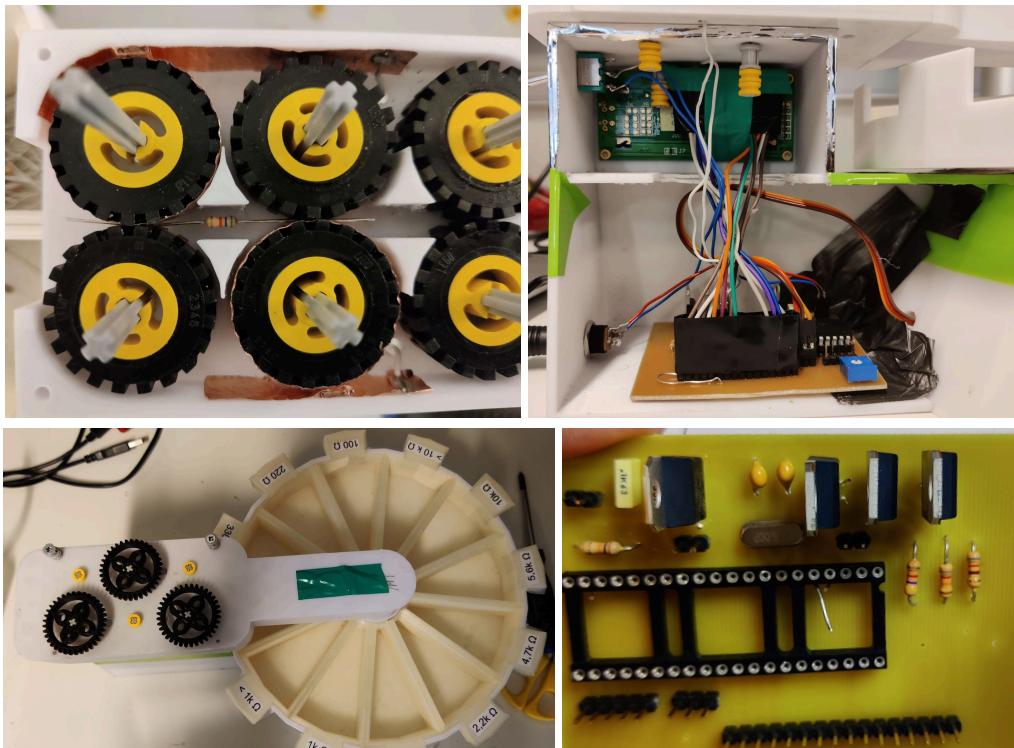


# Projekt 5: Eksamensprojekt

Udviklingen af en autonom modstandssorteringsmaskine med analog- og digitalteknik og elektronik i teknikfaget: El, Udvikling og Produktion A



Noah S.K. Jørgensen, Kristiyan M. Georgiev og Clara Holst, 3.H

Eksamensprojekt: Modstandssorteringsmaskine

Teknikfag: EL, Udvikling og Produktion A

Odense Tekniske Gymnasium

Underviser: Jannich Petersen (JAPE)

Dato for udlevering af projektet: 20/02/2023

Afleveringsdato: 30/04/2023

Tegn uden bilag: 72.025 ~ 30 normalsider

Antal tegn i alt: 92.379 ~ 38,5 normalsider

# Indholdsfortegnelse

<b>Indholdsfortegnelse</b>	<b>2</b>
<b>Resumé</b>	<b>4</b>
<b>Indledning</b>	<b>4</b>
<b>Problemanalyse og forundersøgelse</b>	<b>5</b>
Omfang af eksamensprojektet	5
Projektoplæg: Udstyr til skolen (4)	6
<b>Vejen fra projektoplæg til produkt</b>	<b>6</b>
Motivation	6
Idégenerering	7
Produktets funktioner	10
<b>Projektbeskrivelse</b>	<b>14</b>
Projektafgrænsning	15
Kravspecifikation	15
<b>Planlægning af projektet</b>	<b>17</b>
Fremgangsmåde	17
<b>Teori og fremgangsmåder</b>	<b>17</b>
KiCad	19
Fremstilling af PCB	20
<b>Produktudvikling</b>	<b>21</b>
Opstart af projektet	21
Valg af microcontroller	23
Regulering af indgangsspænding	24
Anvendelsen af en 16 MHz krystal	25
Anvendelsen af LEGO	26

Anvendelsen af 3D-print	26
Identificering af modstande	28
Præcisering af målingen af modstande	28
KiCad kredsløb	31
Anvendelsen af servomotor	33
Implementering af servoen i systemet	35
Indsættelse af modstande i systemet	37
LCD-skærmens funktioner	40
Beskrivelse af programmeringen	40
Beskrivelse af de anvendte registre i programmeringen	44
Beskrivelse af modstandssorteringsmaskinen	45
<b>Opfyldelse af krav</b>	<b>47</b>
<b>Konklusion</b>	<b>50</b>
<b>Evaluering</b>	<b>51</b>
Projektstyring med SCRUM	51
Gruppearbejde og indsats	51
<b>Underskrifter, den 30. April 2023</b>	<b>52</b>
<b>Kildeliste</b>	<b>53</b>
<b>Bilag</b>	<b>54</b>
Bilag 1: Gruppekontrakt	54
Bilag 2: Tidsplan	56
Bilag 3: Programmeringen	60
Bilag 4: Registre til PIC16F877a	65
Bilag 5: SCRUM-projektstyring	68
Bilag 6: Alle beregninger	73

## Resumé

Dette projekt beskriver en autonom modstandssorteringsmaskine, der identificerer modstande og sorterer dem automatisk ved hjælp af en servomotor. Maskinen fungerer ved at fastholde modstanden mellem to kobberplader og sammenligne dens størrelse med foruddefinerede størrelseskategorier. Maskinen bruger en servomotor til at dreje rummene og placere modstanden i det rigtige rum, baseret på dens størrelse.

Programmet initialiserer forskellige porte og ADC'en og kører derefter i en uendelig løkke, der tænder motoren og undersøger for en stabil forbindelse. Modstandens værdi kan vises på en LCD-skærm, og identificeringen af modstandsværdien er autonom ved brug af spændingsdelerformlen og specifikke parallelforbindelser. Samlet set kan den autonome modstandssorteringsmaskine spare enorm tid på manuel sortering og sikre korrekt sortering af modstande.

## Indledning

Dette projekt sigter mod at hjælpe fremtidige elektronik-elever med at sortere og finde ønskede modstande til deres projekter effektivt. Derudover kan det hjælpe Odense Tekniske Gymnasium med at mindske deres forbrug af modstande, og dermed mindske deres økonomiske ressourcer og klimapåvirkning. Projektet indebærer design af en autonom modstandssorteringsmaskine, der identificerer størrelsen af en modstand og placerer den i den korrekte beholder ved hjælp af en servomotor. Maskinen sparar betydelige mængder tid i forhold til manuel sortering og sikrer, at modstandene sorteres nøjagtigt. Maskinen inkluderer også en LCD-skærm, der viser den målte modstandsværdi, hvilket gør den brugervenlig og informativ. Processen er autonom, og identifikationen af modstandsværdien er præcis, hvilket gør maskinen effektiv og effektiv. Vi vil forsøge at skabe dette apparat gennem arbejde der tager udgangspunkt i både analog- og digitalteknik og programmerbar elektronik. Dette bliver programmeret i programmeringssproget: 'C' med IDE'en: MatLab. Heri har vi forsøgt at fremstille et apparat, der kan sortere modstande ved at aflæse deres værdi og derefter placere dem i den korrekte kasse, så der ikke skal udføres noget manuelt arbejde.

# Problemanalyse og forundersøgelse

## Omfang af eksamsprojektet

Eksamensperioden i El-teknik begynder med udleveringen af projektoplægget mandag den 20. februar kl. 12:05 og slutter ved elektronisk aflevering af projektrapporten søndag den 30. april senest kl. 23:00. I alt er eksamsperioden på 96 lektioner. Du kan i afsnittet "Eksamensperiodens tidsplan" se et skema over forløbet af eksamsperioden og under afsnittet "Vigtige datoer og tidspunkter" kan du, som titlen antyder, finde vigtige datoer og tidspunkter.

## Udvælgelse af projektoplæg

Gruppen skulle udvælge ét af de syv projektoplæg. Det projektoplæg, som I vælger, bliver derved vores eksamsprojekt. Som beskrevet i opgavebeskrivelsen er gruppens valg er først bindende i det øjeblik, at projektbeskrivelsen er blevet godkendt. Derefter er det kun muligt at ændre projektbeskrivelsen i samarbejde med gruppens vejleder: Jannich Jakob Petersen (JAPE).

*Titlerne på de fem oplæg ses nedenfor:*

1. Styring, Automatisering og Overvågning
2. Fjernstyret/autonomt transportmiddel eller apparat
3. Interaktiv, Intelligent legeplads
4. Udstyr til skolen
5. Underholdning, lyd og lys
6. Lærings hjælpemidler til fritid, skole og handicap
7. Bestillingsopgaver

Med udgangspunkt i ét af oplæggene, fremstilles et fysisk og praktisk produkt, skrives en rapport om de undersøgelser, overvejelser og valg, der er foretaget i løbet af projektet, samt dokumentere det praktiske arbejde, der er blevet udført.

## Projektoplæg: Udstyr til skolen (4)

Skolen har altid brug for spændende nyt grej til vores laboratorier, der er mange muligheder både for at lave avancerede demonstrationsforsøg, apparater til måling af fysiske, kemiske eller biologiske fænomener, legetøj til matematik eller værktøj til vores teknik-værksteder, der kan i dette projekt arbejdes sammen med elever fra f.eks. maskinretningen om styringen af et maskin-projekt eller fremstillingen af en lille svejser til samlinger af produkter af f.eks. ståltråd eller metalfolie.

Opgavebeskrivelsen er, at vi skal udvikle et apparat der kan anvendes og give nytte på skolen i en undervisningssituation. Her skal vi klarlægge i hvilke situationer apparatet skal anvendes, og hvilke krav man må stille til udstyret såvel sikkerhedsmæssigt som til funktionen og brugergrænsefladen. I kan vælge at fremstille hele apparatet eller dele af det.

## Vejen fra projektoplæg til produkt

### Motivation

Projektets udformning bærer præg af gruppens eget ønske om at hjælpe fremtidige elektronik-elever med at sortere og finde ønskede modstande til deres projekter.

Derudover kan dette projekt også hjælpe Odense Tekniske Gymnasium med at mindske deres forbrug af modstande, og dermed både mindske økonomiske ressourcer og klimapåvirkning ved et overforbrug af ressourcer. Dermed bliver det også nemmere for elever og undervisere at finde og lægge fundne modstande tilbage. Selvom modstande ikke er det dyreste komponent hverken økonomisk eller på klima-kontoen, så vil dette projekt stadig gøre en forskel for skolens økonomi og klimapåvirkning i sidste ende.

Når man arbejder på et elektronikprodukt, er der ofte behov for at sortere og vælge mellem mange forskellige modstande. Hvis man skal gøre det manuelt, kan det tage meget lang tid, især hvis der er mange forskellige modstande at vælge mellem. En sorteringsmaskine kan gøre processen meget mere effektiv ved at sortere modstandene hurtigt og nøjagtigt. I mange tilfælde er det også vigtigt at vælge modstande med en meget præcis værdi.

En sorteringsmaskine kan hjælpe med at sikre, at modstandene er korrekt sorteret og opfylder de ønskede tolerancer og specifikationer. Derudover kan en sorteringsmaskine sikre, at modstandene sorteres på en ensartet måde hver gang, hvilket kan forbedre kvaliteten af det endelige produkt.

Hvis manuelle sorteringsmetoder anvendes, kan forskelle i teknikken og menneskelige fejl føre til uforudsigelige eller uønskede resultater, da farverne skal aflæses. Desuden kan en sorteringsmaskine automatisere en del af produktionsprocessen og reducere behovet for manuel arbejdskraft. Dette kan føre til besparelser på tid og omkostninger.

## Idégenerering

### Brainstorm af idéer fra hele el-holdet

- Overvågning i klasserum
- Fjernstyret lås
- Sensor til UR robot
- Infinite money generator
- Digitalt multimeter

### Gruppens egne idéer fra brainstorm

- Modstandssorteringsmaskine
- Coil gun
- Rubikscube-solver
- Identificering af komponenter - Maskine
- Elektronisk Legetøjsmus til katte

## Omvendt brainstorm

### Problemer:

1. Modstande bliver ikke sorteret
2. Komponenter bliver ikke sorteret
3. Tidskrævende at finde pinouts til alle anvendte komponenter

**Løsninger:**

1. Modstandssorteringsmaskine
2. Identificering af komponenter
3. Tegning af komponenters pinout ved indtastning af komponent bliver vist på en LCD-skærm

**Overordnede krav til vurdering af idéer**

1. Produktet skal kunne nå at blive fremstillet og produceret med omhu inden for eksamens-projektets tidsramme.
2. Produktet skal kunne blive udviklet med de tilgængelige materialer i El-lab og indenfor budgetrammen på maksimalt et beløb på kr. 500,00 inkl. moms pr. elev.  
(1.500 kr i alt)
3. Produktet skal kunne være realistisk og overholde basale fysiske love og kunne lade sig gøre på nuværende tidspunkt ved brug af simple elektronik komponenter.
4. Produktet skal stemme overens med det udvalgte projektoplægs krav og opgavebeskrivelse.
5. Produktet skal inkludere programmering af PIC16F877a og kræve brug af skolens 3D-printer efter gruppens egen interesse i udviklingen af eksamensprojektet.

**Sortering og vurdering af idéer**

Med udgangspunkt i om idéerne stemmer overens med ovenstående krav fremstilles følgende liste af endelige idéer:

1. Modstandssorteringsmaskine
2. Fjernstyret lås
3. Coil Gun

Herefter udvælges den endelige idé efter vejledning fra projektets vejleder og diskussion blandt gruppens medlemmer, om hvilken idé kunne være mest interessant at udvikle. Desuden skulle idéen også være udfordrende nok for at passe til et eksamensprojekt.

## Idéer under hovedidéen: "Modstandssorteringsmaskine"

Elektroniske idéer:

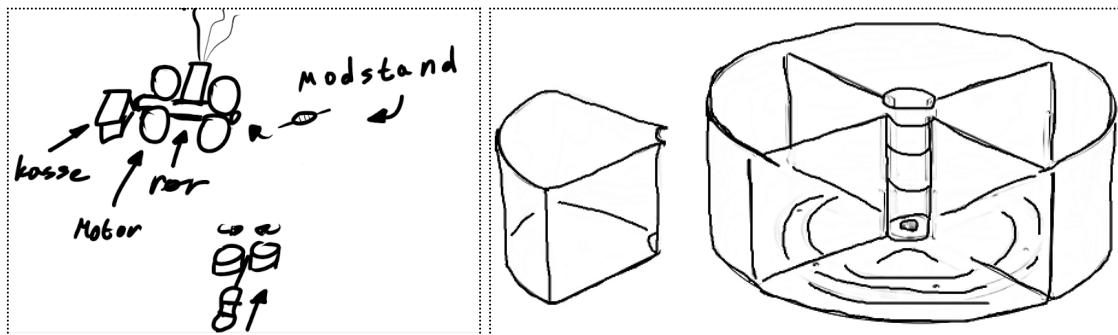
- LCD-skærm, der viser:
- Modstandens værdi
- Antallet af forskellige modstande
- Antal penge, der er sparet pga. denne sortering
- Alarm, hvis der mangler en bestemt type modstande
- Color-sensor, der genkender modstandenes farvekoder
- En mekanisme, der kan måle modstandens værdi

Mekaniske idéer:

- En måde hvorpå modstandene kan rettes ud og udlignes, så de kan identificeres.
- En lille robotarm, der kan sortere modstandene i respektive kasser
- Et rullebånd, der kan dreje modstandene ned mod den rigtige respektive kasse

## Den endelige løsning

Gennem mange idégenerering- og sorteringsmetoder blev den endelige idé og løsning til denne problemstilling fundet: Modstandssorteringsmaskinen.



Den første skitse beskriver selve processen, hvor modstanden bliver målt ved at blive fastholdt mellem to kobberplader i hver side af modstanden.

Når modstanden er blevet identificeret, vil den blive transporteret til i en af de respektive ved brug af en servomotor til at dreje rummene rundt.

Den anden skitse viser sorteringen af de forskellige modstande i forskellige rum, der kan tages ud og transportere modstandene til deres respektive kasser.

## Produktets funktioner

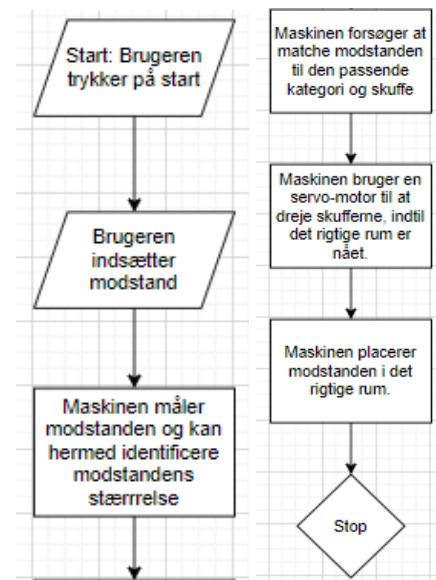
### *Identifikation af produkt med blokdiagram*

Produktet skal have to overordnede funktioner, som indebærer den praktiske sortering af modstande og den mere brugervenlige funktion ved brug af LCD-skærm.

### **Flowchart over modstandssorteringsmaskinens funktioner**

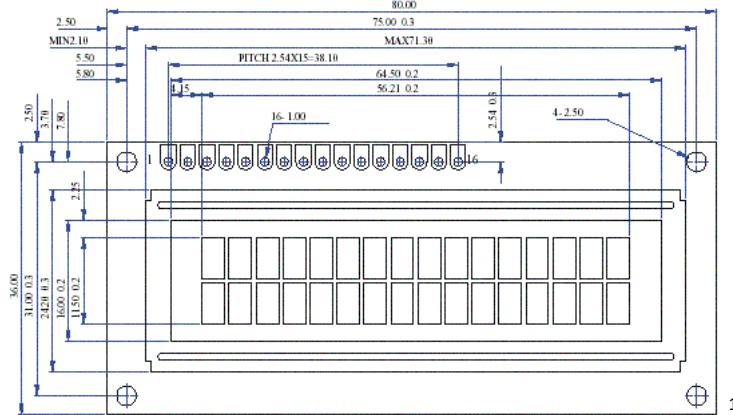
1. Start: Maskinen starter, når brugeren tænder den.
2. Indsæt modstand: Brugeren indsætter modstanden i maskinen.
3. Mål modstanden: Maskinen fastholder modstanden mellem to kobberplader for at måle dens størrelse.
4. Find den passende kategori: Maskinen sammenligner modstandens størrelse med de foruddefinerede størrelseskategorier og identificerer den passende kategori.
5. Brug af servomotor: Maskinen bruger en servomotor til at dreje rummene, indtil det rigtige rum er nået.
6. Sortering af modstande: Maskinen placerer modstanden i det rigtige rum.
7. Gentag processen for alle modstande: Maskinen gentager processen for alle modstande, der er blevet indsat i maskinen.
8. Stop: Maskinen stopper, når alle modstande er sorteret.

### *Selve flowchartet:*



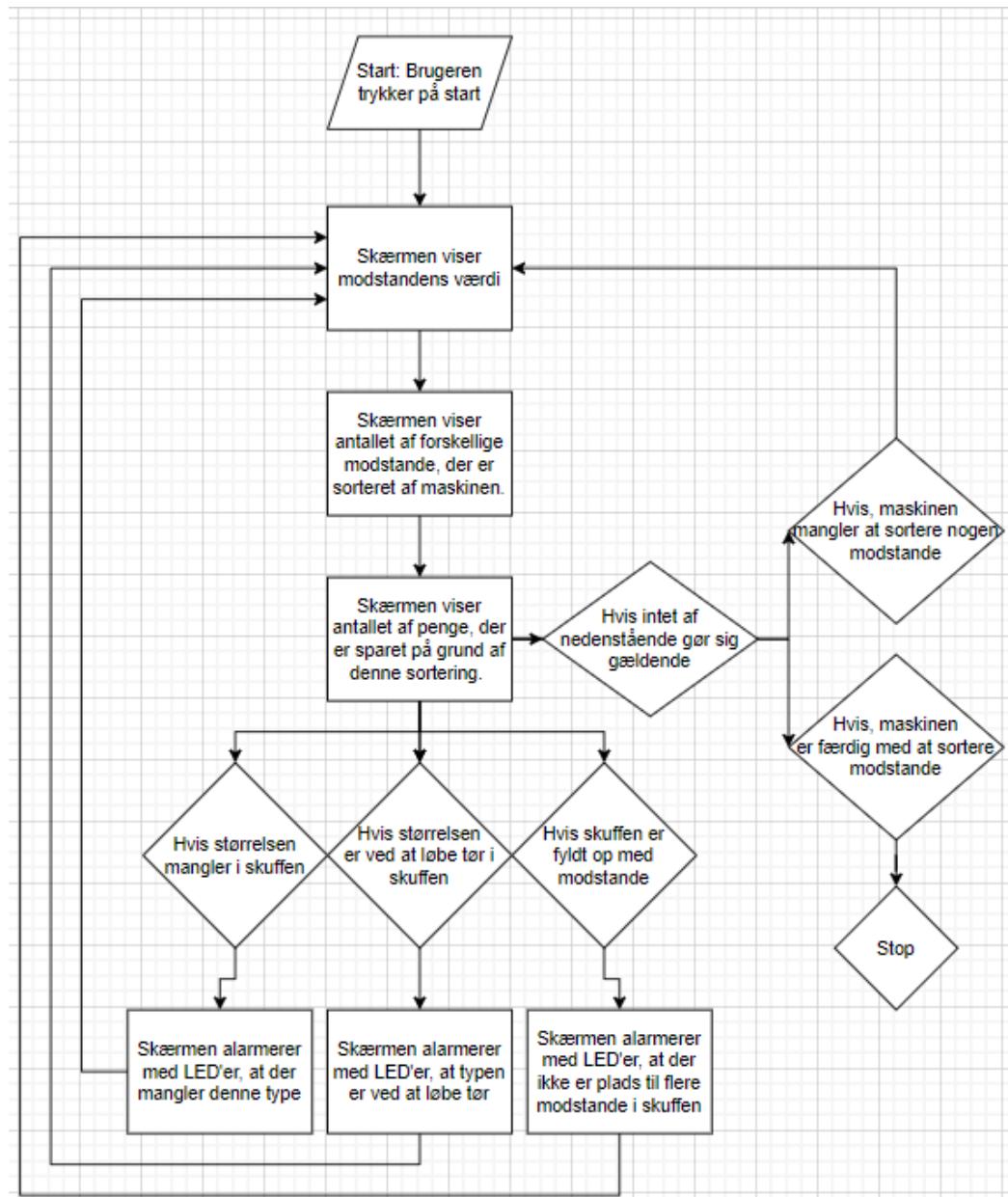
## Flowchart over LCD-skærmens funktioner

1. Start: Skærmen starter, når maskinen tændes.
2. Skærmen viser modstandens værdi, der blev målt af maskinen.
3. Skærmen viser antallet af forskellige modstande, der er sorteret af maskinen.
4. Skærmen viser antallet af penge, der er sparet på grund af denne sortering. Dette tal beregnes ved at multiplicere antallet af sorterede modstande med deres individuelle pris.
5. Tjek for manglende modstande: Skærmen tjekker, om der mangler en bestemt type modstand, der er nødvendig for at fuldføre en angivet opgave.
6. Hvis der mangler en bestemt type modstand, viser skærmen en alarm og oplyser brugerom, at der mangler en specifik type modstand. Desuden kan alarmen også igangsættes, hvis den bestemte type modstand er ved at løbe tør.
7. Gentag processen: Skærmen gentager processen og viser de relevante oplysninger, indtil maskinen slukkes.
8. Stop: Skærmen slukkes, når maskinen slukkes.



Alt i alt, så sikrer skærmens visning af modstandens værdi, antallet af sorterede modstande og det sparede beløb en effektiv og pålidelig sortering af modstandene. Ydermere, tjekker skærmen for manglende modstande og advarer brugerom om dette, hvilket er med til at forhindre fejl i processen. Skærmen gentager processen og viser de relevante oplysninger, så længe maskinen er tændt. Altså bidrager skærmens funktioner til en mere pålidelig og effektiv proces, som sikrer, at de nødvendige modstande anvendes korrekt.

<sup>1</sup> LCD-skærm pinout, <https://www.futurlec.com/LED/LCD16X2.shtml>



Flowchartet beskriver LCD-skærmen, der løbende skifter mellem et udvalg af oplysninger omkring de modstande, der bliver sorteret. Hvis maskinen opdager bare én af de angivne fejl eller advarsler, så vil et bestemt antal af LED'er lyse.

Herefter vil maskinen fortsætte med at sortere resten af modstandene og gentage denne proces. Når maskinen er blevet færdig med at give de væsentlige oplysninger og eventuelle advarsler til brugeren, vil den autonome sorteringsmaskine slukke af sig selv.

Det kan overvejes om de oplysninger, der ikke er advarsler, skal vises på LCD-skærmen løbende og helt automatisk eller om brugeren manuelt skal skifte mellem de forskellige oplysninger efter eget ønske.

**ICO**

I	C	O
Knap til LCD	Pic16F877a	LCD
Knap til start af maskinen	Pic16F877a	LED & LCD
Modstandsmåling	Pic16F877a	LCD

I dette tilfælde er der tre input-komponenter: en knap til LCD, en knap til start af maskinen og en modstandsmåler. Disse input-komponenter giver brugeren mulighed for at interagere med maskinen og udføre forskellige handlinger. Control-komponenten er en Pic16F877a-mikrocontroller, der styrer og koordinerer forskellige funktioner i maskinen. Mikrocontrolleren fungerer som hjernen i maskinen og behandler input fra brugeren og modstandsmåleren, og styrer output-komponenterne. Der er tre output-komponenter i denne maskine: en LCD-skærm og en kombineret LED & LCD-komponent, der viser informationer til brugeren om modstande og andre relevante oplysninger. Derudover er der endnu en LCD-komponent, som sandsynligvis bruges som en form for intern display eller feedback til systemet. Sammen udgør disse input-, control- og output-komponenter en maskine, der er i stand til at måle modstande og sortere dem efter type og værdi, og som giver brugeren feedback omkring denne proces på en LCD-skærm og en LED/LCD-kombination.

*Identifikation af problemstilling bag produktet*

Problemstillingen bag produktet er at skabe et apparat, der kan fungere som et værktøj i undervisningssituationer. Specifikt ønsker projektet at udvikle en maskine, der kan sortere modstande efter deres værdi og placere dem i den korrekte kasse uden behov for manuelt arbejde. Dette kræver en kombination af analog og digital teknologi, programmerbar elektronik og softwareudvikling i programmeringssproget "C" med IDE'en "MatLab". Projektet fokuserer på at skabe en praktisk og anvendelig løsning, der kan hjælpe undervisere med at demonstrere elektroniske koncepter og teknologier på en mere interaktiv måde.

# Projektbeskrivelse

## Problemformulering

Der vil opstå en række ulemper og konsekvenser ved, at man ikke sorterer sine modstande eller genbruger dem efter brug i et el-projekt. Hvis man overhovedet ikke sorterer modstandene, kan det føre til øgede omkostninger for skolen. Dermed vil et overforbrug af disse komponenter også have en negativ indvirkning på miljøet.

På den anden side selv, hvis modstandene bliver sorteret manuelt, så kan der stadigt opstå lignende problematikker. Hvis modstandene ikke er sorteret efter de rigtige værdier, kan det føre til fejl og ustabilitet i det elektroniske kredsløb. Desuden kan det være ret ineffektiv og tidskrævende, at manuelt skulle aflæse modstandenes farvekoder og selv sortere dem i deres respektive kasser.

## Problemstilling

Disse problemer og konsekvenser ved et overforbrug af disse komponenter kan koges ned til en helt klar problemstilling. Dermed kan følgende spørgsmålet angives: **Hvordan kan man reducere spildet af elektriske modstande blandt elever og lærere i teknikfaget: El - Udvikling og Produktion A, ved at udvikle en autonom sorteringsmaskine, og dermed mindske overforbruget af disse komponenter?**

For at realisere ovenstående ønsker projektgruppen at finde yderligere svar på følgende spørgsmål:

- Hvordan kan blandt andet modstanderens værdi vises på en LCD-skærm, så brugeren nemt kan identificere modstandens størrelse?
- Hvilken form for automatiseringen kan bruges under dette projekt, så selve sorteringen foregår så vidt muligt autonomt?
- Hvad kan der gøres for at måle modstanders værdi, uden at skulle manuelt aflæse farvekoderne?

## Projektafgrænsning

Dette projekt kan afgrænses ved at sortere modstandene i følgende kategorier:

1. 100 Ω (Formodstand til 5 V til LED)
2. 220 Ω (Hyppigt anvendt)
3. 330 Ω (Formodstand til 9 V til LED)
4. 470 Ω (Formodstand til 12 V til LED)
5. Modstande lavere end 1k (Rest-skuffe 1)
6. 1k Ω (Hyppigt anvendt)
7. 2,2k Ω (Hyppigt anvendt)
8. 4,7k Ω (Hyppigt anvendt)
9. 5,6k Ω (Hyppigt anvendt)
10. Modstande højere end 1k (Rest-skuffe 2)
11. 10k Ω (Hyppigt anvendt)
12. Modstande højere end 10k (Rest-skuffe 3)

Denne afgrænsning sikrer, at der prioriteres og udvælges de modstande, der er hyppigst anvendt. I stedet for at man skulle kunne sortere alle modstande, fokuseres der på et udvalg af modstande, da det vil effektivisere produktudviklingen. Endelig vil denne afgrænsning stadig kunne sikre at opfylde alle de givne krav og løse problemstillingen.

## Kravspecifikation

### **Bløde krav:**

- Brugervenlighed: Maskinen skal være nem at bruge og betjene, selv for personer uden specialiseret teknisk viden.
- Fleksibilitet: Maskinen bør være i stand til at sortere en bred vifte af modstande og tolerancer.
- Pålidelighed: Maskinen bør have en høj grad af pålidelighed og nøjagtighed for at sikre korrekt sortering af modstandene.
- Sikkerhed: Maskinen bør være designet med passende sikkerhedsfunktioner for at undgå skader på brugere og maskinen selv.

**Hårde krav:**

- Præcision: Maskinen bør være i stand til at sortere modstande præcist med en høj grad af nøjagtighed.
- Hastighed: Maskinen bør kunne sortere modstandene hurtigt for at øge effektiviteten af produktionsprocessen.
- Kapacitet: Maskinen bør have tilstrækkelig kapacitet til at håndtere et stort antal modstande på én gang.

**Tekniske krav:**

- *Eventuelt Sensorteknologi:* Maskinen bør være udstyret med højteknologisk sensorteknologi, der er i stand til at registrere og identificere modstandsværdier præcist.
- Software og programmering: Maskinen bør anvende et programmerbart PICkit, der kan videregive oplysninger om de givne modstande, og sortere modstandene korrekt.
- Robusthed: Maskinen bør være robust og holdbar for at modstå gentagne brug og vedligeholdelse.

**Andre Specifikke krav:**

- Maskinen skal kunne tjekke om nogen af beholderne er fyldte, og derefter kunne advare brugeren om dette.
- Maskinen skal regelmæssigt blive nulstillet, så offset i servo-mekanismen ikke bliver for stor.
- Maskinen skal kunne være i stand til at beregne prisen og antallet af modstande i systemet.
- Beholderne skal let kunne fjernes og lægges i den ydre 3D-printede beholder.
- Advarsler skal vises på både LCD-skærmen og ved blinkende LED'er.

# Planlægning af projektet

## Fremgangsmåde

Følgende fremgangsmåde er de punkter, som følges helt fra starten af projektet til at danne et overblik over projektets udformning med undersøgelse af komponentens mange forskellige indstillinger. Denne oversigt implementeres også yderligere i en konkret modulbaseret tidsplan.

1. Opsætning af breadboard med PIC16F877a
  2. Print af 3D-prints
  3. Implementering af servomotor til breadboardet
  4. Print PCB
  5. Sammensæt hele projektet
- 

## Teori og fremgangsmåder

### Samspil med andre fag

#### *Programmering*

Gennem viden fra programmeringens faget kan vi programmere microcontrollere som PIC16f877A. Programmeringen har givet os muligheden for at kunne læse og forstå de fleste programmeringssprog. Dette giver derfor et godt samspil da programmeringen og EL-tek er tæt forbundet. De faglige mål til begge fag er der også meget samspil mellem. Der fokuseres på problemstillinger og hvordan disse problemer løses gennem værktøjer og viden fra faget.

#### *Matematik*

Matematik har samspil med dette eksamsprojekt, da der foretages beregninger af diverse ting. I vores tilfælde vil vi komme til at bruge Ohms lov hyppigt og derfor er matematik et godt fag at indblande i vores projekt. Et af de faglige mål for matematik er brugen af hjælpemidler, hvilket er noget vi også kommer til bruge gennem dette projekt. Alt i alt vil matematik give et godt samspil med både El-tek og Programmeringen.

## Valg af komponenter

Dette projekt involverer brugen af forskellige elektroniske komponenter og værktøjer. Ledninger bruges til at opbygge breadboard og vias på printplader. Modstande beregnes ud fra Ohms lov for at bestemme den ønskede strømstyrke. LED'er og dioder bruges til at lyse op i forskellige farver og ensrette spænding. Komparatorer bruges til at identificere modstandsværdier. Kondensatorer bruges til at sikre stabilitet og kontrol over spændingen. Potentiometre bruges som varierende modstand, der kan justeres. Derudover er der andre materialer såsom loddestation og hullemaskine, skævbider, krokodillenæb, multimeter og strømforsyning, der også bruges i projektet.

## Valg af teknologier

**Blender og SketchUp:** De er 3D-modelleringsprogrammer, som benyttes til at lave 3D modeller, som kan eksporteres i .stl formatet, hvilket kan 3D-printes.

**MPLab:** MPLab er den teksteditor vi bruger til at kode alt vores kode i. MPLab kan overføre koden, vi skriver i 'c' til vores Pickit.

**Pickit:** Et Pickit bruges til at overføre koden til vores PIC16F877A. Pickit kobles sammen med computer og PIC16F877A kan derefter anvende den skrevne programmering.

**diagrams.net:** Diagrams.net bruges til at fremstille flowcharts og blokdiagrammer til at beskrive kode og andre funktioner der leder til noget andet.

**3D-printer:** 3D-printing bruges ofte i produktion til at fremstille prototyper, men det kan sagtens bruges til at skabe endelige produkter. Derfor bruger vi 3D-printing til at fremstille vores produkt.

**Creality Slicer:** Creality sliceren bruges til at forberede .stl og andre 3D filer til at blive printet gennem en Creality 3D-printer. Dette giver filen i formatet .gcode.

**KiCad:** KiCad bruges til at fremstille diagrammer over elektriske kredsløb og til fremstilling af footprints som bruges til at lave PCB som derefter skal have komponenter loddet på.

**SCRUM:** Det fysiske projektstyringsværktøj, SCRUM, benyttes til at følge op på opgaver og danne et overblik over projektets forløb.

**Google Sheets:** Google Sheets anvendes til at lave en tidsplan og oversigt over projektets tidsramme.

## Analyse af valgte teknologier

### KiCad

I løbet af projektet anvendes KiCad til at først og fremmest tegne diagramtegninger af breadboardet for at dokumentere det elektriske kredsløb. Derudover anvendes KiCad også i høj grad til at fremstille PCB-print fra diagramtegningerne, for at mindske måleusikkerheder og fejlkilder som opstår ved anvendelse af et almindeligt breadboard. For at fremstille optimale KiCad tegninger til PCB-print med udgangspunkt i OTGs bibliotek til KiCad, skal man foretage en række af de følgende handlinger i løbet af fremgangsmåden. For det første optimeres PCB-printet ved at forsøge at fjerne så meget kobbertråd (det sorte i KiCad) fra PCB-tegningen. For det andet anvendes også i dette tilfælde 1,2 mm spændingsregulator, da det stemmer overens med de tilgængelige spændingsregulatorer i El-lab, og sikrer at komponentet kan sættes på PCB-printet. For det tredje anvendes netop et PCB-board for at sammensætte alle de anvendte komponenter i et komplekst system, der skulle lave en servomekanisme ved brug af LM555.

Desuden kan man for at optimere selve produktionen af PCB-printet i KiCad benytte følgende af de mest væsentlige genveje på sin computer:

- Alt + 3: Inspect, som giver dig mulighed for at se en 3D-modellering af printet.
- e: Edit, der bruges til at redigere dele af diagramtegningen.
- m: Move, der bruges til at flytte komponenter eller andre dele af diagramtegningen.
- a: For at åbne KiCad-biblioteket for at kunne indsætte nye komponenter i kredsløbet.

For at tjekke om man har lavet nogen fejl i sit PCB-print, skal man først tjekke efter på følgende måde:

- Trykke på knappen: "Inspicere"
- Trykke på knappen: "Design Rules Checker"
- Trykke på knappen: "Kør DRC"

'Design rules checker' benyttes netop til at debugge eller tjekke efter fejl på PCB-printet.

## Fremstilling af PCB

PCB står for Printed Circuit Board, på dansk kaldet trykt kredsløbskort. Et PCB er en plade af et isolerende materiale, som er dækket af et lag kobber på begge sider. Kobberet er ætset væk på de steder, hvor man ikke ønsker, at der skal være forbindelse, mens det forbliver på de steder, hvor man ønsker, at der skal være forbindelse mellem elektroniske komponenter.

For at fremstille et PCB-print fra sin KiCad-tegning skal man tage følgende forholdsregler og udgangspunkt i en væsentlig fakta, der også nævnes under selve processen. For at PCB-printet kan fremstilles bruges først en lysmaskine, der påvirker den lysfølsomme lakering på kobberpladen, der vil fjernes ved kontakt med basen og syren. For at kun det ønskede kobberlag fjernes, lægges et billede af PCB-printet nederst på glaspladen i denne lysmaskine, så det resterne kobberlag ikke bliver påvirket.

Når man herefter skal placere kobberpladen i henholdsvis basen, vand og syren, er det vigtigt at have det rigtige sikkerhedsudstyr på. Især når man arbejder med så stærke syrere og baser er det vigtigt at have sikkerhedsbriller, sikkerhedshandsker og et forklæde, der er specifikt designet til kemi-laboratorier, på. I forhold til hvor lang tid denne fremstillingsproces tager, så kan det variere lidt efter printets størrelse, men hele processen tager nok omkring et kvarter.

Man skal derfor også tjekke kobberet løbende, og især når den er i syren, for at man ikke giver den mere tid end nødvendigt. Man kan afgøre dette ved, at kobberet skal være gennemsigtigt i de ønskede områder, når det er færdigt. Der er intet, der vil tage mere end en halv time, og efter en time vil det ønskede kobberlag også gå i opløsning og ødelægge PCB-printet.

Slutteligt skal man belyse printet en gang mere i et minut og derefter placere den i syren igen i omkring 30 sekunder for at sikre, at al kobberlakken er ætset væk. Hvis man undlader at udføre dette sidste step, kan man risikere, at der ikke er god nok forbindelse mellem komponenterne og printet.

# Produktudvikling

## Opstart af projektet

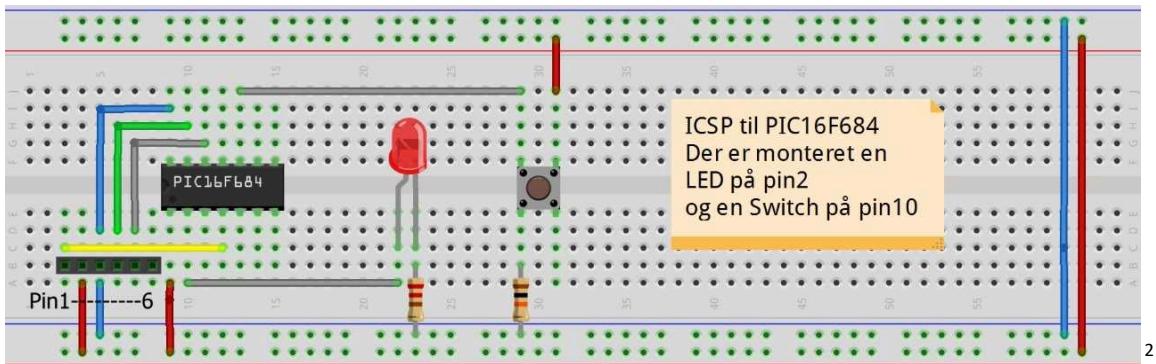


Når man starter et projekt, er det vigtigt at have en plan for, hvordan man vil indsamle de nødvendige komponenter og hvordan man vil opsætte projektet. I dette tilfælde skulle der indsamles de nødvendige modstande og programmeringen skulle opsættes i C i MPLab. Derudover skulle breadboardet også opsættes korrekt, således at alle de nødvendige komponenter kunne tilsluttes og fungere sammen.

Når alle komponenterne var indsamlet og testet, kunne programmeringen i C i MPLab opsættes korrekt og breadboardet kunne tilsluttes og testes for at sikre, at alle komponenter fungerede sammen. Ved at gøre dette sikrede man sig, at projektet blev startet korrekt og at man undgik eventuelle fejl eller problemer, der kunne opstå senere i processen. For at sikre, at den udvalgte servomotoren virkede, blev der også sat en lille servo motor tester op, som kunne bruges til at teste servoer, inden de blev implementeret i breadboardet. Testen blev udført ved hjælp af en Arduino Uno.

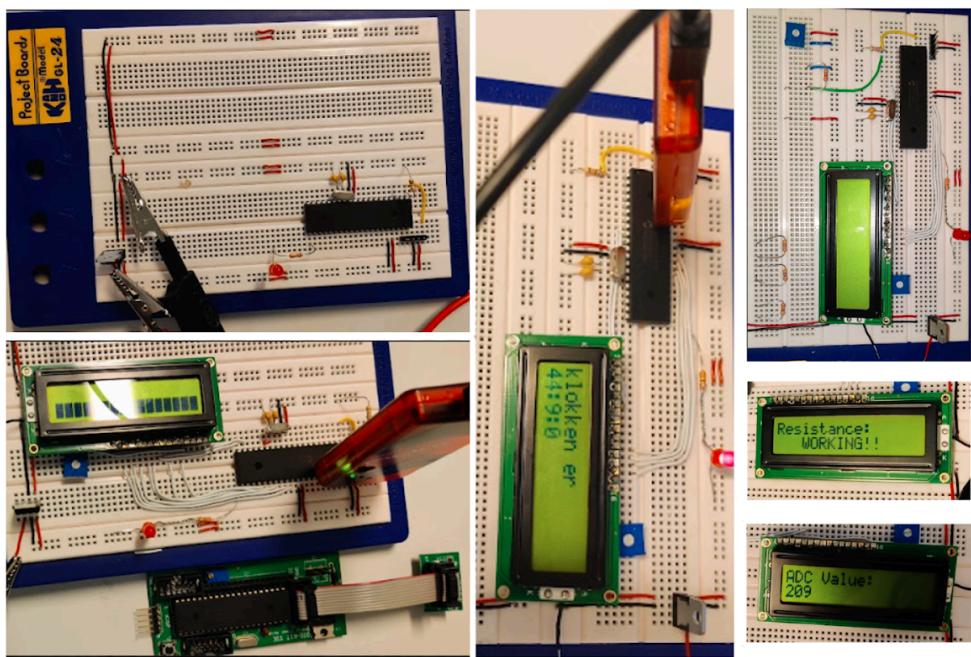
Når alle komponenterne var indsamlet og testet, kunne programmeringen i C i MPLab opsættes korrekt og breadboardet kunne tilsluttes og testes for at sikre, at alle komponenter fungerede sammen. Ved at gøre dette sikrede man sig, at projektet blev startet korrekt og at man undgik eventuelle fejl eller problemer, der kunne opstå senere i processen.

### ICSP til PIC16F684



Først og fremmest tog gruppen udgangspunkt i tegningen af ICSP til PIC16F684, da opsætningen i nærværende opgave med PIC16F877A havde nogenlunde samme opsætning som den ovenstående tegning.

### Billeder af den første opsætning af breadboardet



Opsætningen af LCD-skærmen med microcontrolleren, PIC16F877A, blev foretaget i sammenhæng med tidligere erfaring med denne opsætning og inspiration fra den ovenstående illustration af ICSP til PIC16F684. Først blev programmeringen fra Timer-projektet genbrugt, og herefter modificeret efter projektets nye behov, krav og funktioner. På de sidste billeder vises vores LCD som er tændt og allerede på daværende tidspunkt kunne måle værdien af modstandene.

<sup>2</sup> [https://learn.sde.dk/pluginfile.php/1052489/mod\\_resource/content/1/pic16f684ICSP.jpg](https://learn.sde.dk/pluginfile.php/1052489/mod_resource/content/1/pic16f684ICSP.jpg)

## Valg af microcontroller

I projektet anvendes microcontrolleren

PIC16F877A i sammenhæng med

anvendelsen af en LCD-skærm og til at

foretage en analog-digital konvertering

mellem digital programmering og analogt

PCB-board eller breadboard. For det første

har PIC16F877A en indbygget 8-bit parallel

grænseflade (PP) og kan let oprette

forbindelse til en LCD-skærm uden behov for ekstra hardware. Dette gør det lettere at

implementere en brugergrænseflade og vise data på skærmen. For det andet er

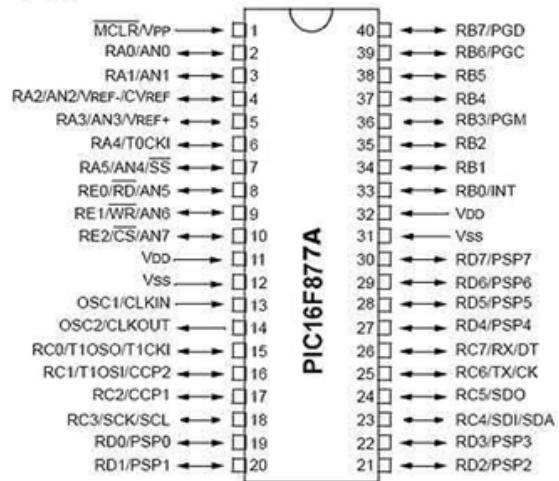
PIC16F877A udstyret med en indbygget 10-bit ADC, der kan konvertere analoge signaler

til digitale værdier med høj præcision. Selvom en PIC16F877 også har mulighed for at

lave en 10-bit ADC, har den en indre krystal på 8 MHz, og projektet kræver en 16 MHz for

at fungere optimalt og mere præcist til at opfylde alle projektets krav.

Når det kommer til at arbejde med C# og PIC16F877A, er der en række udviklingsmiljøer og biblioteker til rådighed, der kan gøre det lettere at programmere og kommunikere med mikrocontrolleren. Dette kan gøre det lettere at implementere en kommunikation mellem mikrocontrolleren og en computer eller anden ekstern enhed.



## Brug af udvalgte registre i projektet

TRISB, TRISC, TRISD, og TRISE-registrene bruges til at indstille retningen af porte på microcontrolleren og i dette tilfælde indstilles de til output. ADCON0 og ADCON1-registrene styrer Analog-Digital Converteren, mens ADRESL og ADRESH-registrene gemmer resultatet af konverteringerne. GO\_DONE er en bit, der sættes i ADCON0 for at starte en konvertering og sikre, at ADC'en fungerer korrekt. Sammen bidrager disse registre til en effektiv og nøjagtig modstandssorteringsproces ved brug af microcontrolleren Pic16F877a.<sup>3</sup>

TRISA	85h
TRISB	86h
TRISC	87h
TRISD <sup>(1)</sup>	88h
TRISE <sup>(1)</sup>	89h

ADRESL	9Eh
ADCON1	9Fh

ADRESH	1Eh
ADCON0	1Fh

<sup>3</sup> <https://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf>

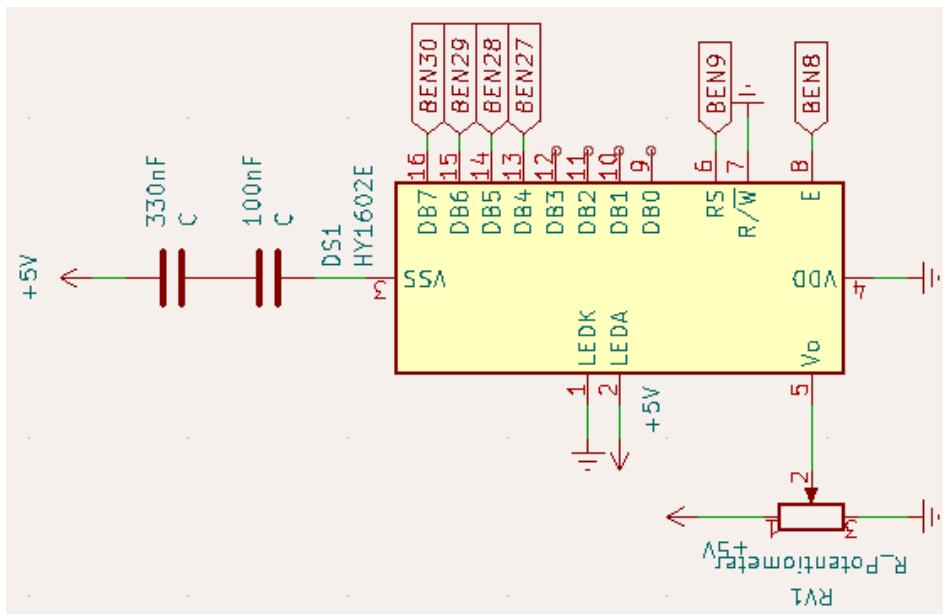
## Regulering af indgangsspænding

Når en spænding højere end 5V påføres PIC16F877A's indgang, kan det forårsage skade på mikrocontrolleren. For at beskytte den mod for høje spændingsniveauer kan man anvende TIP120 som en spændingsregulator, der vil fungere som en slags buffer mellem indgangsspændingen og PIC16F877A.

TIP120 er en NPN Darlington-transistor, der kan anvendes til at regulere indgangsspændingen til PIC16F877A ved at fungere som en spændingsregulator. Ved at koble en passende størrelse capacitor parallelt med indgangen til TIP120, kan spændingen jævnes ud og holde sig på et mere stabilt niveau, hvilket sikrer mere præcis måling af modstande. TIP120 kan styres med en lav strøm og har en høj gain, hvilket betyder, at en lille strøm kan tænde transistoren, og dermed kan større strømme passere igennem den til PIC16F877A. Ved at tilføje en passende størrelse capacitor parallel med indgangen til TIP120, vil den jævne spændingen og sikre, at den forbliver på et mere stabilt niveau, selv når indgangsspændingen varierer.

Denne løsning kan hjælpe med at beskytte PIC16F877A mod skader forårsaget af for høje spændingsniveauer og samtidig sikre en mere præcis måling af modstande.

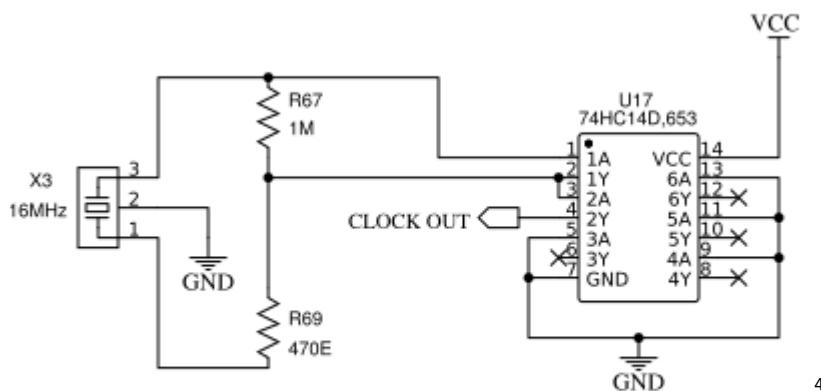
*Kredsløbet for denne spændingsregulering ses herunder:*



## Anvendelsen af en 16 MHz krystal

Krystaller bruges til at generere præcis frekvens i elektroniske kredsløb, og valget af frekvens har både fordele og ulemper afhængigt af applikationen. En højere frekvens kan øge ydeevnen og hastigheden, men kræver mere strøm og kan medføre øget støj. En lavere frekvens kan reducere strømforsyningen og støjen, men også reducere ydeevnen og hastigheden.

Ved valget mellem en 16 MHz krystal og en lavere frekvens kan fordelene ved en højere frekvens være mere værdifulde, hvis hastigheden er afgørende. Hvis strømforsyning og varmeafgivelse er et problem, kan en lavere frekvens være mere passende, medmindre krystallen er direkte tilsluttet til en strømforsyning, hvor hurtigere hastighed er vigtigere end lavere strømforsyning.



Her er et eksempel på, hvordan et kredsløbsdiagram kan indeholde en 16 MHz krystal til at producere et 16 MHz firkantet signal (puls), der kan bruges til at videresende præcise signaler til oscillatoren.

I nærværende projekt anvendes en krystaloscillator på en frekvens på 16 MHz, da det er en passende frekvens for en PIC16F877A microcontroller, da den er i stand til at give en høj ydeevne og præcision i timing af dens interne processer. Servomotorer kræver præcis timing af signalerne, der sendes til dem for at opnå den ønskede position og hastighed. En høj frekvens på 16 MHz betyder, at mikrocontrolleren er i stand til at generere disse signaler med høj nøjagtighed og præcision, og at servomotoren kan styres præcis. Det er nemlig vigtigt at servomotoren kan gå til en bestemt og præcis position, som stemmer overens med en af de tolv beholdere i sorteringen.

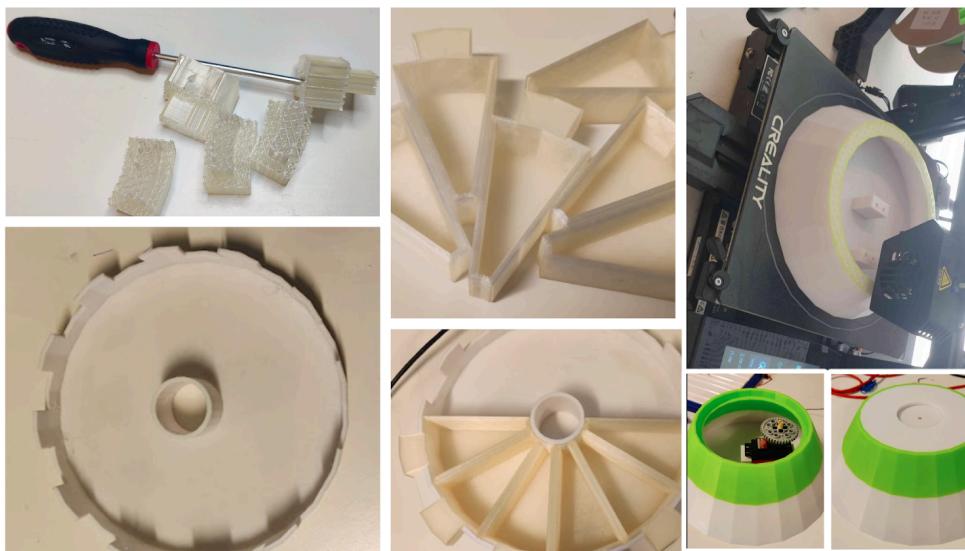
<sup>4</sup> <https://0creativeengineering0.blogspot.com/2019/06/simple-16-mhz-crystal-oscillator.html>

## Anvendelsen af LEGO

Alle LEGO klodser er fremstillet af ABS-plast, hvilket er en meget stærk og holdbar plasttype. Vi har valgt at bruge LEGO tandhjul til vores produkt, da styrken LEGO tandhjul besidder ikke ville kun genskabes af 3D-printede elementer. Dette er fordi 3D-printede elementer er opbygget en linje af gangen og derfor har den ikke den samme styrke som støbt ABS-plast. I sammenhæng med vores servo-motor bliver der brugt LEGO tandhjul til at bevæge de forskellige led, der ses i vores modstand sorteringsmaskine. Derudover bruges der LEGO hjul som erstatning for et transportbånd.



## Anvendelsen af 3D-print



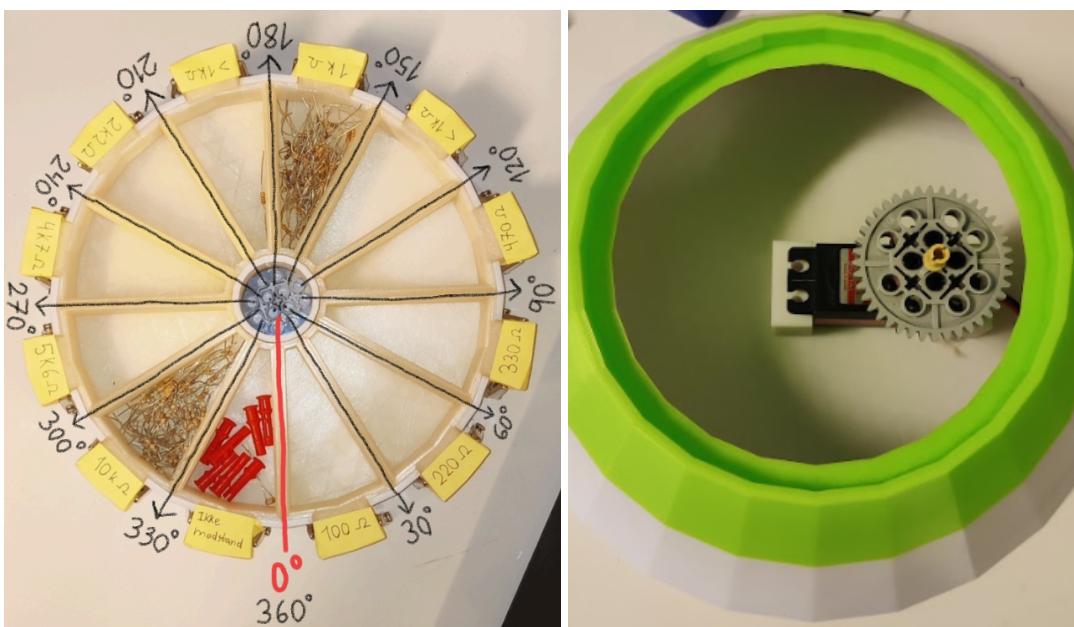
På billedet her ses udviklingen af vores forskellige 3D-print, og hvordan de så kobles sammen til et sammensat produkt.

Det var ikke muligt at anvende LEGOklodser til alle vores elementer i produktet, så vi blev også nødt til at designe en hel del 3D-prints til vores modstandssorteringsmaskine.

Det endelige produkt indeholder en lang række 3D-printede elementer, som vi selv har designet og tilpasset til at kunne passe perfekt sammen. Mens det ville være muligt at genskabe dele af produktet ved hjælp af LEGO, ville dette medføre, at nogle af de fordele, som 3D-printing giver, såsom muligheden for at designe og printe helt personlige tegninger, ville gå tabt. Derfor valgte vi at starte med at designe beholdere, der er specielt designet til at opbevare modstande. Disse beholdere er designet, så de kan samles i en cirkel, hvis de placeres side om side. Cirklen af beholdere kan derefter placeres i vores ramme, som holder alle 12 beholdere til modstande. Derfor krævede det, at vi skulle designe både beholderne og den ydre ramme, så der er en individuel skuffe til hver modstand og deres respektive værdi.

Den ydre beholder fungerer som en drejeskive, der kan dreje alle de andre beholdere rundt og give mulighed for automatisk sortering. Til at holde vores 12 beholdere og den ydre beholder har vi valgt at 3D-printe basen. Denne base er udstyret med en servo motor, som gør det muligt at dreje den ydre beholder og dermed også de 12 beholdere. Denne rotation er afgørende, når vi skal placere modstandene i den tilsvarende skuffe, og til dette formål har vi designet et 3D-printet element, der fungerer som en tragt og sørger for, at modstandene ender, hvor de hører hjemme.

*Roteringen af beholderne ved brug af servomotor:*



## Identificering af modstande

For at identificere en resistor (modstand) i et elektrisk kredsløb anvendes spændingsdelerformlen i et spændingsdelerkredsløb med følgende formel:

$$\bullet \quad V_{out} = V_{in} \cdot \left( \frac{R_2}{R_1 + R_2} \right)$$

Her er  $V_{in}$  er den indgangsspænding, der påtrykkes spændingsdelerkredsen, og  $V_{out}$  er spændingsfaldet over  $R_2$ . En 4k7 Ohm modstand anvendes i dette tilfælde som  $R_2$  i spændingsdelerkredsløbet, da det er nogenlunde mellem 1k og 10k ohm, så målinger mellem disse to områder vil være mere nøjagtige. En modstand på 4k7 ohm er passende til dette område og giver rimelig præcision og opløsning i målingen. En for lille kendt modstand i forhold til den ukendte modstand kan føre til højere usikkerhed i målingen, mens en for stor kendt modstand kan føre til lavere opløsning i målingen. For at holde vores målinger præcise uden for dette område bruger vi yderligere to modstande, som vi kan tænde for at få en parallelforbindelse og effektivt ændre modstanden.

## Præcisering af målingen af modstande

Når man arbejder med modstandsmåling i en microcontroller, kan det være en fordel at anvende et 10-bit-system i stedet for et 8-bit-system for at få mere præcision i målingerne. Hvis man anvender et 8-bit system, vil man kun have 256 forskellige værdier til rådighed, når man mäter modstande. Dette kan være tilstrækkeligt i visse applikationer, men hvis man ønsker større præcision i målingerne, kan man overveje at anvende et 10-bit system. Et 10-bit system giver derimod 1024 forskellige værdier til rådighed, hvilket betyder, at man kan opnå en langt mere præcis måling af modstande, der ligger i intervallet mellem 0 og 1023. Dette kan være særligt relevant, hvis man arbejder med modstande, der har en lille variationsbredde, eller hvis man ønsker at kunne måle forskelle på meget små modstandsværdier. PIC16F877a microcontrolleren har indbygget en 10-bit ADC (Analog to Digital Converter), hvilket betyder, at den er i stand til at håndtere 10-bit målinger. Ved at anvende dette 10-bit system vil man derfor kunne opnå en langt mere præcis måling af modstande, og dermed øge nøjagtigheden og pålideligheden af ens målinger.

## Andre idéer til præcision af modstands-målingen

### *Idé 1: Udviklingen af flere spændingsdelerkredsløb*

En 1k ohm modstand kan bruges som  $R_2$  i et spændingsdeler-kredsløb for at måle en modstand under 100 ohm, da spændingsfaldet over  $R_2$  vil være større og mere målbart.

En 100k ohm modstand kan bruges som  $R_2$  for at måle en modstand over 10k ohm, da spændingsfaldet over  $R_2$  vil være mindre og mere målbart. En spændingsdeler-kredsløb kan bruges til at måle en ukendt modstand, men spændingsfaldet over den ukendte modstand kan være for lille eller for stort, og derfor kan forstærkning af signalet være nødvendigt for at opnå præcise målinger.

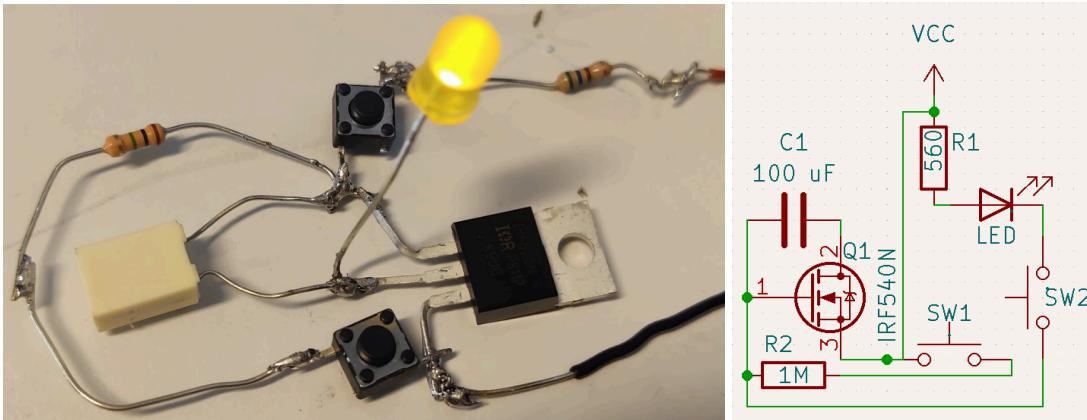
### *Eventuel anvendelse af en Spændingsforstærker*

For at kunne måle meget små spændingsfald over en meget lille modstandsværdi, kunne en spændingsforstærker også implementeres i kredsløbet. En spændingsforstærker er en elektrisk kreds, der kan forstærke et svagt elektrisk signal og øge dets amplitude til et niveau, der kan aflæses og behandles af en controller eller en anden elektrisk enhed. På samme måde kan en spændingsforstærker også bruges til at dæmpe et meget stort spændingsfald over en meget stor modstandsværdi og derved gøre signalet mere målbart.

### *Idé 2: Variabel modstand med MOSFET*

Video: <https://youtu.be/uJlIYiUP-J8>

En MOSFET kan fungere som en variabel modstand i et kredsløb. Dette skyldes, at MOSFET'en kan styres ved hjælp af en gate-spænding, som kan variere, og dermed kan modstanden også ændres. Ved at anvende en MOSFET som variabel modstand kan man undgå at skulle justere en fysisk potentiometer manuelt, når man ønsker at opnå en optimal modstand i et spændingsdelerkredsløb. Dette kan gøres ved at ændre på MOSFET'ens gate-spænding ved hjælp af en kontrolkreds, som kan tilpasse modstanden efter behov. Dette kan være en fordel i situationer, hvor man har behov for præcis styring af modstanden i kredsløbet og ønsker at undgå unøjagtigheder i modstandsværdierne, som kan opstå ved manuel justering af en potentiometer.

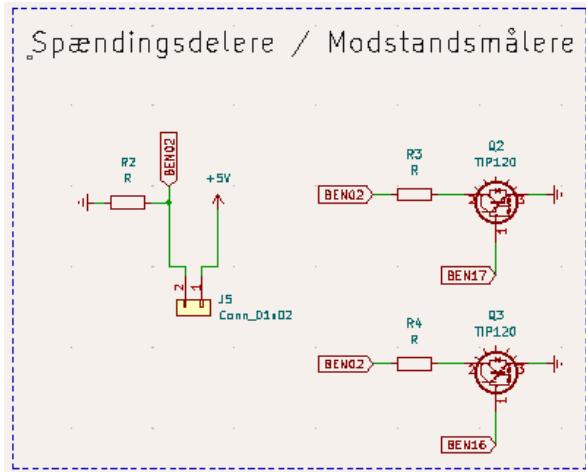


Hvis man aktiverer den første knap (SW1), vil LED'en blive tændt med fuld styrke. Hvis man derimod trykker på den anden knap, vil LED'en gradvist dæmpes for hvert tryk på knappen.

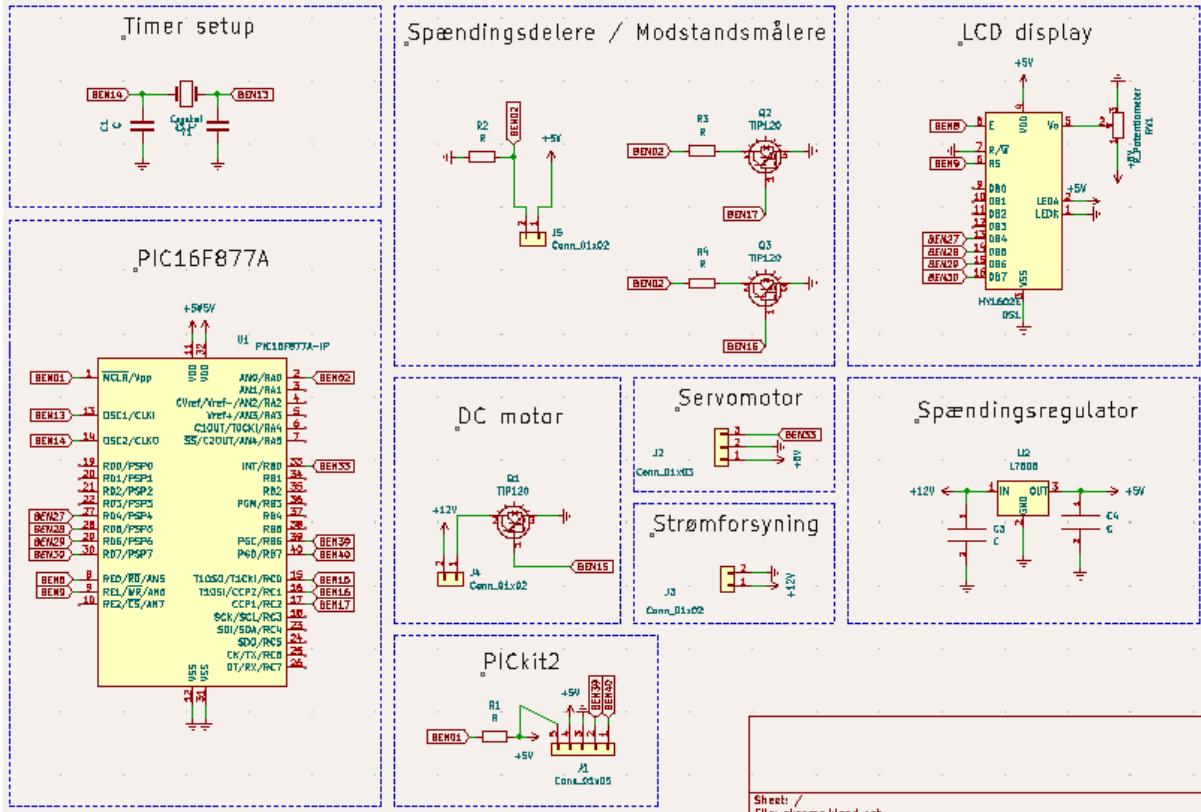
#### *Endelige idé: Parallelforbindelser i spændingsdelerne*

En parallelkobling af modstande giver flere fordele i forhold til forskellige spændingsdelerkredsløb. Ved at parallelkoble modstandene kan man opnå flere forskellige kendte modstande, uden at det kræver manuel flytning af modstanden mellem kredsløbene. Dermed kan man opnå større fleksibilitet og hurtigere målinger af modstande. Til at måle modstandene i parallelkoblingen bruger vi to yderligere spændingsregulatorer (TIP120). Disse regulerer strømmen gennem modstandene i parallelkoblingen, og sammen med spændingsdelerne bruges de til at måle spændingsfaldet over hver modstand og dermed beregne den ukendte modstand.

Samlet set resulterer denne tilgang i en mere effektiv og nøjagtig måling af modstande.

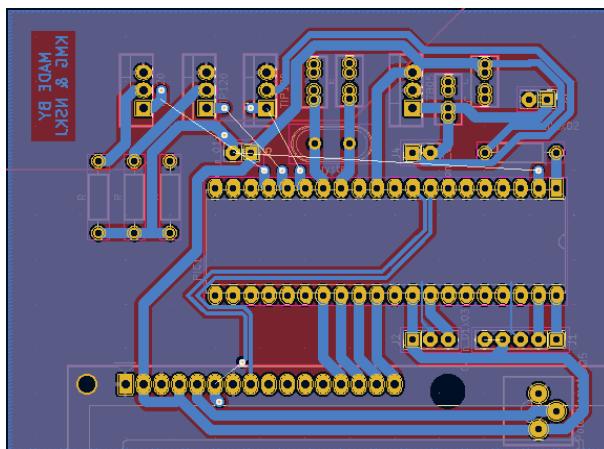


## KiCad kredsløb



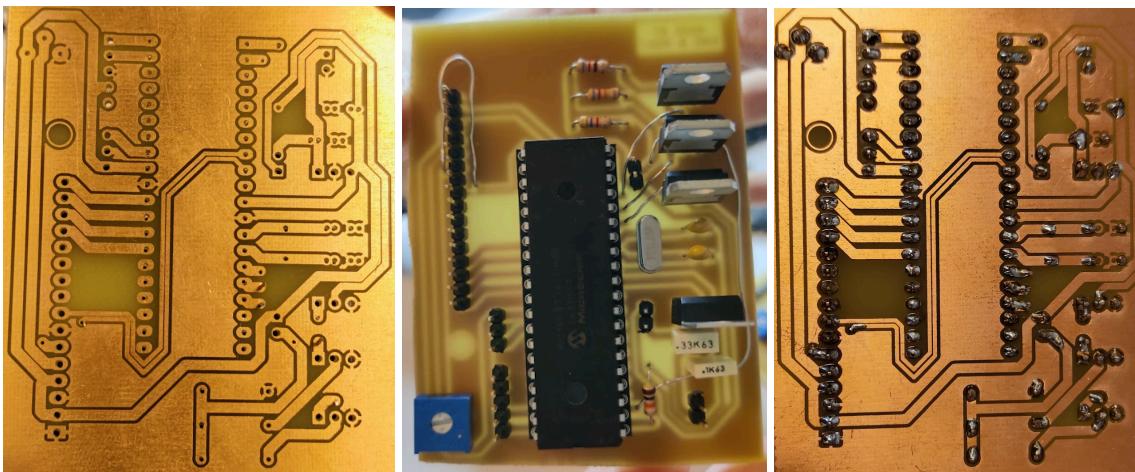
Her ses diagrammerne til vores kredsløb, som vi først har bygget på et breadboard. Dette diagram brugte i sammenhæng med fremstillingen af vores kredsløb på PCB. I stedet for at bygge det hele som samlet kredsløb, har vi valgt at inddale vores kredsløb i 9 sektioner, der viser hvordan de individuelle komponenter fungerer til kredsløbets samlet. Denne opsætning gør det også nemmere for os at inddale footprints på vores PCB.

### Footprint til PCB:



Disse footprints defineres, når vi laver diagram-tegningerne over vores kredsløb. Herefter vises kun de footprints, som også ses på det ovenstående billede.

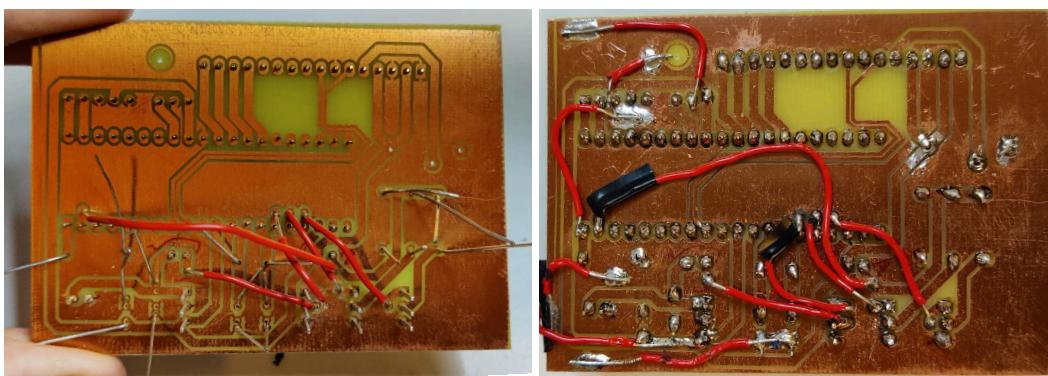
### Det fysiske PCB-print



I første omgang havde vi i kredsløbstegningen vendt spændingsregulatorene, altså TIP120 og LM7805, om.

For at optimere PCB'en endnu sørgede vi for at blandt andet mindske den krævede plads og mindske antallet af nødvendige vias.

For at implementere dette PCB-print i selve modstandssorteringsmaskinen, krævede det at optimere dens pladsbrug, så det passede i 3D-printet. Desuden skulle ledningerne passe til LCD-skærmen, som skulle sidde længere væk i toppen af 3D-printet. Denne pladsbrug bestemmes af komponenternes koblinger og de forbindelser som der køres gennem PCB'en. Derfor bliver der ofte nødt til at lave nogle kompromiser når det gælder ensidigt PCB-print.



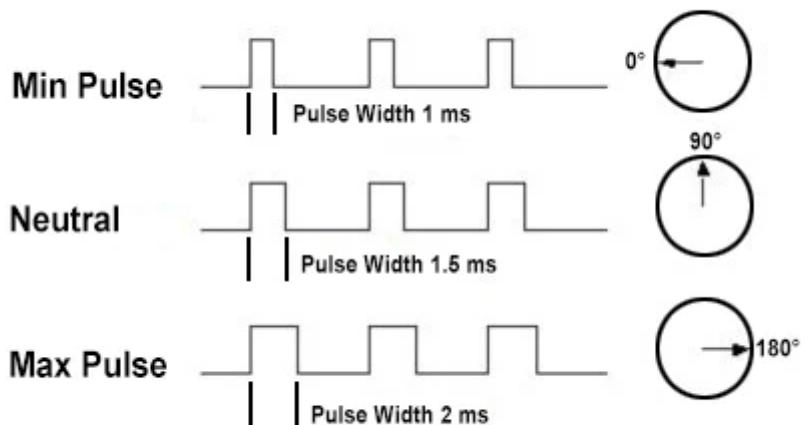
Lodningen af vias på PCB-boardet. Herefter optimerede vi PCB-printet, så det ikke krævede samme antal vias.

Gennem brugen af vias kan der skabes forbindelser mellem komponenter uden at have dem inkluderet på vores PCB-print. En via fungerer ved at lave et hul, som går fra en leder til det tilsvarende forbindelsespunkt. Derefter bruges der en ekstern ledning som loddes fast til disse huller og skaber en forbindelse.

## Anvendelsen af servomotor

At konfigurere servomotoren til at fungere med mikrochippen var vanskeligere, end vi først forventede. Andre mikrokort som Arduino har biblioteker til servo, så de er meget nemmere at bruge end PIC16F877a mikrochippen. For at konfigurere servo'en er det først vigtigt at forstå, hvordan en servo fungerer. En servo er en motor med gear indeni, som kan lede motoren til en præcis placering. Dette gør dem til en bedre mulighed at bruge i vores tilfælde, fordi vi kræver nøjagtig rotation af sorterings-delen. Derfor har en servo 3 ledninger, en til 5V og jord og en tredje til at kommunikere med en kontrol-chip. Vi eksperimenterede med en chip specielt lavet til servover, såsom timeren, LM555, men til sidst besluttede vi os for at bruge PIC16F877A, da vi har mere erfaring med det. Derfor blev det meste af konfigurationen udført i koden.

Styrekanalen på servo'en er lidt specielt. Den forventer en puls af 5V, hvorfra længden bestemmer, til hvilken position motoren skal dreje. Den forventede spænding hvert 20. millisekund, selvom nogle servoer kan have et område fra 10ms til 50ms, så det betyder normalt ikke så meget. Det afgørende er spændings-længden, som skal være mellem 1 og 2 ms. Vi fandt ud af under en test, at denne værdi også kan variere, og de fleste servoer kan stadig dreje med spænding på omkring 0,6 ms.



5

Derfor bør vi, når vi programmerer det, sigte efter at have en puls, hvor vi efter n antal millisekunder ikke må have nogen spænding, der løber igennem i 20 - n millisekunder.

<sup>5</sup> Kilde til billede: <https://www.jameco.com/Jameco/workshop/Howitworks/how-servo-motors-work.html>

## Løsning 1 - PWM

Ved den oprindelige anvendelse af servoen var vi ikke bekendt med dens specifikke krav, og vi antog derfor, at en simpel PWM-løsning (pulsbreddemodulation) ville være tilstrækkelig. Imidlertid opererer vores krysaloscillator ved en frekvens på 16 MHz, og den resulterende puls var simpelthen for hurtig til at kunne anvendes af servoen.

## Løsning 2 - Timer

En anden løsning, som vi prøvede, var at oprette en timer i selve mikrochippet, der kunne håndtere servoen. Timer-systemet refererer til en funktion i programmets kode, der normalt bruges til at tælle tid og udføre handlinger på bestemte tidspunkter. I dette tilfælde blev timer-systemet oprindeligt planlagt til at blive brugt i programmet til at styre servo-motoren. Det burde fungere i teorien, men vi fik inkonsistente resultater med oscilloskopet, så til trods for vores bedste anstrengelser, besluttede vi os for noget andet.

På grund af en indbygget forsinkelse i LCD-skærmen, var det ikke muligt for servomotoren og LCD-skærmen at følge programmets delay på samme måde. Dette skyldtes, at LCD-skærmen havde en indbygget forsinkelse, som gjorde, at den ikke kunne reagere så hurtigt, som programmets delay dikterede. På grund af dette var det ikke muligt at synkronisere servo-motoren og LCD-skærmen ved hjælp af timer-systemet. Trods dette udførte vi en omfattende række beregninger på timeren i et forsøg på at anvende denne metode. Nedenunder er de nødvendige værdier, vi kom frem til for opsætningen. For at beregne RegValue til delay til et Timer0-system med 8 bit, anvendes følgende formel:

$$RegValue = 256 - \frac{Delay \cdot Fosc}{Prescalar \cdot 4}$$

Værdier i dette tilfælde:

- Delay = 1 µs (0,0001 s)
- Frekvensen af Oscillator (Fosc): 10.000.000 ( $10 \cdot 10^6$ )
- Prescaler: 1:16 (16)

Med indsatte værdier:

$$256 - \left( \frac{(0.0001 \cdot 10000000)}{(16 \cdot 4)} \right)$$

$$240.3750000$$

### Løsning 3 - Delay

Til trods for ulemper i forsøget ved at bruge de hidtidige metoder, besluttede vi os for at bekæmpe ild med ild. Den løsning, som viste sig at fungere bedst, er samtidig den mest letforståelige og letteste at implementere i resten af systemet. Ved at anvende forsinkelser til at forsinke leveringen af spænding til servomotoren, kan vi opnå den ønskede forsinkelse uanset det nødvendige tidsinterval, hvorefter der ikke efterlades nogen spænding i de resterende 20 ms. Dette kan pakkes sammen i en while-løkke, så pulsen ville køre i et par sekunder, hvilket sikrer, at servo'en har tid nok til at komme i position, og herefter fortsætter koden.

Der opstår tilmed to ulemper ved denne tilgang, der heldigvis ikke påvirker os. Den første er, at mens servo'en bevæger sig, kan vi ikke kontrollere resten af vores system på grund af forsinkelserne. Dette er ikke et problem for os, fordi koden er meget segmenteret. Dog skal vi sikre os, at servo'en er i den rigtige position, før vi fortsætter. En anden ulempe er, at når servo'en har bevæget sig og pulserne slutter, er det muligt at flytte det manuelt, men det er igen ikke et problem, så længe brugeren ikke roder med mekanismen.

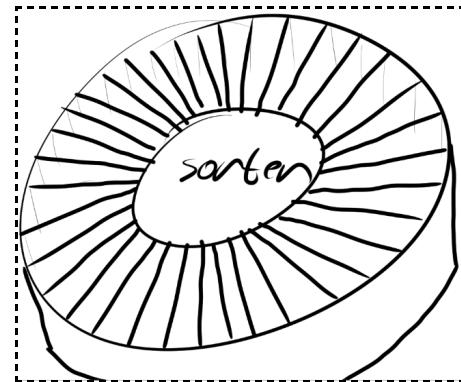
### Implementering af servo'en i systemet

For at implementere servo motoren i resten af modstandssorteringsmaskinen, og dermed sammensættes med hele systemet, startede vi først med at udarbejde nogle sketches, tegninger og mock-ups af idéer til dette. Derefter udvalgte vi den bedste og mest realistiske sketch, og forsøgte at 3D-modellere dette i henholdsvis Blender eller Sketchup, så de kunne 3D-printes.

Herefter kunne 3D-modellerne blive sliced i Creality Slicer, så de kunne klargøres til skolens Creality Slicere i den ønskede størrelse og med det ønskede filament. Hertil var det vigtigt, at vi anvendte omkring 10% infill i 3D-printningen, da det gav den mest stabile struktur, samtidigt med at det tog kortest muligt tid at printe. Det var nemlig vigtigt at kunne nå at disponere tiden ordentligt, så 3D-printsne kunne nå at færdiggøres til inden vi skulle anvende dem i projektet.

### Mock Up Sketch 1

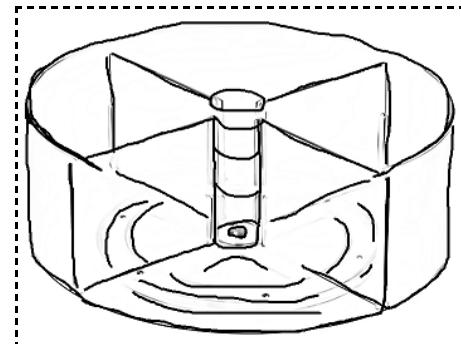
Den første sketch illustrerer et bud på at opdele sorteringsmaskinen i forskellige beholdere til mange forskellige modstande, så den kunne sortere mest muligt. Dog ville det for det første være svært at nå at sørge for, at servo motoren kunne bevæge sig præcist til en bestemt beholder.



For det andet ville man ikke have mulighed for at sortere særligt mange af hver type modstand, hvis beholderne var så små. Sluteligt besluttede vi i gruppen at tage udgangspunkt i en anden tegning og idé, som involverede færre beholdere og stemte overens med problemafgrænsningens antal af hyppigst anvendte modstande.

### Mock Up Sketch 2:

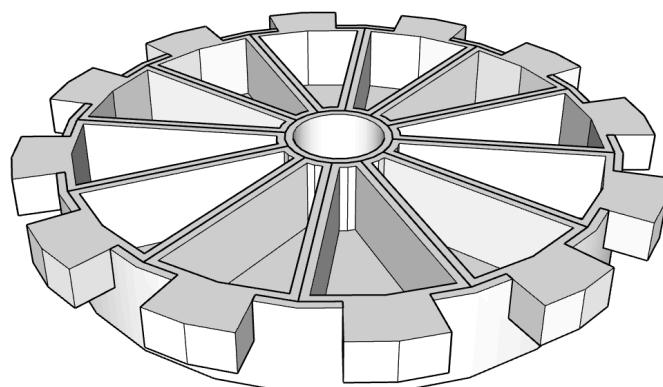
Herefter tænkte vi på at have 8 kasser i alt, da dette var et fint lavt antal af beholdere, der ville være overskueligt og nemt at lave. Dog besluttede vi under designprocessen, at vi kunne optimere antallet af beholdere, så der var plads til 4 mere, der i alt gav os 12 kasser. Dette var det mest optimale antal, der også stemmer overens med problemafgrænsningens angivelse af de hyppigst anvendte modstande. Desuden kunne der med dette antal beholdere være plads til restskuffer med henholdsvis under 1k, over 1k og over 10k modstande.



### 3D-modellering i SketchUp

Her ses den endelige 3D-model af sorteringsmetoden med de respektive tolv beholdere.

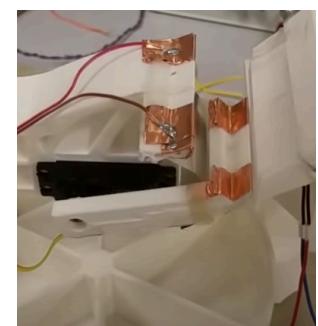
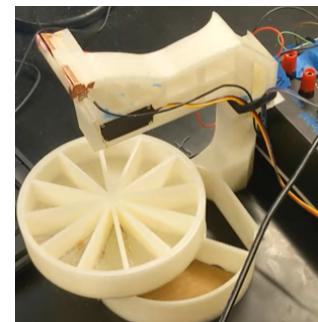
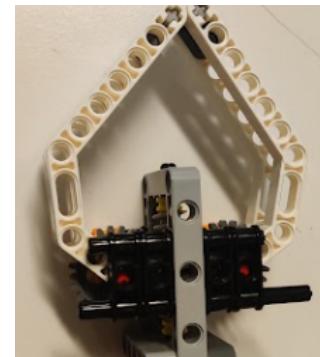
Hvor ti af beholderne er dedikeret til en bestemt modstand, og de resterende to beholderne er tilskrevet som restbeholdere.



## Indsættelse af modstande i systemet

### Idé: Robotarm

Idéen om anvendelsen af en lille robotarm til at indsætte modstande i et modstandssorterings-system går ud på at automatisere processen med at sortere og teste modstande ved hjælp af en robotarm, der kan indsætte og fjerne modstande i et testkredsløb. Robotarmen ville være i stand til at tage en modstand fra en bunke og indsætte den i det korrekte sted i testkredsløbet. Når modstanden er testet, ville robotarmen derefter fjerne den fra testkredsløbet og placere den i den korrekte sorteringsbunke baseret på dens værdi og tolerancer. Ved at automatisere denne proces med en robotarm ville det være muligt at reducere den tid og det arbejde, der er involveret i manuelt at sortere og teste modstande, samtidig med at nøjagtigheden og pålideligheden forbedres. Det ville også gøre det muligt at håndtere store mængder af modstande på en mere effektiv måde, hvilket kan være afgørende for produktionen af elektroniske produkter. Denne idés oprindelse var inspiration fra andre, der også forsøgte at lave en autonom modstandssorteringsmaskine.<sup>6</sup>



Rent teknisk ville denne løsning indebære, at anvende en solenoide, der er en oprullet elektrisk ledning (elektrisk spole) og en massiv magnetisk fjederbelastet (jern)cylinder. Her ville man kunne placere en transistor tilsluttet power-resistors til at styre solenoiden i feeding-mekanismen til indsættelse af modstande i systemet.



Solenoide, 5V<sup>7</sup>

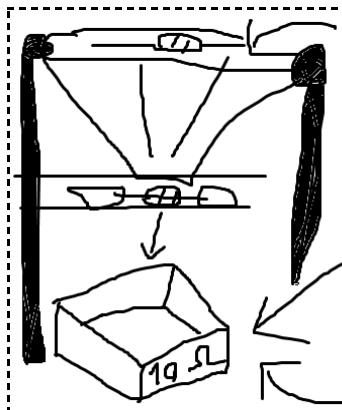
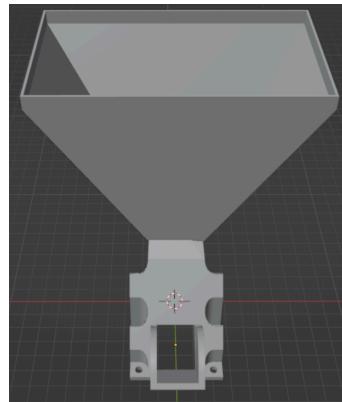
<sup>6</sup> <https://youtube.com/shorts/-8SATs1ip4?feature=share>

<sup>7</sup> <https://let-elektronik.dk/cache/3/2/5/0/3/9/6/fit-640x640x100.webp>

### Idé: Tragt

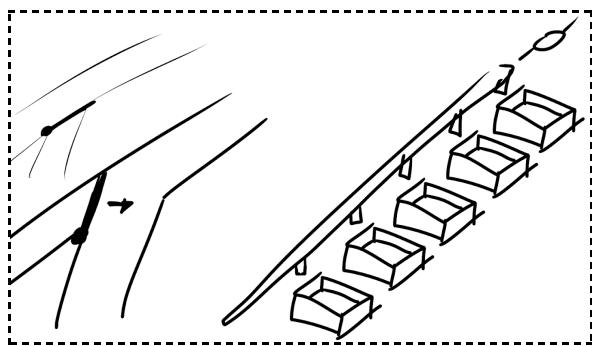
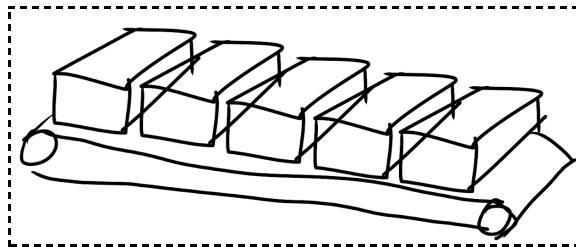
En tragt kunne bruges til at sortere modstande i et modstandssorteringssystem ved at bevæge modstandene gennem forskellige måleinstrumenter, der mäter egenskaber som modstandsværdi og tolerance ved enden af tragten.

Tragten kan placeres øverst i systemet, og den er forbundet til en serie af forskellige måleinstrumenter, der er i stand til at mäter modstandens værdi. Når en modstand indsættes i tragten, bevæger den sig ned gennem systemet, og til sidst ender i en af de respektive beholdere. Desuden er tragten en god måde at opbevare flere modstande på en gang, så man minimere mængden af manuelt arbejde. Det eneste brugerne skulle gøre ville være at rette en modstand ud og placere den i toppen af tragten, så ville den autonome modstandssorteringsmaskine gøre resten af arbejdet.



### Idé 4: Rullebånd med kasser

Rullebåndet fungerer som en transportmekanisme, der fører modstandene fra et startpunkt til et endepunkt, hvor de er blevet sorteret efter deres resistansværdi. Modstandene placeres i små kasser, der bevæger sig langs rullebåndet og som er designet til at holde forskellige modstandsværdier. Systemet fungerer ved, at modstandene sættes ind i kasserne på rullebåndet.



Disse kasser bevæger sig langs rullebåndet, og som de bevæger sig, bliver modstandene identificeret ved en eller flere spændingsdelere, der er placeret i systemet.

### Endelige idé: Transportbånd



Billeder af udviklingen af et transportsystem, hvor modstanden kan indsættes og måles på vej ind i modstandssorteringsmaskinen.

Inspirationen bag dette er, hvordan disken bliver trukket ind på en spillekonsol. Her anvendes gummihjul og friktion for at flytte modstandene fra den ene ende til den anden. Ved hjælp af et kobberbelagt tape på to af hjulene, hvor modstanden bevæger sig ned, sikrer vi kontakt med begge sider. Dermed er det muligt at måle modstandenes værdi, da kobber-tapen kobles til en ledning til microcontrolleren. For at udarbejde denne idé, eksperimenterede vi med to versioner og placeringer af LEGO-hjulene. Den første version havde til hensigt at mindske tom plads, men modstandene ville bøje sig rundt om hjulet og køre uden for den tilsigtede sti. Den anden version løser dette problem, men det er stadig muligt for modstandene at sidde fast, hvis de sættes for lavt, fordi de kan passere under hjulene. Ydermere kunne man tilføje en lille motor, der kunne rykke på hjulene automatisk, så det eneste bruger skal gøre, er at indsætte en rettet modstand i systemet. Dog endte vi ikke med at prioritere denne anvendelse af motoren, da det var nemmere at konfigurere og teste, hvis man manuelt skulle dreje modstanden ind i systemet, til LCD-skærmen registrerede et input fra modstandsmålingen.

## LCD-skærmens funktioner

Oprindeligt var tanken, at LCD-skærmens skulle vise en række oplysninger omkring de sorterede modstande til brugeren i denne brugergrænseflade ved brug af LCD-skærmens som GUI. Dog endte vi med at blot anvende LCD-skærmens til at give et enkelt output, der fortæller brugeren om den præcise modstandsværdi, som modstandssorteringsmaskinen mäter via kobber-tapen i starten af systemet.

Der oprettes herefter et while-loop, som anvendes til at vise modstandsværdien, der er defineret som funktionen: ADC\_Read\_Int(). Modstandsværdien vises på LCD-skærmens så længe ADC\_Read\_Int er større end 0, altså registrerer et input:

```
while (ADC_Read_Int() > 0) {
    __delay_ms(100);
}
```

Hvis LCD-skærmens ikke mäter eller registrer en modstandsmåling, så vil skærmens blot vise et lille stillestående ikon (---0000---), der minder om en modstand:

```
} else {
    Lcd_Clear();
    Lcd_Set_Cursor(1,1);
    Lcd_Print_String("----0000----");
    __delay_ms(100);
```

## Beskrivelse af programmeringen

Denne kode er et program skrevet i programmeringssproget C. Det er beregnet til at styre en servomotor og måle modstanden i en kreds ved hjælp af en analog-til-digital konverter (ADC). Koden er opdelt i flere sektioner, hver med et specifikt formål.

Den første sektion af programmeringen indeholder flere #define-erklæringer, som bruges til at definere konstanter, der bruges i hele koden. For eksempel er konstanten \_XTAL\_FREQ sat til 16000000, som repræsenterer frekvensen af krystaloscillatoren, der bruges i kredsløbet. Derudover tildeles flere pins på mikrocontrolleren til specifikke variable, såsom RS, EN, D4, D5, D6 og D7.

Den anden sektion indeholder flere header-filer, der er nødvendige for, at koden kan fungere korrekt. Disse filer inkluderer "xc.h", som indeholder definitioner for specifikke mikrocontrollerinstruktioner, og "lcd\_lib\_header.h", som er en bibliotek til interaktion med en LCD-skærm. Derudover inkluderer koden de standard C-biblioteker "stdlib.h", "string.h" og "stdio.h", samt biblioteket "math.h" til matematiske funktioner.

Den tredje sektion indeholder flere #pragma-erklæringer, som er kompiler-specifikke direktiver, der bruges til at indstille konfigurationsindstillinger for mikrocontrolleren. Disse indstillinger inkluderer oscillatorvalg, watchdog timer aktivering, power-up timer aktivering, brown-out reset aktivering, lavspændings in-circuit serial programmering aktivering, data EEPROM hukommelsesbeskyttelse og flash-programhukommelsesbeskyttelse.

Den næste sektion omhandler mere specifikt selve modstandssorteringen, og ikke blot opsætningen af programmet. Her defineres to funktioner, DelayUp og DelayDown, der bruges til at styre servomotoren:

```
void DelayUp(uint8_t n) {
    for (; n > 0; n--){
        __delay_us(10);
    }
}

void DelayDown(uint8_t n) {
    n = 2000-n;
    for (; n > 0; n--){
        __delay_us(10);
    }
}
```

Disse funktioner tager et argument n, som repræsenterer forsinkelsestiden i mikrosekunder. DelayUp-funktionen tilføjer en forsinkelse på 10 mikrosekunder i n iterationer, mens DelayDown-funktionen beregner antallet af iterationer, der er nødvendige for at oprette en forsinkelse på 20 millisekunder (20000 mikrosekunder) minus n mikrosekunder.

ADC\_Init-funktionen initialiserer den analog-til-digitale konverter ved at sætte ADCON0- og ADCON1-registrene til specifikke værdier. ADC\_Read\_Int-funktionen læser udgangen af ADC'en, når den kaldes som en integer, mens ADC\_Read\_Float-funktionen læser udgangen af ADC'en, når den kaldes som en float. Altså er der tale om en forskellig registrering af modstandens input ud fra om den er med flere betydende cifre i form af decimaltal (floating point):

```
float ADC_Read_Float()
```

eller om den blot modtager en værdi som heltal (integer):

```
int ADC_Read_Int()
```

Hovedfunktionen (main) er, hvor størstedelen af koden udføres. Den første sektion af hovedfunktionen erklærer flere variable, herunder unsigned integers a og i samt integers resistance og servo. Char-arrayet str er også erklæret til senere brug til at vise cifrene af den målte modstands værdi på LCD-skærmen. TRIS-registrene bliver herefter sat til at konfigurere mikrocontrollerens pins til input eller output. LCD-displayets pins (TRISD og TRISE) bliver sat som outputs, mens servo-motorens og spændingsdelerens pins (TRISB og TRISC) bliver sat som outputs. Lcd\_Start funktionen bliver kaldt for at initialisere LCD-displayet, og ADC\_Init funktionen bliver kaldt for at initialisere den analog-til-digital konvertering.

En for-løkke bliver derefter udført 1000 gange for at sætte servo-motoren til dens standardposition. Inde i for-løkkens bliver PORTB-registret sat til 1 for at forsyne servo-motoren med strøm, og DelayUp-funktionen bliver kaldt med et argument på 150 for at skabe et forsinkelse på 1,5 millisekunder. PORTB-registret bliver derefter sat til 0 for at slukke for strømmen til servo-motoren, og DelayDown-funktionen bliver kaldt med et argument på 150 for at skabe et forsinkelse på 20 millisekunder minus 1,5 millisekunder:

```
for(i=0;i<1000;i++) { // Sets the servo to 0
    PORTB = 1;          // Has power
    DelayUp(150);      // Delay of 1.5ms
    PORTB = 0;          // No power
    DelayDown(150);    // Delay of 20 - 1.5 ms
}
```

En while-løkke bliver derefter udført uendeligt for at måle modstanden i kredsløbet og styre servo-motoren baseret på den målte modstand. Inde i while-løkken bliver PORTC-registret sat til 0b00000001 for at tænde for motoren.

```
while(1)
{
    PORTC = 0b00000001; // Turns on the motor
    if (ADC_Read_Int() > 0) { // Check if there's connection
        __delay_ms(1000);
    }
}
```

Derefter kaldes ADC\_Read\_Int-funktionen for at læse modstandsværdien fra det analoge input-pin A0. Funktionen returnerer en integer-værdi, som derefter bruges til at styre positionen af servo motoren. Værdien, der bliver returneret af ADC\_Read\_Int-funktionen, skaleres til en værdi mellem 0 og 180 grader, som svarer til bevægelsesområdet for servomotoren. Denne værdi skrives derefter til servo motoren ved hjælp af Servo\_Write-funktionen. Funktionen tager vinklen og et servo-objekt som parametre og sætter motoren til den ønskede position. Efter at servo motoren er sat til den ønskede position, indføres en forsinkelse på 200 millisekunder ved hjælp af Delay\_ms-funktionen for at give motoren tid til at falde til ro i positionen. PORTC-registeret sættes derefter til 0b00000000 for at slukke for motoren.

While-loopet gentages derefter, og processen med at måle modstanden og styre motoren fortsætter ubegrænset. Dette loop sikrer, at motoren kontinuerligt tilpasser sig modstandsændringer i kredsløbet og opretholder den ønskede position.

```
while (ADC_Read_Int() > 0) {
    __delay_ms(100);
}
```

Alt i alt bruger koden en analog inputpin, en servo motor og en mikrocontroller til at styre positionen af motoren baseret på modstanden i kredsløbet. Mikrocontrolleren mäter kontinuerligt modstanden ved hjælp af det analoge inputpin og tilpasser positionen af servomotoren tilsvarende for at opretholde en konstant modstandsværdi.

## Beskrivelse af de anvendte registre i programmeringen

Dette program er skrevet i sproget C og er designet til at styre en microcontroller og dens tilsluttede hardware. Programmet indeholder flere registre, som spiller en central rolle i styringen af microcontrolleren. De forskellige registre er alle vigtige for at sikre, at microcontrolleren fungerer korrekt og kan udføre de ønskede opgaver.

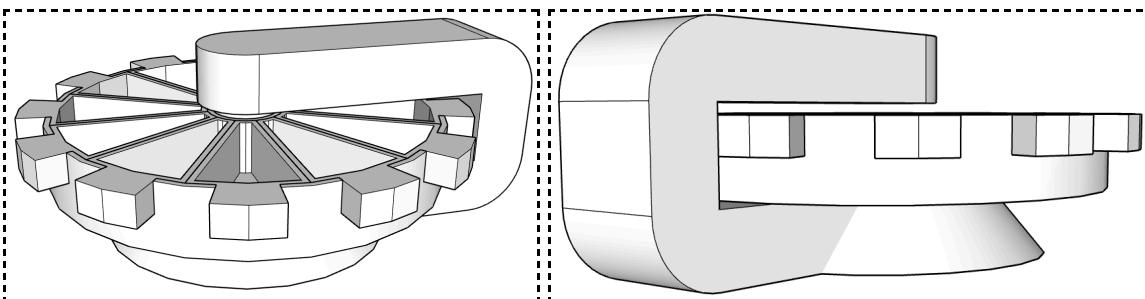
De første registre, TRISB, TRISC, TRISD, og TRISE, bruges til at styre retningen af hver af portene på microcontrolleren. Disse registre indstilles til at angive, om portene skal fungere som input eller output. I dette tilfælde er alle porte indstillet til at fungere som output. ADCON0 og ADCON1 er registre, der styrer Analog-Digital Converteren (ADC). ADC'en er i stand til at måle analoge signaler og omdanne dem til digitale værdier. Disse registre er indstillet til at styre ADC'en og sikre, at de korrekte værdier opnås.

Resultaterne gemmes i ADRESH og ADRESL registrene. GO\_DONE er en bit, der sættes i ADCON0 for at starte en ADC-konvertering. Denne bit er vigtig for at starte konverteringsprocessen og sikre, at ADC'en fungerer korrekt. Derudover bruges registrene PORTB, PORTC, PORTD og PORTE til at læse og skrive til de tilsvarende porte på microcontrolleren. Disse registre bruges til at styre motorer, servoyer og LCD-skærme. RS, EN, D4, D5, D6 og D7 er makrodefinitioner, der bruges til at styre LCD-skærmen. Disse makrodefinitioner bruges til at sende kommandoer eller karakterer til LCD-skærmen og indeholder de høje bits i LCD-dataen.

Programmet indeholder også nogle konfigurationsregistre, der indstiller microcontrollerens adfærd. Disse registre er konfigureret ved hjælp af #pragma config og sikrer, at microcontrolleren fungerer korrekt. Konfigurationsregistrene omfatter FOSC, WDTE, PWRTE, BOREN, LVP, CPD, WRT og CP. Disse registre sikrer, at microcontrolleren bruger en højfrekvenskrytal på 16 MHz som oscillator, og at Watchdog Timer, Power-up Timer og Brown-out Reset er korrekt konfigureret. Derudover er Low-Voltage Programming slået fra, og Data EEPROM Memory Code Protection og Flash Program Memory Code Protection er også slået fra for at sikre, at EEPROM og programhukommelsen kan skrives til og læses fra.

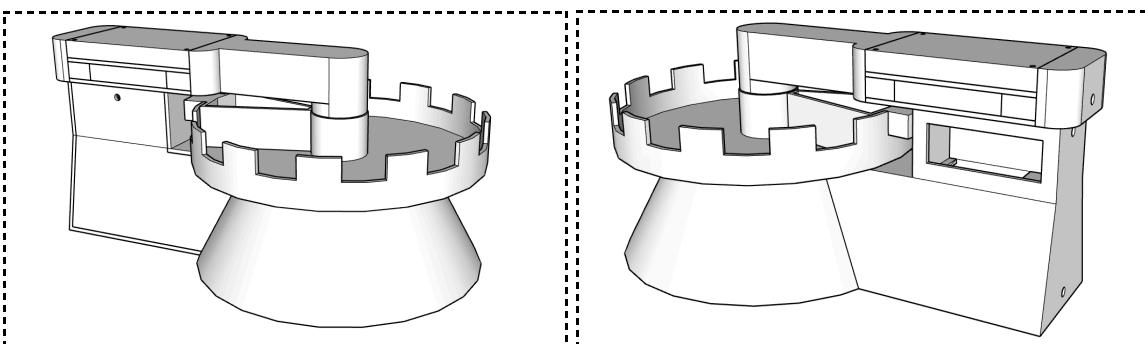
## Beskrivelse af modstandssorteringsmaskinen

### Prototypen til en 3D-model af modstandssorteringsmaskinen



På denne prototype og overordnet tegning af modstandssorteringsmaskinen, var der hverken implementeret plads til en servomotor eller en LCD-skærm i selve maskinen.

### Den endelige 3D-model af modstandssorteringsmaskinen



Den endelige 3D-model ser sådan ud, og inkluderer alle modstandssorteringsmaskinens funktioner.

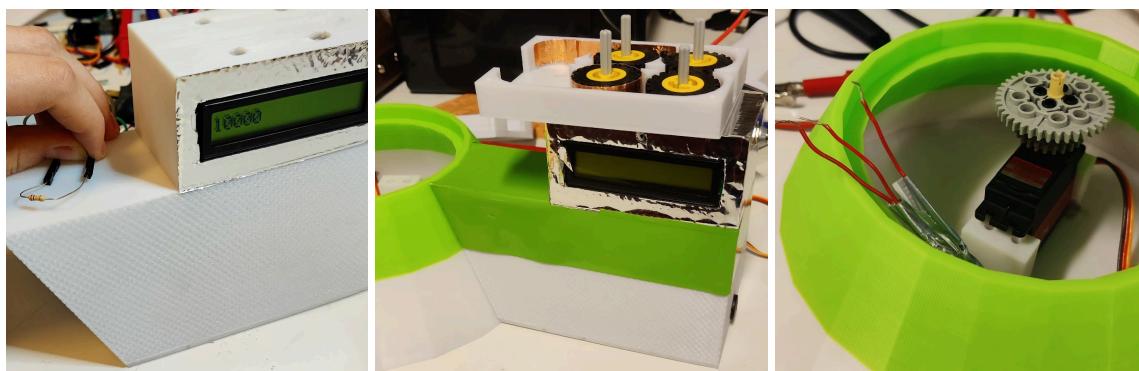
Desuden har denne model gjort plads til både LCD-skærmen, servomotoren og også en eventuel implementering af en motor til indsættelsen af modstande i systemet.

Maskinen starter, når brugeren tænder den. Når brugeren ønsker at måle størrelsen på en modstand, indsætter de modstanden i maskinen. Herefter fastholder maskinen modstanden mellem to kobberplader for at måle dens størrelse. Efter at have målt modstanden, sammenligner maskinen dens størrelse med de foruddefinerede størrelseskategorier og identificerer den passende kategori.

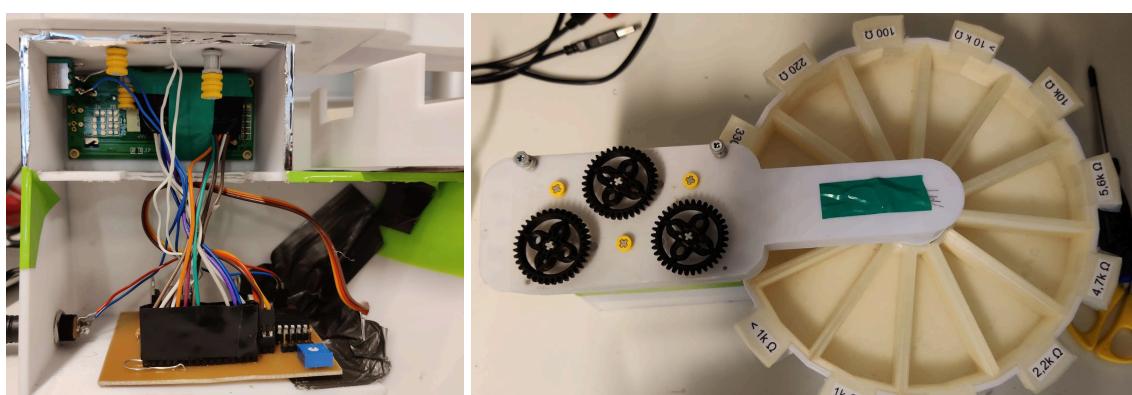
For at sortere modstandene bruger maskinen en servomotor til at dreje rummene, indtil det rigtige rum er nået, hvorefter maskinen placerer modstanden i det rigtige rum. Processen gentages for alle modstande, der er blevet indsat i maskinen. Når alle modstande er sorteret, stopper maskinen automatisk. Samlet set kan denne proces effektivisere arbejdet med at sortere modstande og sikre, at de er placeret korrekt.

For at dette kan lade sig gøre, begynder programmet rent teknisk med at initialisere de forskellige porte og ADC'en. Herefter kører programmet i en uendelig løkke, hvor det først tænder motoren og tjekker for en forbindelse ved at læse værdier fra ADC'en. Hvis der er forbindelse, venter programmet et sekund for at sikre, at forbindelsen er stabil. Derefter undersøger programmet, hvilken modstand der er tilsluttet, ved at læse værdier fra ADC'en. Hvis værdien fra ADC'en svarer til en bestemt modstandsværdi, sætter programmet en korresponderende værdi for servoens position. Denne værdi bestemmer, hvilken mekanisk position systemet til sorterings af modstandene skal tage. Til sidst gentages løkken igen og igen indtil alle de ønskede modstande er blevet sorteret, eller brugeren har manuelt slukket for modstandssorteringsmaskinen igen.

### Opsætningen af modstandssorteringsmaskinen



Billeder af den færdige modstandssorteringsmaskine



### Videodokumentation af en demonstration af modstandssorteringen:

- <https://youtu.be/H8uLL0rGSkU>
- <https://youtu.be/09GmGWsfW1E>
- <https://youtu.be/KqqIMkx1wqM>

# Opfyldelse af krav

## Bløde krav:

*Brugervenlighed: Maskinen skal være nem at bruge og betjene, selv for personer uden specialiseret teknisk viden:* Brugervenlighed opnås ved at have et enkelt design. Det eneste, brugeren skulle have gjort, er at tænde for det og derefter sætte modstande gennem hullet på bagsiden, og det vil automatisk registrere det, måle det og justere rotationen. Desværre i denne nuværende tilstand skal brugeren manuelt dreje gearene øverst, men hvis maskinen havde en motor som vi havde planlagt, ville det ikke have været et problem.

*Fleksibilitet: Maskinen bør være i stand til at sortere en bred vifte af modstande og tolerancer: Nøjagtighed* opnås ved at bruge mere end en modstand i maskinen til specifikke modstands-områder. Derudover har vi hardcoded specifikke værdier og rotationspositioner for specifikke modstande, men ikke for alt, derfor hvis maskinen detekterer en modstand, der ikke er kodet i koden, vil den lave beregninger og vise modstanden på displayet.

*Pålidelighed: Maskinen bør have en høj grad af pålidelighed og nøjagtighed for at sikre korrekt sortering af modstandene:* Nogle aflæsninger fra mikrochippen er lidt underlige, så vi kan aldrig få en præcis værdi for modstanden. Men vi har en fail safe, hvor en anden modstand bruges til at måle forskellige modstande, så estimater bør være tæt nok på den faktiske værdi, til at brugeren selv kan finde ud af, hvad modstanden er.

*Sikkerhed: Maskinen bør være designet med passende sikkerhedsfunktioner for at undgå skader på brugere og maskinen selv:* Hullet, hvor man sætter modstande i, er for lille til, at en finger kan komme igennem, så der er ingen fare for at få fingeren fast derinde. Ved måling af modstande drejer sorteringsenheden dog relativt hurtigt. Hvis man ikke er forsiktig omkring det, kan det ramme dem, og det vil gøre ondt. At sætte et beskyttende kabinet rundt om sorteringen, ville have gjort maskinen for omfangsrig, og det var ikke vores topprioritet, da vores fokus var på elektronikken og de hårde krav.

**Hårde krav:**

*Præcision: Maskinen bør være i stand til at sortere modstande præcist med en høj grad af*

*nøjagtighed:* Ved nøjsom udvælgelse af en præcis metode til at måle og identificere

modstandene, er det lykkedes at opfylde dette hårde krav. Præciseringen af

modstandsmålingen indebar nemlig anvendelsen af et 10-bit system med PIC16F877a og

flere parallelforbindelser, der agerede som forskellige spændingsdelere. Desuden kan

servomotoren med 16 MHz krystallen til oscillatoren nøjagtigt bevæge beholderne til

den rigtige position.

*Hastighed: Maskinen bør kunne sortere modstandene hurtigt for at øge effektiviteten af*

*produktionsprocessen:* Servomotoren bevæger sig hurtigt nok til at kunne nå at flytte

beholderne til den ønskede modstandsværdi inden brugeren kan nå at indsætte

modstanden i systemet, så dette krav er også blevet opfyldt. Dog kunne

videreudviklingen af projektet med fordel indeholde en motor til indsættelsen af

modstande, for at gøre denne proces mere autonom, mere effektiv og hurtigere.

*Kapacitet: Maskinen bør have tilstrækkelig kapacitet til at håndtere et stort antal*

*modstande på én gang:* Beholderne har plads til 125 modstande hver, som er mere end

nok til at kunne håndtere et stort antal af modstande. Ved videreudviklingen af projektet

kunne man desuden implementere en tragt, der kunne gøre det muligt at sortere flere

modstande på én gang i selve sorteringen.

**Tekniske krav:**

*Eventuelt Sensorteknologi: Maskinen bør være udstyret med højteknologisk*

*sensorteknologi, der er i stand til at registrere og identificere modstandsværdier præcist:* I

løbet af udviklingen af projektet viste det sig at være langt mere fordelagtigt at anvende

spændingsdelerkredsløb til at identificere modstandenes værdi, så vi endte slet ikke med

at anvende sensorteknologi såsom farvesensorer. Desuden er målingen meget mere

præcis, når det foregår på den måde i stedet for at stole på, at alle modstandenes

farkekoder er lige letlæselige eller helt samme farver.

*Software og programmering: Maskinen bør anvende et programmerbart PICkit, der kan videregive oplysninger om de givne modstande, og sortere modstandene korrekt:*

Projektet anvender løbende et PICkit, der kan anvendes til at foretage en analog-digital-konvertering mellem den digitale programmering og microcontrolleren, PIC16F877a. Dermed er dette krav også opfyldt i projektet, da et PICkit bliver brugt hyppigt i løbet af processen til at implementere al programmeringen i produktet.

*Robusthed: Maskinen bør være robust og holdbar for at modstå gentagne brug og vedligeholdelse:* Modstandssorteringsmaskinen er fremstillet med hovedsageligt 3D-print, som er meget robust plastik med 10% infill, der dermed er holdbar nok til at kunne sortere en lang række modstande. Desuden er PCB-printet, elektronik og alle ledninger gemt væk i 3D-printet, som både giver maskinen et flot æstetisk udtryk, samt sørger for at brugere ikke kommer til at rode rundt i elektronikken.

#### **Andre Specifikke krav:**

*Maskinen skal kunne tjekke om nogen af beholderne er fyldte, og derefter kunne advare brugeren om dette:* I starten af projektet blev en optælling af det maksimale antal af modstande foretaget, der konstaterede, at der maksimalt kunne være 125 modstande i en enkelt beholder. Derfor valgte vi at angive det maksimale antal til 100, for at være på den sikre side. Når modstandssorteringsmaskinen har talt op til 100 modstande af en bestemt type i en bestemt beholder, var meningen, at den skulle advare omkring dette på LCD-skærmen, og kræve et brugerinput før den kan fortsætte igen.

*Maskinen skal regelmæssigt blive nulstillet, så offset i servo-mekanismen ikke bliver for stor:* Hver gang brugeren trykker på tænd-knappen til modstandssorteringsmaskinen, vil servomotoren gå til en neutral eller nulstillings-position, der vil sikre at servomotoren ikke løbende får et for stort og usikkert offset.

*Maskinen skal kunne tælle og beregne prisen for de modstande, som man har sparet skolen for at bruge penge på:*

En modstand koster i gennemsnit 0,25 kr (25 øre), så denne værdi kan implementeres i programmeringen, samt blive fremvist på LCD-skærmen. Dog valgte vi at fremprioritere, at LCD-skærmen blot skulle vise den identificerede modstandsværdi.

## Konklusion

Maskinen starter, når brugeren tænder den, og derefter indsætter brugeren modstanden i maskinen. Maskinen mäter størrelsen på modstanden ved at fastholde den mellem to kobberplader og sammenligne dens størrelse med de foruddefinerede størrelseskategorier. Når størrelsen er blevet identificeret, bruger maskinen en servomotor til at dreje rummene og placere modstanden i det rigtige rum.

Teknisk set initialiserer programmet de forskellige porte og ADC'en og kører derefter i en uendelig løkke. Programmet tænder motoren og undersøger for en stabil forbindelse. Hvis der er forbindelse, undersøger programmet modstandsværdien ved at læse værdier fra ADC'en. Programmet fastlægger derefter servoens position baseret på modstandsværdien og gentager løkken indtil alle modstande er sorteret. Samlet set kan den autonome modstandssorteringsmaskine spare enorme mængder af tid brugt på manuel sortering og sikre, at modstandene er sorteret korrekt.

Modstandens værdi kan vises på en LCD-skærm, så brugeren nemt kan identificere modstandens størrelse, hvis den ikke stemmer direkte overens med en af de respektive beholdere. Selvom modstanden ville blive sorteret i en af rest-beholderne, ville det være rart for brugeren at kunne have mulighed for at identificere alle modstande, som de har behov for at kende. Desuden illustrerer LCD-skærmen også den målte modstandsværdi, da det er brugervenligt og informativt at vide, hvilken beholder modstanden kommer til at placeres i og om det stemmer overens med modstandens målte værdi.

Denne proces fungerer hovedsageligt autonomt, da selve identificeringen af modstande og servomotorens sortering fungerer automatisk og efter hensigten.

Derudover er identificeringen af modstandsværdien autonom, da den måles ved brug af spændingsdelerformlen og specifikke parallelforbindelser, der gør målingen endnu mere præcis.

# Evaluering

## Projektstyring med SCRUM

- <https://youtu.be/HILs1kQWbd4>

SCRUM var en nyttig projektstyringsmetode at anvende til dette eksamsprojekt. Det gav gruppen øget fleksibilitet, bedre kommunikation og risikostyring, hvilket hjalp med at identificere og afhjælpe risici tidligt i processen og øge chancerne for at opnå kravene. Først var opgaverne en opsætning af breadboardet, 3D-modellering, tegningen af mock-ups og yderligere idégenerering. Derefter arbejdede vi på at teste og anvende servomotor i projektet. På daværende tidspunkt begyndte vi allerede rapportskrivningen og kredsløbstegningerne til et fremtidigt PCB-print. I løbet af teknikfagsugen blev alle 3D-printsne færdiggjort, servomotoren blev sammensat med resten af det analoge system med PIC16F877a, PCB-printet blev fremstillet og hele projektet blev finpudset og optimeret.

## Gruppearbejde og indsats

Først og fremmest var det overordnede gruppearbejde og gruppens indsats meget tilfredsstillende for alle gruppemedlemmer. Dette kunne lade sigøre, da vi overholdte vores indbyrdes aftaler i gruppekontrakten og fordelinger af arbejdet, samt sørgede for at give plads til at hygge og snakke sammen med hinanden. For at optimere projektet havde vi valgt at påtage os forskellige ansvarsområder. Noah var ansvarlig for at lave diagramtegninger i KiCad og flowcharts, samt fremstilling af PCB og at bore huller i PCB'en. Kristiyan tog sig af programmering, mens han og Clara arbejdede sammen om 3D-modellering i Sketch-up eller Blender. Derudover var Clara ansvarlig for tidsplanen og anvendte SCRUM-metoden til at organisere vores arbejde. Hun tog også sig af dokumentation af projektet og lodning af komponenter på PCB'en.

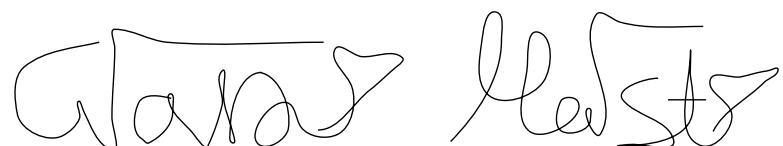
Den interne kommunikation fungerede også godt, da vi havde oprettet en fælles "Discord"-gruppe, hvor vi løbende kunne stille spørgsmål til hinanden, og det gjorde hele processen meget mere effektiv. Selvom vi alle havde vores egne opgaver, hjalp vi hinanden med vores arbejde for at opnå kravene sammen.

**Ny opnået viden fra projektet**

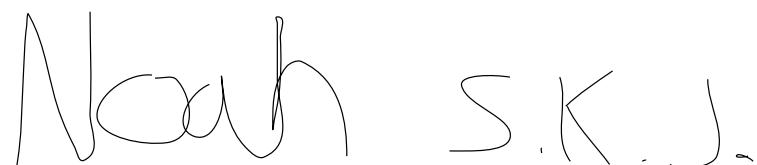
Gennem projektet lærte vi at arbejde med komponenter og samle dem til et større produkt, både gennem programmering og elektriske kredsløb. Denne viden vil helt klart kunne gavne os i produktionen af fremtidige produkter, som er opbygget af disse elementer, vi har anvendt under projektet. Vi var også særligt glade for at arbejde med 3D-print, da det er en fælles interesse i vores gruppe at arbejde med både 3D-modellerings-software og se det realiseret med en 3D-printer. Det blev hyppigt anvendt under teknikfagsugen, hvor gruppen også arbejdede hårdt på at optimere den tid, de havde tilbage, og gå i dybden med at udvikle projektet. Det var også en god beslutning at skrive på vores skriftlige rapport undervejs i projektet. Det gav os en fordel i forhold til at nå at beskrive og medtage alle væsentlige elementer af vores projekt, og vores beskrivelser kunne blive mere præcise, da vi nedskrev de teoretiske elementer lige, da vi arbejdede med det. Samlet set var det en rigtig god oplevelse at arbejde sammen på dette projekt, og vi har lært meget, som vi kan tage med os videre.

**Underskrifter, den 30. April 2023**

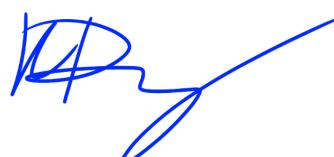
Clara Holst:



Noah Jørgensen:



Kristiyan Georgiev:



# Kildeliste

*KiCad:*

- <https://www.kicad.org/>

*PCB:*

- [https://en.wikipedia.org/wiki/Printed\\_circuit\\_board](https://en.wikipedia.org/wiki/Printed_circuit_board)

*Kilder til billederne i løbet af projektet:*

- <https://www.futurlec.com/LED/LCD16X2.shtml>
- [https://learn.sde.dk/pluginfile.php/1052489/mod\\_resource/content/1/pic16f684ICS\\_P.jpg](https://learn.sde.dk/pluginfile.php/1052489/mod_resource/content/1/pic16f684ICS_P.jpg)
- <https://www.jameco.com/Jameco/workshop/Howitworks/how-servo-motors-work.html>
- <https://let-elektronik.dk/cache/3/2/5/0/3/9/6/fit-640x640x100.webp>

Datasheets:

- <https://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf>
- <https://ww1.microchip.com/downloads/en/DeviceDoc/31002a.pdf>
- <https://cdn-shop.adafruit.com/datasheets/TIP120.pdf>
- <https://www.sparkfun.com/datasheets/Components/LM7805.pdf>
- <https://www.ti.com/lit/ds/symlink/ne555.pdf>
- [http://www.ee.ic.ac.uk/pcheung/teaching/DE1\\_EE/stores/sg90\\_datasheet.pdf](http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf)

Inspiration

- <https://youtu.be/uJIIYiUP-J8>
- <https://youtube.com/shorts/-8SATs1iip4?feature=share>

Videodokumentation af en demonstration af modstandssorteringen:

- <https://youtu.be/H8uLL0rGSkU>
- <https://youtu.be/09GmGWsfW1E>
- <https://youtu.be/KqqIMkx1wqM>

Projektstyring med SCRUM:

- <https://youtu.be/HILs1kQWbd4>

## Bilag

### Bilag 1: Gruppekontrakt

1. Hvordan er gruppens holdning til at være til stede i timerne?
  - Vi er til stede, så vidt muligt. (Medmindre, der er tale om sundhedsmæssigt fravær) Dertil deltager vi aktivt i timerne.
  - Man må godt holde en aftalt pause, hvor man kan snakke med andre grupper eller helt andre elever et andet sted end i El-lab.
2. Vil I kunne arbejde sammen med andre i gruppen efter skoletid?
  - Ja, så længe det er muligt at få planlagt i sammenhæng med vores alles skema.
3. Vil gruppen være indstillet på at arbejde med opgaverne hjemme?
  - JA! Her kan det skriftlige arbejde udarbejdes og finpudsses.
  - Selve de fysiske moduler kan i stedet fungere som møder med fælles aftaler, fordeling af arbejde og mest af alt produktudvikling.
4. Hvad vil gruppen gøre, hvis den bliver utilfreds med en af medlemmernes arbejdsindsats?
  - Først og fremmest vil vi diskutere denne problematik med personen, og i fællesskab finde på løsninger til at kunne motivere personen fremadrettet.
5. Hvilket resultat vil gruppen samlet stræbe efter at opnå?
  - I vores gruppe vil vi stræbe efter at have det bedste gåpåmod og højeste ambitionsniveau, som tiden og ressourcerne tillader. Vi vil altid gøre vores bedste i dette projekt.
6. Hvor langt vil de enkelte medlemmer strække sig for at opfylde punkt 5?
  - De enkelte medlemmer vil gøre sit bedste ved at arbejde på projektet efter skoletid og deltage aktivt i timerne. Kæmpe indtil de fælles mål og aftaler er nået.

7. Hvordan kontaktes gruppen, hvis man pludselig må melde afbud?

- Kommunikationstjenesten: Discord. Her har gruppen en fælles discord-gruppe, hvor dokumentation og lignende også deles.

8. Hvad er lovligt fravær?

- Sundhedsmæssigt skyldende fravær, eller psykologiske årsager.
- På nuværende tidspunkt tager både Noah og Clara kørekort, og kan derfor være fraværende på grund af køretimer eller lignende.

9. Hvad stiller gruppen op, hvis en person, der skulle have sørget for noget vigtigt, bliver væk?

- Kontakter den enkelte person, og sørger for at alle dokumenter er delt med alle i gruppen. Derudover skal fuldstændigt individuelt arbejde begrænses.

10. Hvilke arbejdsopgaver tildeles/påtager gruppens medlemmer sig

- Diagramtegninger i KiCad og flowcharts: Noah
- Fremstilling af PCB: Noah
- Bore huller i PCB: Noah
- Programmering: Kristiyan
- 3D-modellering i Sketch-up eller Blender: Kristiyan og Clara
- 3D-printning af modellerne: Alle
- Tidsplan og SCRUM: Clara
- Dokumentation af projektet: Clara
- Lodning af komponenter på PCB: Clara

11. Hvem skal være ansvarlig for de skriftlige afleveringer?: Clara

12. Hvem skal være ansvarlig for produktet?: Kristiyan

## Bilag 2: Tidsplan

	Mandag	Tirsdag	Onsdag	Torsdag	Fredag
Uge 8	<b>Udlevering</b> 4	2			<b>Valg af oplæg og gruppe</b> 4
Uge 9	<b>Aftaleseddel</b> 4	2			
Uge 10					<b>Projektbeskrivelse</b> 4
Uge 11	4	2			4
Uge 12	4	2			4
Uge 13	4	2			<b>Materialeliste</b> 4
Uge 14	<b>Påske</b>	<b>Påske</b>	<b>Påske</b>	<b>Påske</b>	<b>Påske</b>
Uge 15	<b>Påske</b>	2			4
Uge 16	<b>Værksted</b> 8	<b>Værksted</b> 8	<b>Værksted</b> d 8	<b>Værksted</b> 8	<b>Værksted</b> 8
Uge 17	<b>Skrivedag</b> 8 <b>Aflevering produkt</b> <b>senest kl. 16</b>				<b>SØNDAG</b> <b>Aflevering rapport</b> <b>senest kl. 23</b>

**Gruppens tidsplan**

Dato	Uge nr.	Uge dag	Antal moduler	Skriftligt arbejde	Produktudvikling	Deadlines	Status
20/02/23	8	Mandag	2	Idégenerering Gruppedannelse		Udlevering af projekt	✓
21/02/23	8	Tirsdag	1	Skitser			✓
24/02/23	8	Fredag	2	Gruppekontrakt Skitser SCRUM	Indsamling af valgte modstande 3D-modellering af sorterings-idé	Valg af oplæg, gruppe og idé	✓
27/02/23	9	Mandag	2	Færdiggørelse af Tidsplan og Projektbeskrivelse	Påbegyndt program til programmering af Pickit i MPLab Færdiggørelse af 3D-model til sortering (Med servo-motor) Print 3D-modellen		✓
28/02/23	9	Tirsdag	1	SCRUM	Undersøgelse og beregning af udvalgte modstande med måling. PicKit programmering på breadboard	Aftaleseddel Materialeliste	✓
10/03/23	10	Fredag	2	SCRUM	PicKit programmering på breadboard Opbygning af måleapparat.	Projektbeskrivelse	✓
13/03/23	11	Mandag	2	SCRUM	PicKit programmering på breadboard		✓

					Kredsløbstegninger		
14/03/23	11	Tirsdag	1	SCRUM	PicKit programmering på breadboard KiCad til PCB-print		✓
17/03/23	11	Fredag	2	SCRUM	PicKit programmering på breadboard Fremstilling af PCB Lodning af PCB		✓
20/03/23	12	Mandag	2	SCRUM	PicKit programmering på breadboard Begyndelse af servo		✓
21/03/23	12	Tirsdag	1	SCRUM	PicKit programmering på PCB-print Servomotor med 3D-modellen		✓
24/03/23	12	Fredag	2	SCRUM	PicKit programmering på PCB-print Servomotor med 3D-modellen		✓
27/03/23	13	Mandag	2	SCRUM	PicKit programmering på PCB-print Servomotor med 3D-modellen		✓
28/03/23	13	Tirsdag	1	SCRUM	PicKit programmering på PCB-print Servomotor med 3D-modellen		✓
31/03/23	13	Fredag	2	SCRUM	PicKit programmering på PCB-print Servomotor med 3D-modellen	Materialeliste	✓
11/04/23	15	Tirsdag	1	SCRUM	PicKit programmering på PCB-print Servomotor med 3D-modellen		✓
14/04/23	15	Fredag	2	SCRUM	PicKit programmering på PCB-print Servomotor med 3D-modellen		✓

17/04/23	16	Man dag	4	SCRUM	PicKit programmering på PCB-print Servomotor med 3D-modellen		<input checked="" type="checkbox"/>
18/04/23	16	Tirsd ag	4	SCRUM	PicKit programmering på PCB-print Servomotor med 3D-modellen		<input checked="" type="checkbox"/>
19/04/23	16	Ons dag	4	SCRUM	PicKit programmering på PCB-print Servomotor med 3D-modellen		<input checked="" type="checkbox"/>
20/04/23	16	Tors dag	4	SCRUM	PicKit programmering på PCB-print Servomotor med 3D-modellen		<input checked="" type="checkbox"/>
21/04/23	16	Fred ag	4	SCRUM	PicKit programmering på PCB-print Finpuds funktioner i sorteringen og LCD-skærm skal optimeres. Servomotor skal virke		<input checked="" type="checkbox"/>
24/04/23	17	Man dag	4	SCRUM	Færdiggørelse af produktet og løsning	Aflevering af produktet kl 16	<input checked="" type="checkbox"/>
30/04/23	17	Sønd ag	0	Færdiggørelse af rapporten		Aflevering af rapporten	<input checked="" type="checkbox"/>
I alt:		24	52				

## Bilag 3: Programmeringen

```
#define _XTAL_FREQ 16000000

#define RS RE1
#define EN RE0
#define D4 RD4
#define D5 RD5
#define D6 RD6
#define D7 RD7

#include <xc.h>

#include "lcd_lib_header.h" // Includes the LCD library
#include "stdlib.h"
#include <string.h>
#include <stdio.h>
#include <math.h>

#pragma config FOSC = HS          // Oscillator Selection bits (HS oscillator)
#pragma config WDTE = OFF         // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = ON          // Power-up Timer Enable bit (PWRT enabled)
#pragma config BOREN = ON          // Brown-out Reset Enable bit (BOR enabled)
#pragma config LVP = OFF           // Low-Voltage (Single-Supply) In-Circuit Serial
Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used for
programming)

#pragma config CPD = OFF           // Data EEPROM Memory Code Protection bit (Data
EEPROM code protection off)
#pragma config WRT = OFF           // Flash Program Memory Write Enable bits (Write
protection off; all program memory may be written to by EECON control)
#pragma config CP = OFF            // Flash Program Memory Code Protection bit (Code
protection off)

void DelayUp(uint8_t n) { // Pre defined functions for servo
    for (; n > 0; n--)      // Runs until n = 0
    {
        __delay_us(10);     // Adds a delay of 10 microseconds
    }
}
```

```

        } // This way if we set delay up of 10, 100 microseconds
will go by
}

void DelayDown(uint8_t n) { // Same idea here
    n = 2000-n; // Only this time it's 20 000 microseconds minus n
    for (; n > 0; n--) // This way we can have a 20 ms pulse
    {
        __delay_us(10);
    }
}

void ADC_Init() // Initializes Analog Digital Converter
{
    ADCON0=0b00000001; // Sets AD converter to on
    ADCON1=0b10001110; // Right justified and only using AN0 for analog input
}

int ADC_Read_Int() // Reads the ADC output when called as int
{
    GO_DONE = 1;
    while(GO_DONE == 1);
    return(ADRESH*256 + ADRESL);
}

float ADC_Read_Float() // Reads the ADC output when called as float
{
    GO_DONE = 1;
    while(GO_DONE == 1);
    return(ADRESH*256 + ADRESL);
}

int main()
{
    unsigned int a;
    unsigned int i;
    int resistance = 0; // Default resistance
    int servo = 0; // Default servo
}

```

```
char str[1]; // Used to show digits correctly on the LCD
TRISB = 0x00; // Servo
TRISC = 0x00; // Motor RC0 and Spændingsdeler RC1
TRISD = 0x00; // LCD
TRISE = 0x00; // LCD
Lcd_Start();
ADC_Init();

for(i=0;i<1000;i++) { // Sets the servo at 0,
    PORTB = 1;          // Has power
    DelayUp(150);      // Delay of 1.5ms
    PORTB = 0;          // No power
    DelayDown(150);    // Delay of 20 - 1.5 ms
}
// This way we can control the servo
while(1)
{
    PORTC = 0b00000001; // Turns on the motor
    if (ADC_Read_Int() > 0) { // Check if there's connection
        __delay_ms(1000); // 1 second delay to make sure the connection is
good
        if (ADC_Read_Int() == 1008) { // If resistance is 100 // These were
all tested by hand
            resistance = 100; // sets the correct resistance
            servo = 90; // sets the servo
        } else if (ADC_Read_Int() == 992) { // 220
            resistance = 220;
            servo = 100;
        } else if (ADC_Read_Int() == 960) { // 330
            resistance = 330;
            servo = 110;
        } else if (ADC_Read_Int() == 928) { // 470
            resistance = 470;
            servo = 120;
        }
    }
}
```

```

} else if (ADC_Read_Int() == 839) { // 1000
    resistance = 1000;
    servo = 140;

} else if (ADC_Read_Int() == 704) { // 2200
    resistance = 2200;
    servo = 165;

} else if (ADC_Read_Int() == 512) { // 4700
    resistance = 4700;
    servo = 180;

} else if (ADC_Read_Int() == 480) { // 5600
    resistance = 5600;
    servo = 195;

} else if (ADC_Read_Int() == 323) { // 10000
    resistance = 10000;
    servo = 70;

} else if (ADC_Read_Int() > 845) { // Check if resistance is under 1k
// If it's not hard coded then estimate

    PORTC = 0b00000011; // Turn on 820 resistor // This way we can
have more precise measurements

    resistance = (1024/ADC_Read_Float() - 1) * 698; // Some math
    servo = 130;
    PORTC = 0b00000001;

} else if (ADC_Read_Int() < 320) { // Over 10k
//PORTC = 0b00000101; // Turn on 82k resistor // Won't work
because it's parallel with 4k7

    resistance = (1024/ADC_Read_Float() - 1) * 4700;
    servo = 80;
//PORTC = 0b00000001;

} else { // 1k - 10k
    resistance = (1024/ADC_Read_Float() - 1) * 4700;
    servo = 150;
}

}

```

```
if (resistance > 0) { // If resistance is recorded

    // Display

    Lcd_Clear();           // Clear the display
    Lcd_Set_Cursor(1,1);   // Set cursor at the top
    sprintf(str, "%d", resistance); // Add the resistance to str
    Lcd_Print_String(str); // Print str

    // Servo

    for(i=0;i<1000;i++) { // Exactly the same as above, but instead of
150 it's the variable servo

        PORTB = 1;
        DelayUp(servo);
        PORTB = 0;
        DelayDown(servo);

    }

    while (ADC_Read_Int() > 0) { // Loop while ADC can still detect
resistance in the wheels

        __delay_ms(100);
    }

    // Restore

    __delay_ms(3000); // Some delay to make sure the resistor has left
the chamber (originally the servo would have spun back to 0)

    resistance = 0; // Resets both variables here
    servo = 0;

} else { // Else while resistance is not being detected

    // Default

    Lcd_Clear();
    Lcd_Set_Cursor(1,1);
    Lcd_Print_String("--0000--"); // Prints instead this string which
looks like a resistor on the screen

    __delay_ms(100);

}

} // That's about it - and endless cycle of checking if a resistor is
detected
}
```

## Bilag 4: Registre til PIC16F877a

- <https://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf>

**FIGURE 2-3: PIC16F876A/877A REGISTER FILE MAP**

File Address	File Address	File Address	File Address	File Address
Indirect addr. <sup>(*)</sup>	00h	Indirect addr. <sup>(*)</sup>	80h	Indirect addr. <sup>(*)</sup>
TMR0	01h	OPTION_REG	81h	TMR0
PCL	02h	PCL	82h	PCL
STATUS	03h	STATUS	83h	STATUS
FSR	04h	FSR	84h	FSR
PORTA	05h	TRISA	85h	
PORTB	06h	TRISB	86h	PORTB
PORTC	07h	TRISC	87h	
PORTD <sup>(1)</sup>	08h	TRISD <sup>(1)</sup>	88h	
PORTE <sup>(1)</sup>	09h	TRISE <sup>(1)</sup>	89h	
PCLATH	0Ah	PCLATH	8Ah	PCLATH
INTCON	0Bh	INTCON	8Bh	INTCON
PIR1	0Ch	PIE1	8Ch	EEDATA
PIR2	0Dh	PIE2	8Dh	EEADR
TMR1L	0Eh	PCON	8Eh	EEDATH
TMR1H	0Fh		8Fh	EEADRH
T1CON	10h		90h	
TMR2	11h	SSPCON2	91h	
T2CON	12h	PR2	92h	
SSPBUF	13h	SSPADD	93h	
SSPCON	14h	SSPSTAT	94h	
CCPR1L	15h		95h	
CCPR1H	16h		96h	
CCP1CON	17h		97h	General Purpose Register 16 Bytes
RCSTA	18h	TXSTA	98h	
TXREG	19h	SPBRG	99h	
RCREG	1Ah		9Ah	
CCPR2L	1Bh		9Bh	
CCPR2H	1Ch	CMCON	9Ch	
CCP2CON	1Dh	CVRCON	9Dh	
ADRESH	1Eh	ADRESL	9Eh	
ADC0N0	1Fh	ADCON1	9Fh	
General Purpose Register 96 Bytes	20h		A0h	General Purpose Register 80 Bytes
			EFh	General Purpose Register 80 Bytes
			F0h	accesses 70h-7Fh
			FFh	accesses 70h-7Fh
Bank 0	7Fh	Bank 1	Bank 2	Bank 3
 Unimplemented data memory locations, read as '0'. * Not a physical register.				
<b>Note 1:</b> These registers are not implemented on the PIC16F876A. <b>2:</b> These registers are reserved; maintain these registers clear.				

**FIGURE 2-4: PIC16F873A/874A REGISTER FILE MAP**

File Address	File Address	File Address	File Address
Indirect addr. <sup>(1)</sup>	Indirect addr. <sup>(1)</sup>	Indirect addr. <sup>(1)</sup>	Indirect addr. <sup>(1)</sup>
00h TMR0	01h OPTION_REG	00h TMR0	00h Indirect addr. <sup>(1)</sup>
02h PCL	02h PCL	01h OPTION_REG	01h Indirect addr. <sup>(1)</sup>
03h STATUS	03h STATUS	02h PCL	02h Indirect addr. <sup>(1)</sup>
04h FSR	04h FSR	03h STATUS	03h Indirect addr. <sup>(1)</sup>
05h PORTA	05h TRISA	04h FSR	04h Indirect addr. <sup>(1)</sup>
06h PORTB	06h TRISB	05h PORTB	05h Indirect addr. <sup>(1)</sup>
07h PORTC	07h TRISC	06h TRISB	06h Indirect addr. <sup>(1)</sup>
08h PORTD <sup>(1)</sup>	08h TRISD <sup>(1)</sup>	07h TRISC	07h Indirect addr. <sup>(1)</sup>
09h PORTE <sup>(1)</sup>	09h TRISE <sup>(1)</sup>	08h TRISD <sup>(1)</sup>	08h Indirect addr. <sup>(1)</sup>
PCLATH	PCLATH	09h TRISE <sup>(1)</sup>	09h Indirect addr. <sup>(1)</sup>
INTCON	INTCON	0Ah PCLATH	0Ah PCLATH
PIR1	PIE1	0Bh INTCON	0Bh INTCON
PIR2	PIE2	0Ch EEDATA	0Ch EECON1
TMR1L	PCON	0Dh EEADR	0Dh EECON2
TMR1H		0Eh EEDATH	0Eh Reserved <sup>(2)</sup>
T1CON		0Fh EEADRH	0Fh Reserved <sup>(2)</sup>
TMR2	SSPCON2	10h	10h
T2CON	PR2	11h	11h
SSPBUF	SSPADD	12h	12h
SSPCON	SSPSTAT	13h	13h
CCPR1L		14h	91h
CCPR1H		15h	92h
CCP1CON		16h	93h
RCSTA		17h	94h
TXREG	TXSTA	18h	95h
RCREG	SPBRG	19h	96h
CCPR2L		1Ah	97h
CCPR2H		1Bh	98h
CCP2CON	CMCON	1Ch	99h
ADRESH	CVRCON	1Dh	9Ah
ADCON0	ADRESL	1Eh	9Bh
	ADCON1	1Fh	9Ch
		20h	9Dh
General Purpose Register 96 Bytes	General Purpose Register 96 Bytes	FFh	9Eh
Bank 0	7Fh	Bank 1	FFh
			120h
			A0h
		accesses 20h-7Fh	16Fh
			170h
			17Fh
			Bank 2
			accesses A0h - FFh
			16Fh
			170h
			17Fh
			Bank 3
			1EFh
			1F0h
			1FFh

Unimplemented data memory locations, read as '0'.  
 \* Not a physical register.

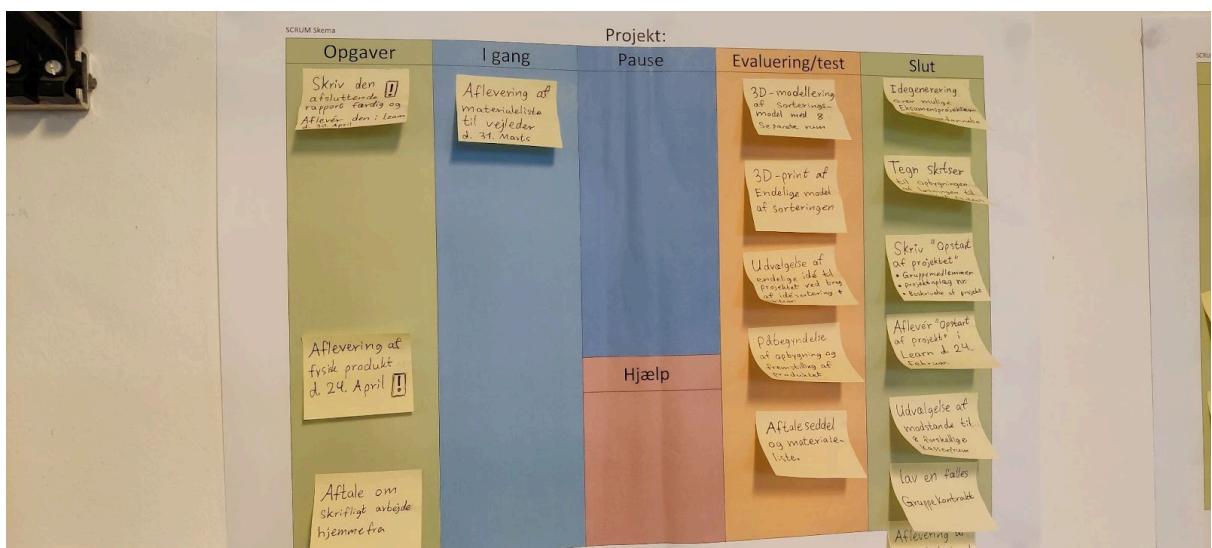
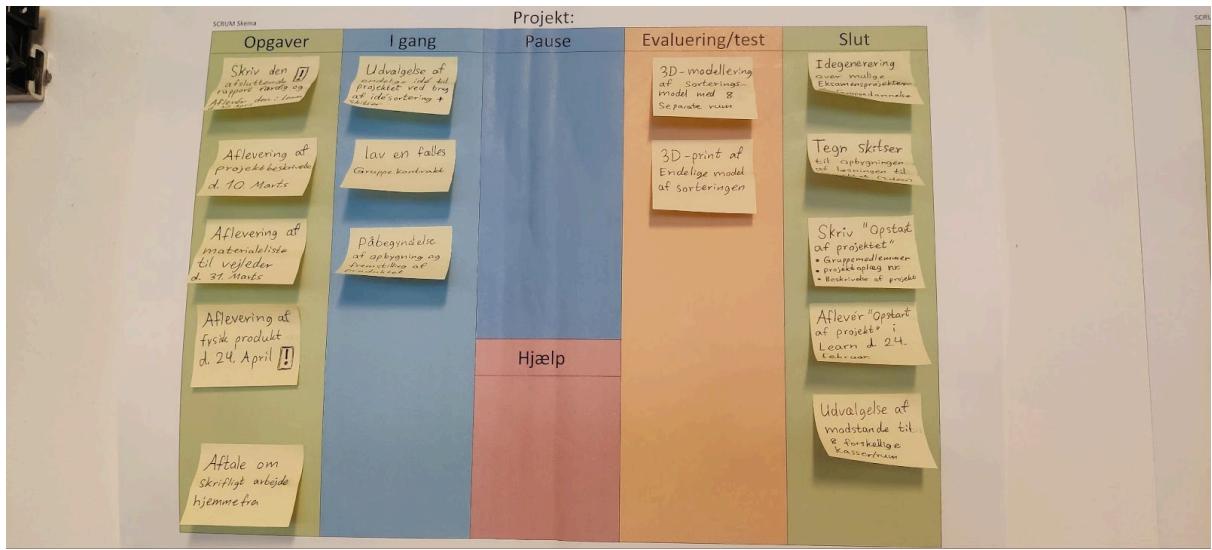
**Note 1:** These registers are not implemented on the PIC16F873A.

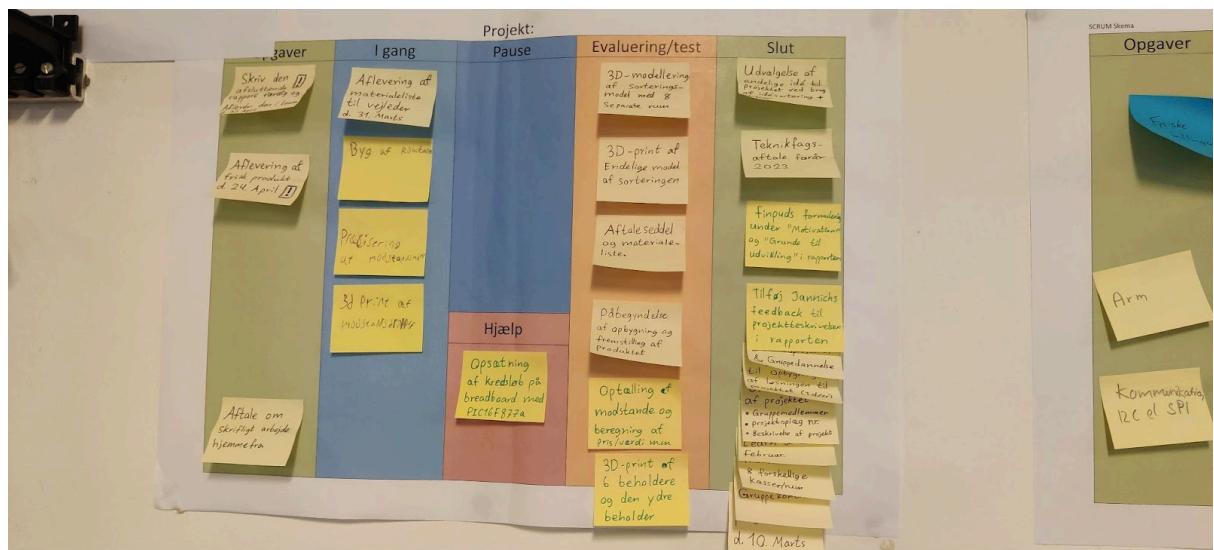
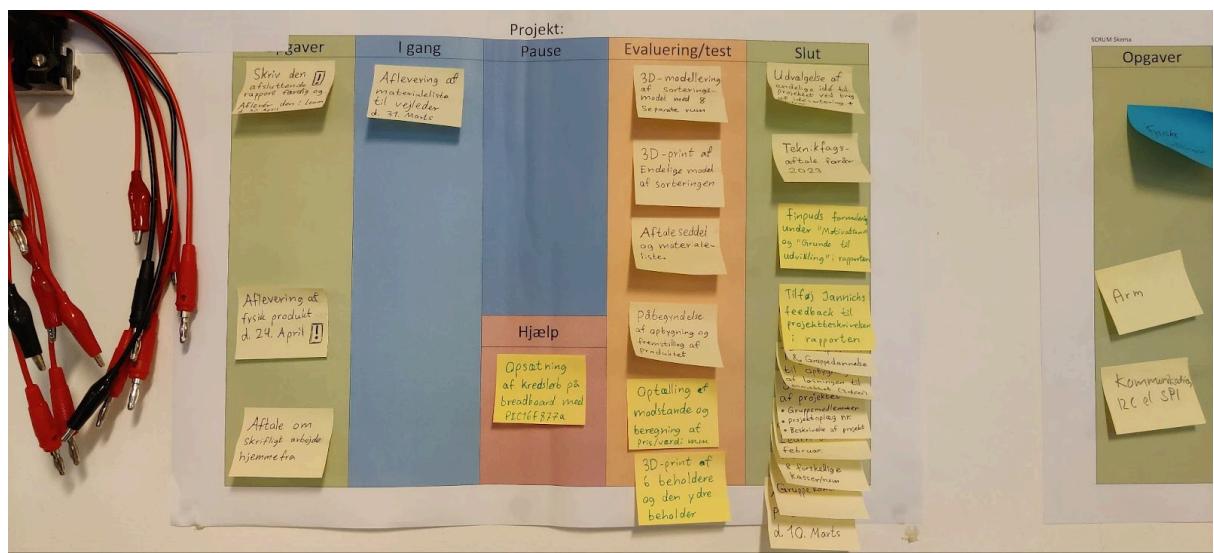
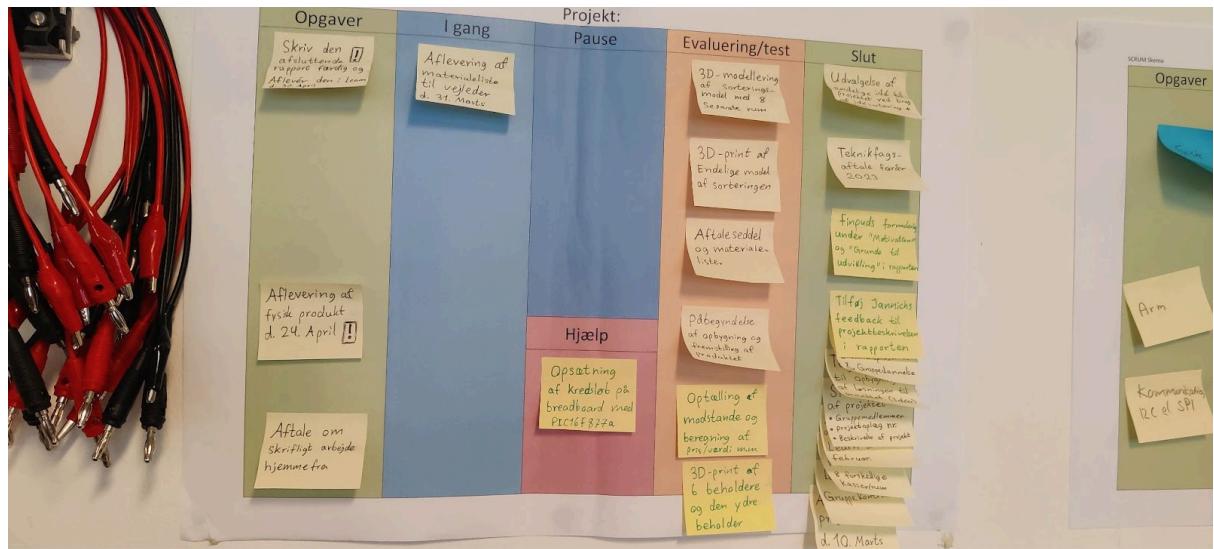
**Note 2:** These registers are reserved; maintain these registers clear.

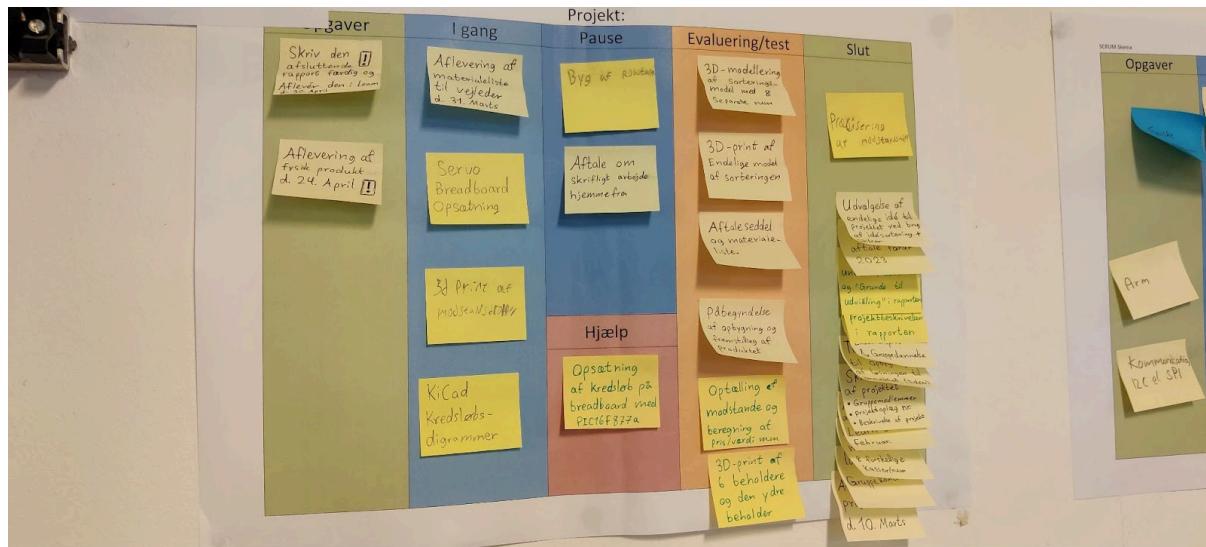
**TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY**

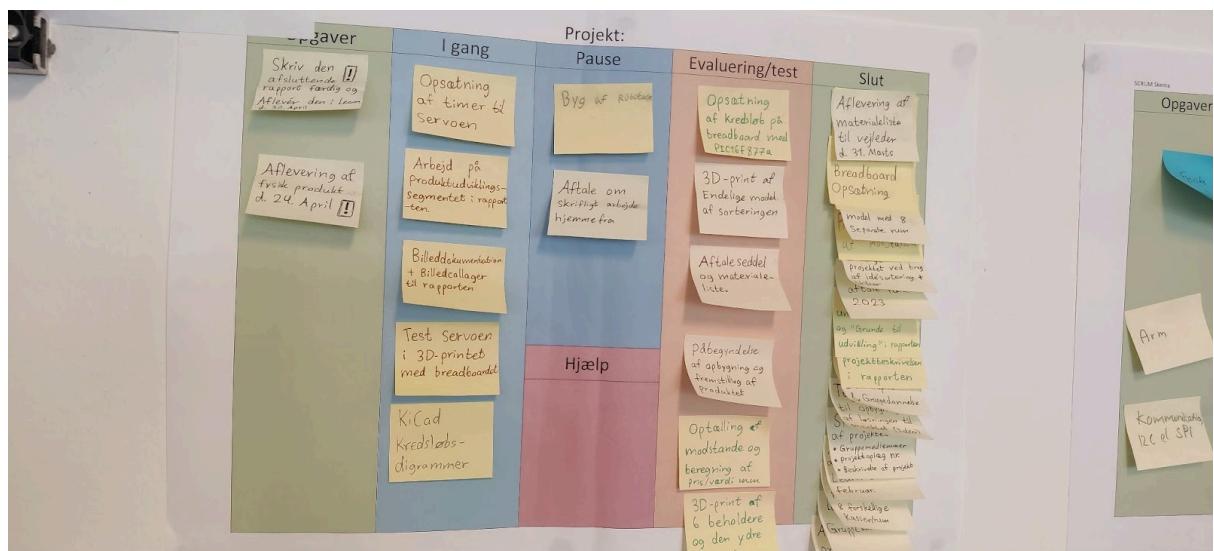
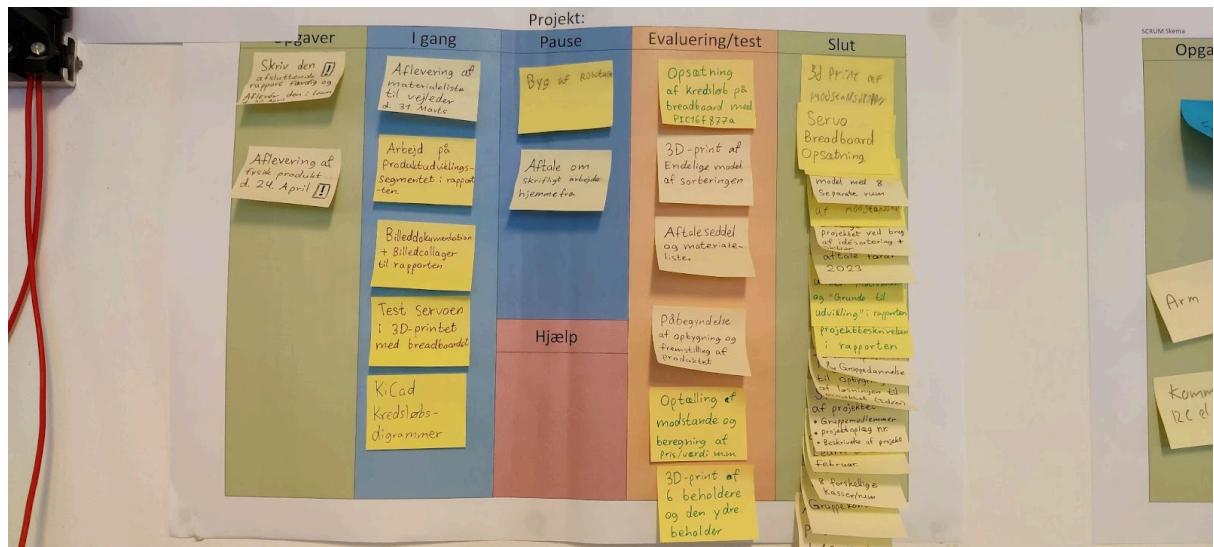
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:	
<b>Bank 0</b>												
00h <sup>(3)</sup>	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150	
01h	TMR0	Timer0 Module Register								xxxxx xxxx	55, 150	
02h <sup>(3)</sup>	PCL	Program Counter (PC) Least Significant Byte								0000 0000	30, 150	
03h <sup>(3)</sup>	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	0001 1xxxx	22, 150	
04h <sup>(3)</sup>	FSR	Indirect Data Memory Address Pointer								xxxxx xxxx	31, 150	
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read								
06h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxxx xxxx	45, 150	
07h	PORTC	PORTC Data Latch when written: PORTC pins when read								xxxxx xxxx	47, 150	
08h <sup>(4)</sup>	PORTD	PORTD Data Latch when written: PORTD pins when read								xxxxx xxxx	48, 150	
09h <sup>(4)</sup>	PORTE	—	—	—	—	RE2	RE1	RE0	----	-xxxx	49, 150	
0Ah <sup>(1,3)</sup>	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter							
0Bh <sup>(3)</sup>	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150	
0Ch	PIR1	PSPIF <sup>(3)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	26, 150	
0Dh	PIR2	—	CMIF	—	EEIF	BCLIF	—	—	CCP2IF	-0-0 0--0	28, 150	
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxxx xxxx	60, 150	
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxxx xxxx	60, 150	
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	57, 150	
11h	TMR2	Timer2 Module Register								0000 0000	62, 150	
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	61, 150	
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxxx xxxx	79, 150	
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	82, 82, 150	
15h	CCPR1L	Capture/Compare/PWM Register 1 (LSB)								xxxxx xxxx	63, 150	
16h	CCPR1H	Capture/Compare/PWM Register 1 (MSB)								xxxxx xxxx	63, 150	
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	64, 150	
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	112, 150	
19h	TXREG	USART Transmit Data Register								0000 0000	118, 150	
1Ah	RCREG	USART Receive Data Register								0000 0000	118, 150	
1Bh	CCPR2L	Capture/Compare/PWM Register 2 (LSB)								xxxxx xxxx	63, 150	
1Ch	CCPR2H	Capture/Compare/PWM Register 2 (MSB)								xxxxx xxxx	63, 150	
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	64, 150	
1Eh	ADRESH	A/D Result Register High Byte								xxxxx xxxx	133, 150	
1Fh	ADC0N0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON	0000 00-0	127, 150	

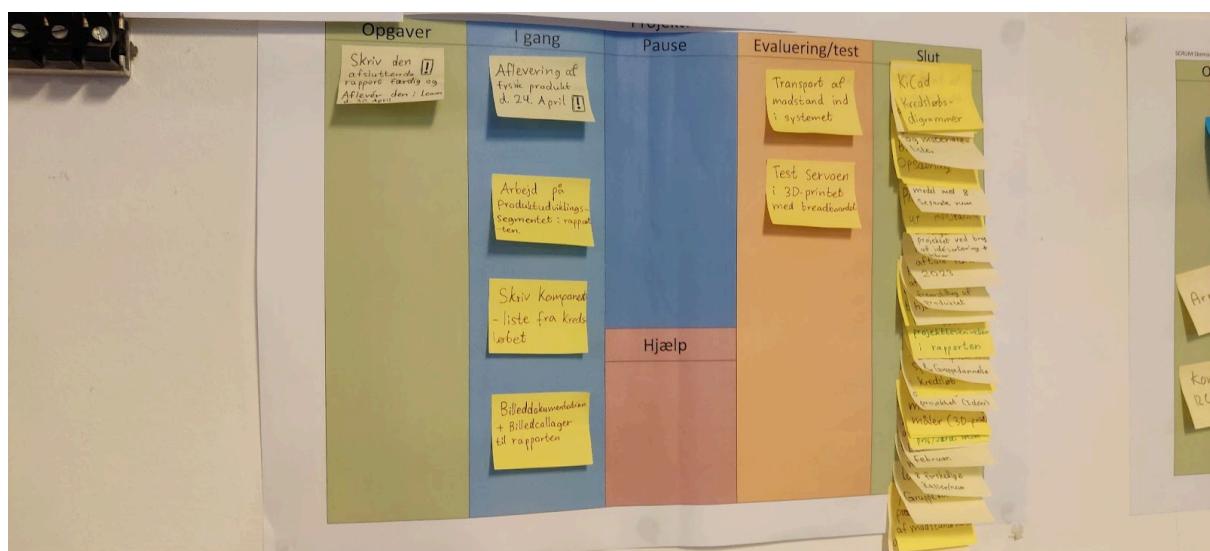
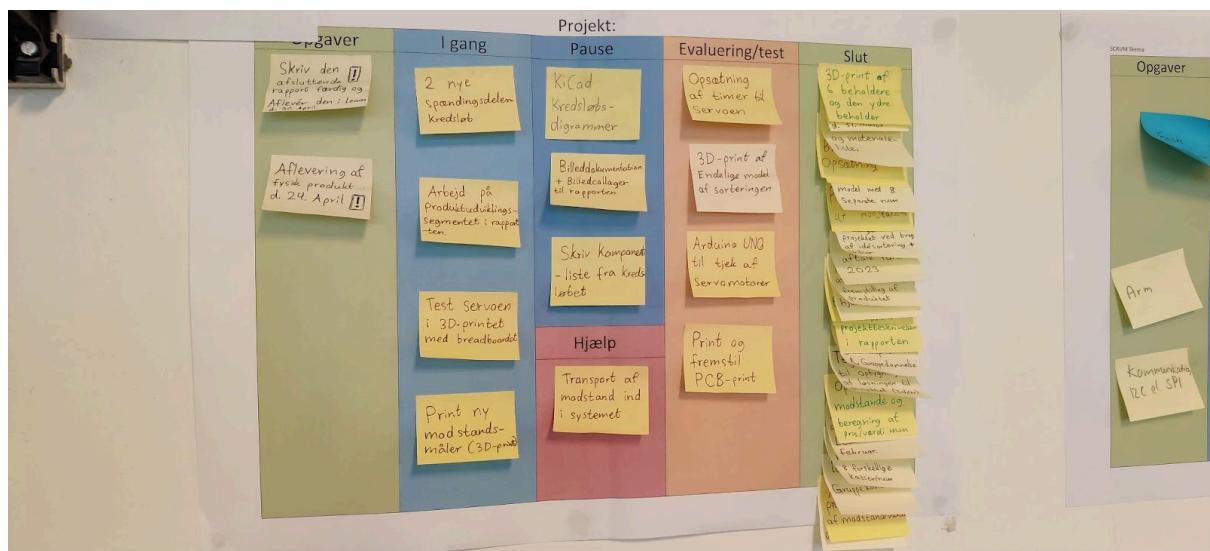
## Bilag 5: SCRUM-projektstyring











## Bilag 6: Alle beregninger

### Identificering af modstande

#### Liste af beholdere

1.  $100 \Omega$
2.  $220 \Omega$
3.  $330 \Omega$
4.  $470 \Omega$
5. Modstande lavere end 1k (Rest-skuffe 1)
6.  $1k \Omega$
7.  $2,2k \Omega$
8.  $4,7k \Omega$
9.  $5,6k \Omega$
10. Modstande højere end 1k (Rest-skuffe 2)
11.  $10k \Omega$
12. Andre komponenter, der ikke er modstande (Rest-skuffe 3)

#### Formler

- Ohms lov:  $U = R \cdot I$

Kan omisoleres til:

- $V = I \cdot R$  (Spænding = Strømstyrke ganget med resistansen/modstanden)
- $R = \frac{V}{I}$  (Resistance = Voltage divided by Current)
- $I = \frac{V}{R}$  (Current = Voltage Divided by Resistance)

**Ved brug af Ohms lov**

- 100  $\Omega$  (Formodstand til 5 V til LED)
  - Spænding: 5 V
  - Strømstyrke: 0.05 A
- 220  $\Omega$  (Hyppigt anvendt)
  - Spænding: 10 V
  - Strømstyrke: 0.05 A
- 330  $\Omega$  (Formodstand til 9 V til LED)
  - Spænding: 9 V
  - Strømstyrke: 0.03 A
- 470  $\Omega$  (Formodstand til 12 V til LED)
  - Spænding: 12 V
  - Strømstyrke: 0.025 A
- Modstande lavere end 1k (Rest-skuffe 1)
  - Spænding: 10 V
  - Strømstyrke: 0.01-10 A
- 1k  $\Omega$  (Hyppigt anvendt)
  - Spænding: 10 V
  - Strømstyrke: 0.01 A
- 2,2k  $\Omega$  (Hyppigt anvendt)
  - Spænding: 10 V
  - Strømstyrke: 0.0045 A
- 4,7k  $\Omega$  (Hyppigt anvendt)
  - Spænding: 10 V
  - Strømstyrke: 0.002 A
- 5,6k  $\Omega$  (Hyppigt anvendt)
  - Spænding: 10 V
  - Strømstyrke: 0.0018 A
- Modstande højere end 1k (Rest-skuffe 2)
  - Spænding: 10 V
  - Strømstyrke: 0.001-0.01 A
- 10k  $\Omega$  (Hyppigt anvendt)
  - Spænding: 10 V
  - Strømstyrke: 0.001 A

**Ved brug af spændingsdeler formlen****Når R1 = 100 Ω:**

$$U_2 = U \cdot \frac{R_2}{R_1 + R_2} = U_2 = 5 \cdot \frac{4700}{100 + 4700} = 4.88 \text{ V}$$

**Når R1 = 220 Ω:**

$$U_2 = U \cdot \frac{R_2}{R_1 + R_2} = U_2 = 5 \cdot \frac{4700}{220 + 4700} = 4.44 \text{ V}$$

**Når R1 = 330 Ω:**

$$U_2 = U \cdot \frac{R_2}{R_1 + R_2} = U_2 = 5 \cdot \frac{4700}{330 + 4700} = 4.25 \text{ V}$$

**Når R1 = 470 Ω:**

$$U_2 = U \cdot \frac{R_2}{R_1 + R_2} = U_2 = 5 \cdot \frac{4700}{470 + 4700} = 3.85 \text{ V}$$

**Når R1 = 1k Ω:**

$$U_2 = U \cdot \frac{R_2}{R_1 + R_2} = U_2 = 5 \cdot \frac{4700}{1000 + 4700} = 2.72 \text{ V}$$

**Når R1 = 2.2k Ω:**

$$U_2 = U \cdot \frac{R_2}{R_1 + R_2} = U_2 = 5 \cdot \frac{4700}{2200 + 4700} = 1.87 \text{ V}$$

**Når R1 = 4.7k Ω:**

$$U_2 = U \cdot \frac{R_2}{R_1 + R_2} = U_2 = 5 \cdot \frac{4700}{4700 + 4700} = 2.5 \text{ V}$$

**Når R1 = 5.6k Ω:**

$$U_2 = U \cdot \frac{R_2}{R_1 + R_2} = U_2 = 5 \cdot \frac{4700}{5600 + 4700} = 2.03 \text{ V}$$

**Når R1 = 10k Ω:**

$$U_2 = U \cdot \frac{R_2}{R_1 + R_2} = U_2 = 5 \cdot \frac{4700}{10000 + 4700} = 1.39 \text{ V}$$

**Vinkler til beholderne (Servo)**

1.  $100 \Omega$ 
  - 1 - 30
  - 15
2.  $220 \Omega$ 
  - 31 - 60
  - 45,5
3.  $330 \Omega$ 
  - 61 - 90
  - 75,5
4.  $470 \Omega$ 
  - 91 - 120
  - 105,5
5. Modstande lavere end 1k (Rest-skuffe 1)
  - 121 - 150
  - 135,5
6.  $1k \Omega$ 
  - 151 - 180
  - 165,5
7. Modstande højere end 1k (Rest-skuffe 2)
  - 181 - 210
  - 195,5
8.  $2,2k \Omega$ 
  - 211 - 240
  - 225,5
9.  $4,7k \Omega$ 
  - 241 - 270
  - 255,5
10.  $5,6k \Omega$ 
  - 271 - 300
  - 285,5
11.  $10k \Omega$ 
  - 301 - 330
  - 315,5
12. Andre komponenter, der ikke er modstande (Rest-skuffe 3)
  - 331 - 360 (0)
  - 335,5

### Usikkerhed på 5% til og fra alle modstande

1.  $100 \Omega$ 
  - 95 - 105
2.  $220 \Omega$ 
  - 209 - 231
3.  $330 \Omega$ 
  - 313,5 - 346,5
4.  $470 \Omega$ 
  - 446,5 - 493,5
5. Modstande lavere end 1k (Rest-skuffe 1)
  - Værdi != øvrige variationsbredder og lavere end 1k
6.  $1k \Omega$ 
  - 950 - 1010
7. Modstande højere end 1k (Rest-skuffe 2)
  - Værdi != øvrige variationsbredder og højere end 1k
8.  $2,2k \Omega$ 
  - 2090 - 2310
9.  $4,7k \Omega$ 
  - 4465 - 4935
10.  $5,6k \Omega$ 
  - 5320 - 5880
11.  $10k \Omega$ 
  - 9500 - 10500
12. Andre komponenter, der ikke er modstande (Rest-skuffe 3)
  - Over 10k i stedet for at medtage andre komponenter ???
  - Værdi != øvrige variationsbredder og højere end 10k

---

*Vi håber, at I nød at læse rapporten!*