CAPY

Teknisk rapport

Fagkode:IN1060 - Bruksorientert design

Gruppe: CAPY

Gruppemedlemmer: Cathrine Framstad Thurmann-Nielsen, Astrid Stenvold Malm, Chi Yin Philip Lo, Yahya Isam

Antall ord: 2706

Dato: 13.06.22

UNIVERSITETET
I OSLO

Innholdsfortegnelse

1. Introduksjon

1.1. Målgruppe

Målgruppen vi har valgt å jobbe med definerer vi som *unge voksne i alderen 20 til 29, som aktivt har spilt dataspill i flere år, og anser seg selv som en gamer*.

1.2. Tema

Temaet vårt er *søvn*, og **problemstillingen** vår er *"hvordan få gamere til å avslutte dataspillet for å unngå søvnmangel eller uønsket døgnrytme"*.

1.3. Konsept, formkonsept og visjon

Konseptet vårt er *lys*, og **formkonseptet** vårt er *timeglass*. Vår **visjon** er at *brukerne skal kunne se lyst på en fungerende døgnrytme*.

1.4. Målet med prosjektet

Ut ifra en idémyldringsøkt tidlig i prosjektet, kom vi frem til at gamere var en interessant målgruppe. Etter flere iterasjoner med datainnsamling og analyse, kom det frem at behov for søvn var et aktuelt og interessant behov å oversette som et krav. Det ble tydelig at våre brukeres aktivitet gikk på bekostning av deres døgnrytme. En bruker forskjøv hele timeplanen sin, han prioriterte å sove 7-8 timer uansett hvor sent han gamet til. En annen bruker fikk mindre søvn fordi han måtte rekke jobb dagen etter. Under andre iterasjon samlet vi data fra brukerne våre. Til tross for stor ulikhet mellom brukerne hadde alle noen utfordringer med å huske "leggetid" under aktiviteten gaming. Vi valgte å løse deres utfordring ved å lage et timeglass som gjør dem oppmerksom på deres "leggetid". I tillegg ønsker vi å gi brukeren funksjonalitet for å stille inn på både når brukeren ønsker å stå opp, og hvor mange timer med søvn brukeren ønsker å få. Dette skal videre kobles opp med timeglasset som skal illustrere hvor lenge det er igjen til at brukeren ønsker å legge seg.

En nøye beskrivelse av idégenereringen, undersøkelsene og analysene som brakte oss frem til denne problemstillingen kan leses om i hovedrapporten.

1.5. Krav

Funksjonelle krav	**Forklaring**
Artefaktet viser at tiden har gått ut.	Artefaktet skal løse brukernes problem med å holde oversikt over tiden.
Artefaktet skal kunne brukes i et rom der lyset er påskrudd.	De fleste av brukerne har på litt lys mens de gamer, og det er viktig at lysene er sterke nok til å synes når lyset er på.
Artefaktet skal kunne brukes i et rom der lyset er avskrudd.	Det er også viktig at artefaktet ikke lyser så sterkt at det blir ubehagelig for brukeren å se på det før 'leggetid'.
Artefaktet skal gi feedback i form av lys.	Det er viktig for brukerne å ikke forstyrres i aktiviteten, lys er en varslende, men lite forstyrrende form for feedback.
Brukeren skal kunne stille inn når hen vil stå opp.	Det er viktig at brukerne kan bruke artefaktet uavhengig når de selv ønsker å stå opp.
Brukeren skal kunne stille inn antall ønsket timer søvn.	Artefaktet skal oppfordre til en jevnere døgnrytme og komme seg i seng til ønsket tid, men brukeren skal likevel selv kunne velge hvor mange timer søvn de ønsker.

Alle de grønne diodene skal lyse når all tid er igjen.	Grønn representerer start. Ingen av brukerne våre er fargeblinde, og vi valgte derfor å bruke grønn for å representere den første halvdelen av tiden.
Når tiden er ute skal alle de grønne diodene være avskrudd.	Når første halvdel av tiden er ferdig er det viktig at det er tydelig for brukeren at det er de rød som lyser.
Når tiden er ute skal alle de rød diodene lyse.	Rød representerer slutt.
Alle de rød diodene skal være avskrudd når tiden er fylt opp.	I første halvdel av tiden er det viktig at det er tydelig for brukeren at det er de grønne som lyser.
Artefakten skal kunne nullstilles.	Det er praktisk for brukeren å kunne nullstille artefaktet uten å måtte dra kontakten ut og inn.
Det skal være en 24-timers klokke for ønsket tid å stå opp.	Artefaktet skal ikke oppfordre brukeren til en bestemt døgnrytme, men heller være en oppfordring til jevn døgnrytme og nok antall timer søvn. En 24-timers klokke gir liten sjans for forvirring i motsetning til en 12-timers klokke.
Artefakten nullstilles ved å stille antall timer søvn til 0 og ønsket tid å stå opp til 00:00.	Brukerne ønsket å ikke ha for mange interaksjonsmomenter i artefaktet. Det ble dermed formålsmessig å bruke potensiometerne vi allerede hadde implementert enn å legge til en knapp i tillegg.

Brukerne skal kunne se hvor lang tid som er igjen.	Det skal være intuitivt for brukeren å forstå hvor lang tid som er igjen til hen må legge seg.

Tabell 1: Tabell med oversikt over de funksjonelle kravene vi har satt til artefaktet.

Ikke-funksjonelle krav	**Forklaring**
Det skal være mulig å vri 365º rundt med potensiometerne.**	For å tilsvare en hel sirkel.

Tabell 2: Tabell med oversikt over de ikke-funksjonelle kravene vi har satt til artefaktet.

** Ønsket funksjonalitet, men ikke implementert i endelig løsning

2. Utforskning

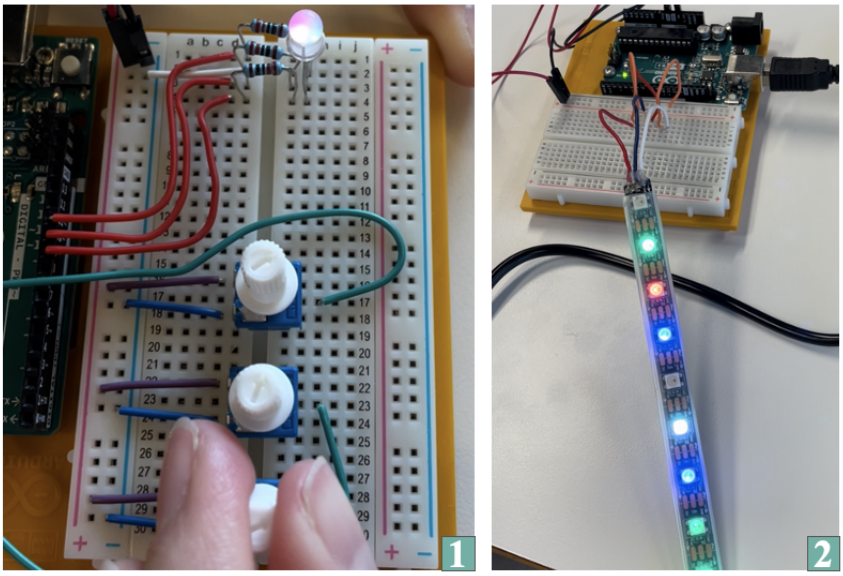
Vi har brukt Arduino-boken (Fitzgerald & Shilo, 2012) gjennom hele prosjektet for å lære og bli inspirert.

2.1. RGB-lys

2.1.1. Bakgrunn

I utgangspunktet ønsket vi å bruke RGB-lys for å kunne endre farge. En av tankene bak dette var å kunne endre fra grønt (over halvparten av tiden igjen) til gult (halvparten av tiden igjen) til rødt (under halvparten av tiden igjen).

2.1.2. Utforskning



Figur 1: Utforskning 1: RGB-lys med fire ben, utforskning 2: RGB-lysstripe

Vi utforsket med RGB-lysstripe. Først måtte vi sette oss inn i Adafruit sitt Neopixel-bibliotek. Alle LED-lysene kan kodes individuelt, og det hadde vært optimalt å bruke denne funksjonalitet i timeglasset. Da kunne vi lekt med overgangen fra grønt til gult til rødt. Her vurderte vi også om vi skulle gå til innkjøp av en RGB-matrise som også kunne visualisert sandkorn som faller.

2.1.3. Konklusjon

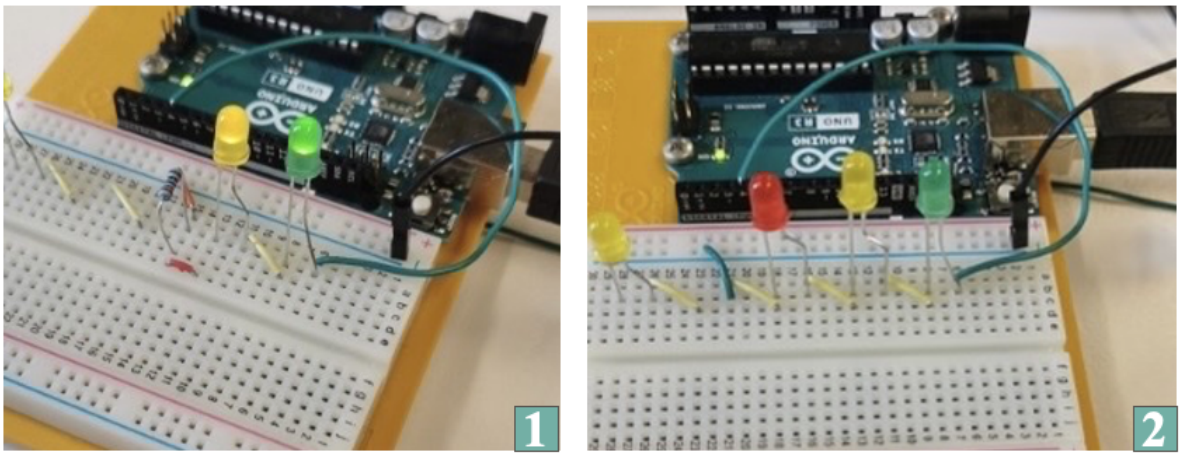
Den største årsaken til at vi gikk bort i fra RGB-lys var kostnadene med å gå til innkjøp av en RGB-lysstripe eller flere RGB LED-lys. Dette førte til at vi landet på å bruke de rød og grønne lysdioene vi hadde i settene våre.

2.2. Seriekobling vs. parallellkobling

2.2.1. Bakgrunn

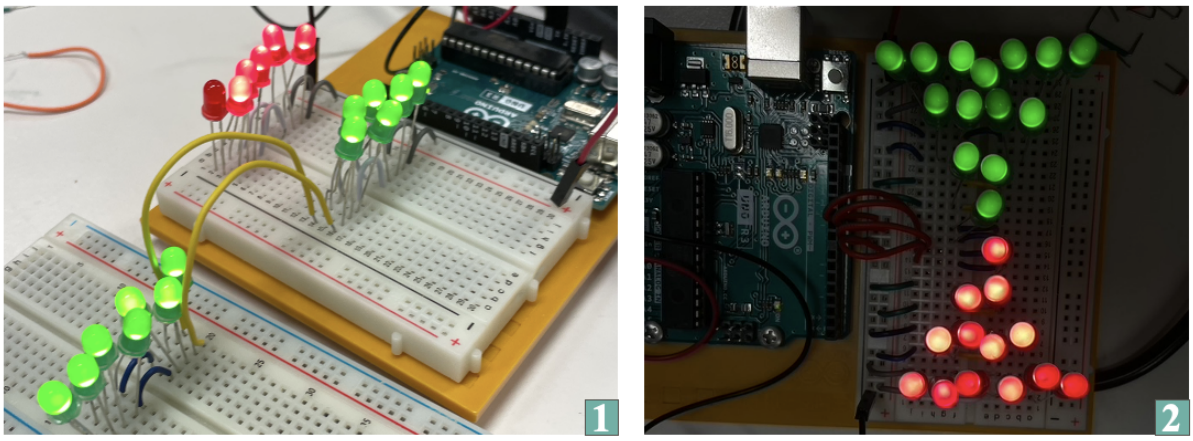
Da vi hadde landet på å bruke LED-lysene som følger med Arduino-settet fikk vi en utfordring med strømforsyning og tilgang på antall digitale porter.

2.2.2. Utforskning



Figur 2: Utforskning 1: seriekobling av to lys, utforskning 2: seriekobling av tre lys

Da vi benyttet oss av seriekobling, så vi at det fungerte helt fint med to lys. Det fungerte ikke med flere enn to lys. En stor ulempe med seriekobling er at når det først er problemer med ett lys, så er det ingen av lysene som lyser.



Figur 3: Utforskning 1: Parallellkobling der 13 av 13 grønne lys lyste, utforskning 2: parallellkobling med alle lysene som skulle brukes i det endelige artefaktet.

Da vi satte opp en parallellkobling, så vi at det fungerte med flere lys enn det vi faktisk skulle bruke i det endelige artefaktet. I seriekobling er strømmen lik i hele kretsen, mens i parallellkobling er spenningen lik i hele kretsen. Dette gjør også at dersom det er problem med et lys i en parallellkobling, lyser fortsatt de andre.

2.2.3. Konklusjon

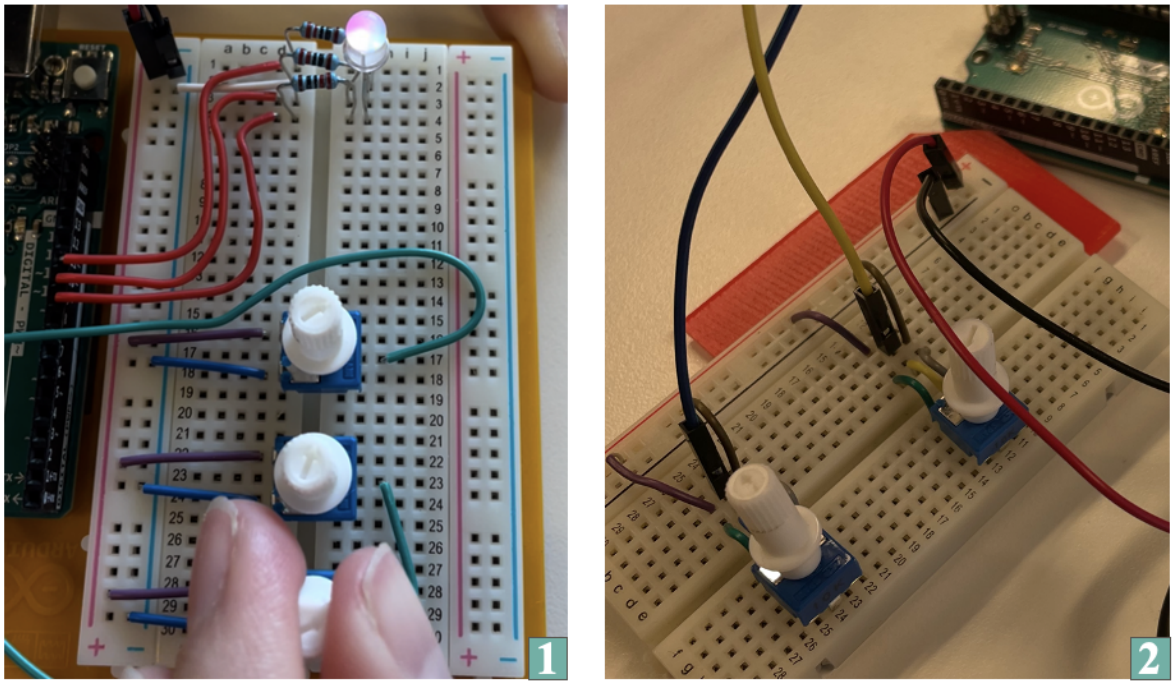
Det var tydelig at det var parallellkobling som var det beste alternativet for vår bruk.

2.3. Potensiometere

2.3.1. Bakgrunn

For å hente input fra brukeren til antall timer søvn og ønsket tid å stå opp måtte vi bruke noen komponenter som gjorde dette. Potensiometerne i Arduinosettet er ustabile og går ikke 365º rundt, og vi ønsket derfor å utforske disse før vi implementerte dem inn i den endelige løsningen.

2.3.2. Utforskning



Figur 4: Utforskning 1: 3 potensiometere for å la brukeren stille inn nøyaktig ønsket farge, utforskning 2: utforskning av klokkepotensiometer og tidspotensiometer til endelig løsning.

Potensiometere ble utforsket ved å knyttet et RGB-lys til tre potensiometere som styrer hver sin farge. I løsningen vår hadde vi tenkt at koden skulle styre fargen på lysene.

Da vi helt konkret utforsket innhenting av informasjon fra potensiometerne prøvde vi flere metoder. Vi eksperimenterte først med forskjellige typer potensiometere, som var tilgjengelige på universitetets lokaler, og hvordan de styrte variablene i arduino-koden.

Vi prøvde også å mappe input-verdien fra potensiometerne, dette gjorde det oversiktelig og greit med verdien til antall timer søvn og verdien til tidspunkt brukeren ønsket å stå opp.

2.3.3. Konklusjon

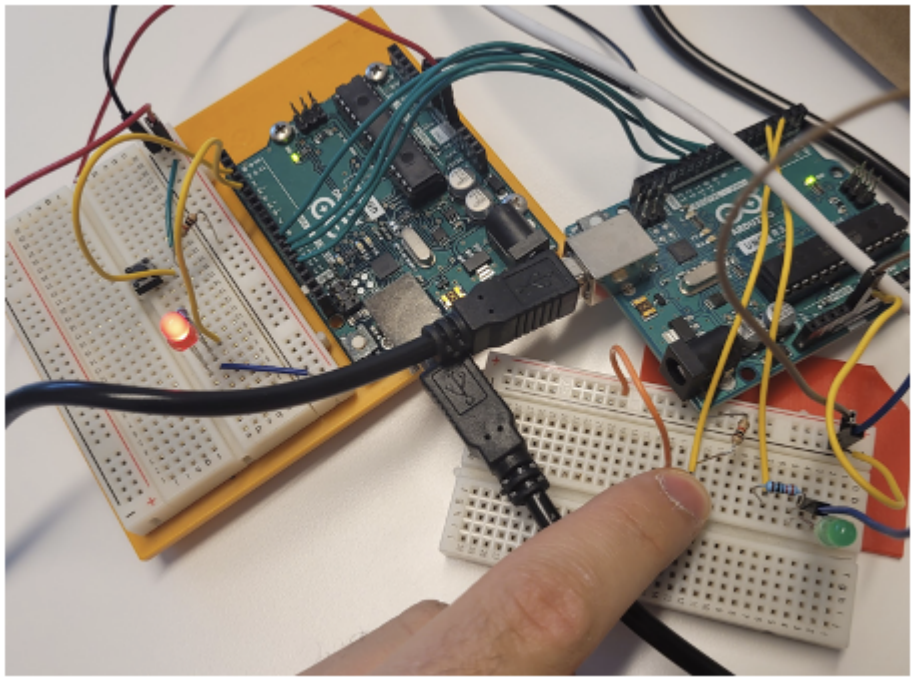
Potensiometerne henter inn en rekke verdier og gjør det enkelt å mappe i ønsket antall deler. Til tross for at potensiometerne både er ustabile og ikke går 365º rundt bestemte vi oss for å bruke de som er i settet.

2.4. Kommunikasjon mellom to Arduinoer: SPI-protokollen

2.4.1. Bakgrunn

Ettersom vi bestemte oss for å bruke LED-lysene som følger med Arduinosettet fikk vi en utfordring når det kom til strømforsyning. Vi trengte nok strøm til at 13 lys kunne lyse samtidig, og hente inn inputverdier fra to potensiometere. På bakgrunn av dette ønsket vi å utforske muligheten med å bruke to arduinoer. Det vi ønsket å oppnå var at én Arduino tok seg av lysene mens den andre skulle skulle ta seg av potensiometerne.

2.4.2. Utforskning



Figur 5: Utforskning av kommunikasjon mellom to Arduinoer.

For å få opprette kommunikasjon mellom to Arduinoer må man benytte seg av SPI-biblioteket. Dette innebærer at en arduino styrer (master) og en er perifer (slave). Å lære seg dette biblioteket bydde på flere utfordringer. En utfordring innebærte inkonsistent og uklar dokumentasjon. For eksempel, blir "ss" (slaveSelect) brukt i den faktiske dokumentasjonen for å velge slave-pin for master-, og ble dermed implementert i testkoden vår, men guiden på Arduinos egne nettsider (Dewey-Hagborg, 2022) opplyser at "cs"(chipSelect) er foretrukket terminologi, men det var uklart at dette ikke var implementert i den faktiske koden. Dette førte til mye forvirring for gruppen, og vi diskuterte hvorvidt det var mulig å basere oss på bare én arduino.

Bruken av to arduinoer i stedet for én ga i tillegg et plassproblem som ble tydelig under denne utforskningen.

2.4.3. Konklusjon

I utgangspunktet fungerte enkle SPI-implementasjoner, der den ene Arduinoen hadde ansvaret for å hente input fra en knapp og LED-lysene som den andre Arduinoen hadde ansvar for lyste

når knappen ble trykket på. Grunnet uklar dokumentasjon av SPI-protokollen og praktiske forhold så vi at det kunne bli utfordrende å implementere ønsket funksjonalitet ved bruk av to Arduinoer. Etter diskusjon i gruppen og nyttig rådføring med faglærer Heidi Bråthen, bestemte vi oss for å gå bort fra to Arduinoer og SPI-protokollen.Vi måtte dermed finne en løsning der det var mulig å kun bruke ett Arduino Uno-brett.

3. Endelig løsning

Den endelige løsningen sentreres rundt to nøkkelfunksjoner:

1. Et timeglass for å visualisere tid som representeres i form av flere rekker med lyskomponenter.
2. To komponenter som kan dreies, og tillater brukeren å styre henholdsvis:
    a. Når brukeren ønsker å stå opp
    b. Hvor mange timer brukeren ønsker å sove

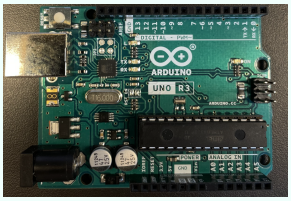
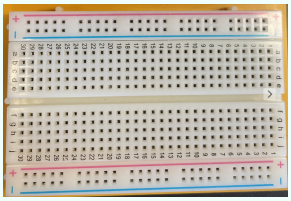
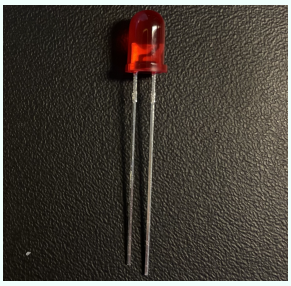
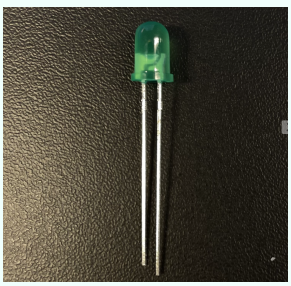
Løsning består av én arduino med to potensiometere som styrer LED-lys av to forskjellige farger i parallellkobling satt opp i en timeglassfasong.

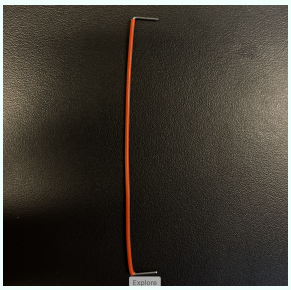
Ideelt sett hadde vi nok trengt flere arduinoer, eller en sterkere strømkilde, da det er tydelig at det er på grensen til overbelastning for arduinoen.

3.1. Link til video

https://youtu.be/AyWqk1BW17Y

3.2. Komponenter

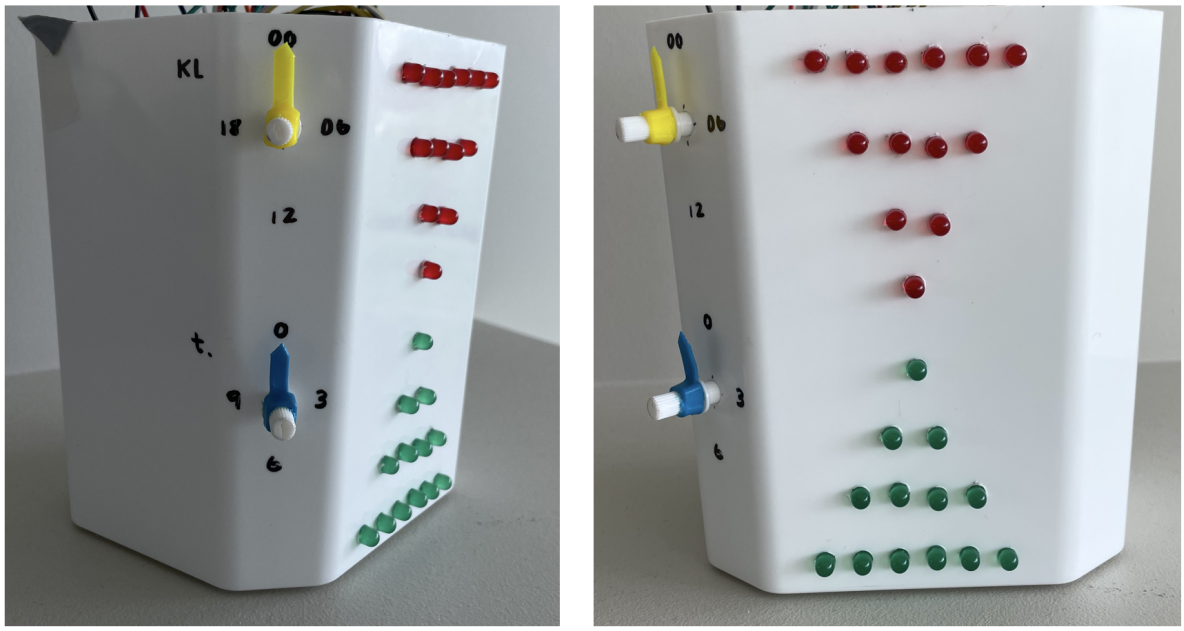
Komponent	Bilde	Beskrivelse og bruksområde	Antall
Arduino Uno		Mikrokontroller. Brukes for å programmere inn funksjonalitetene i artefaktet.	1
Breadboard		Koblingsbrett. Brukes for å koble hele kretsen, som består av dioder og potensiometere, sammen.	2
Rød LED-lys		Diode som lyser rødt når den kobles til strøm. Brukes i øvre halvdel i timeglasset til å varsle at tiden renner ut.	13
Grønne LED-lys		Diode som lyser grønt når den kobles til strøm. Brukes i nedre halvdel i timeglasset til å varsle hvor lang tid som er igjen.	13
Potensiometer		Kan vris 225º rundt. Måler verdier fra 0 til 1023. Brukes som input-parameter, der vi henter inn ønsket antall timer søvn og klokkeslettet bruker vil stå opp.	2

Jumper-ledninger		Brukes for å koble sammen komponenter.	102
Boks	-	Hvit boks i plast. Dimensjoner:	1
Strips	-	Ble brukt til å lage visere og organisere ledningene inni boksen.	2
Tape	-	Ble brukt for å styrke kontaktpunktet mellom komponentene og ble markert med nummer for å holde styr på alle koblinger.	-

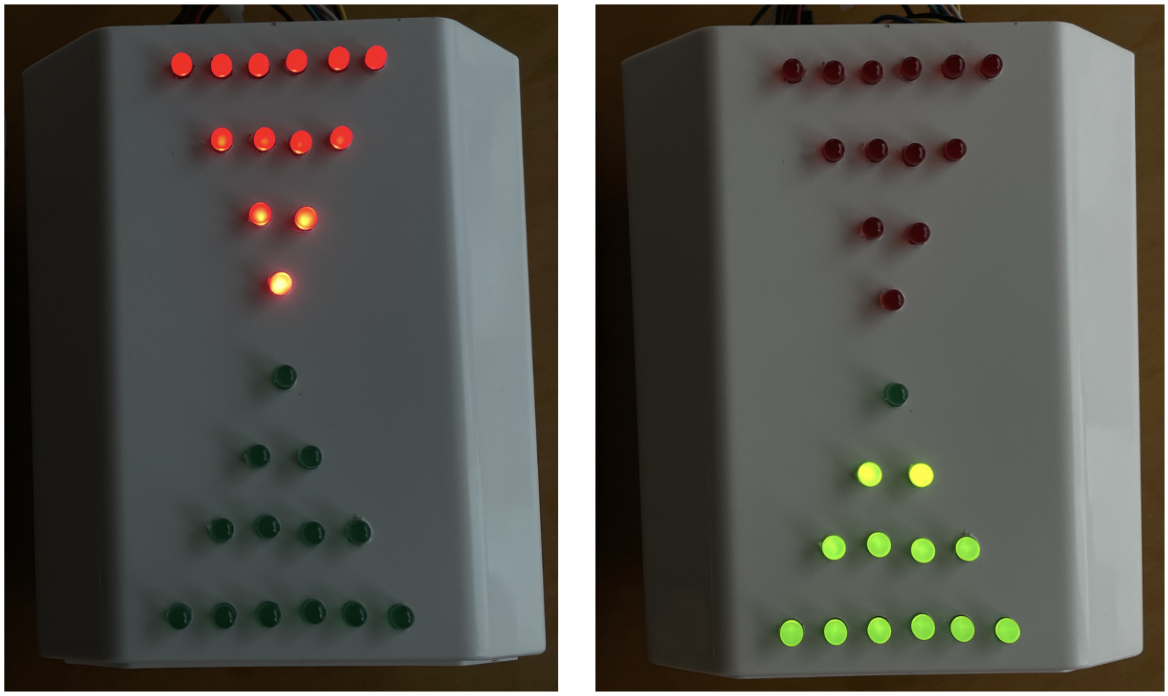
Tabell 3: Tabellen viser en oversikt over komponentene med forklaring på hvordan de brukes i løsningen vår.

3.3. Diagrammer

3.3.1. Bilde av endelig løsning

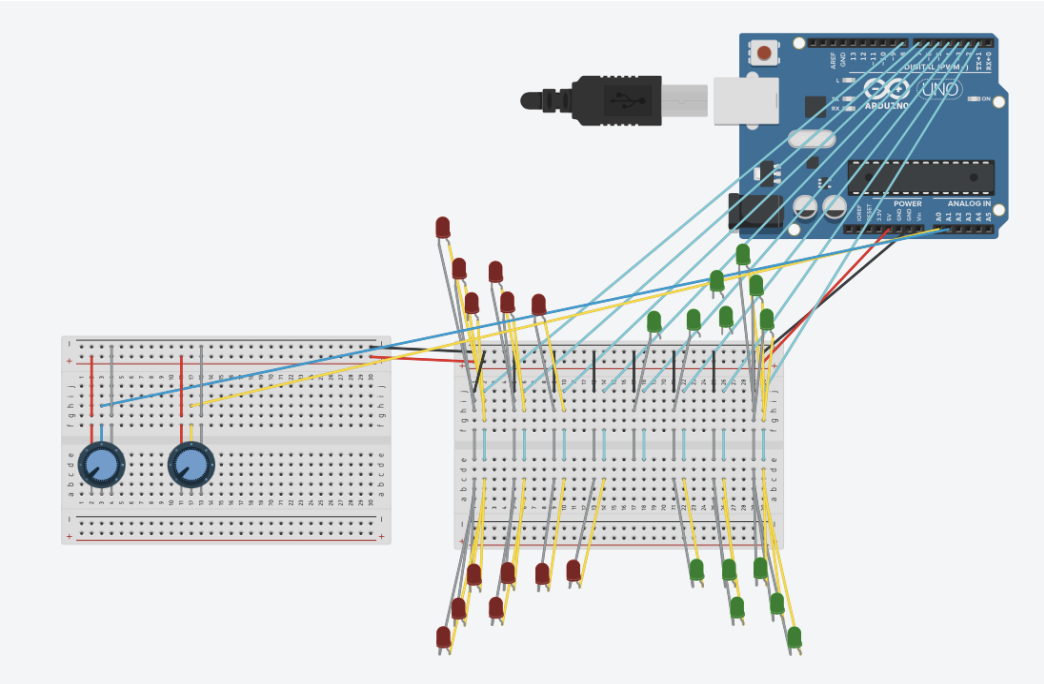


Figur 6: Endelig løsning.

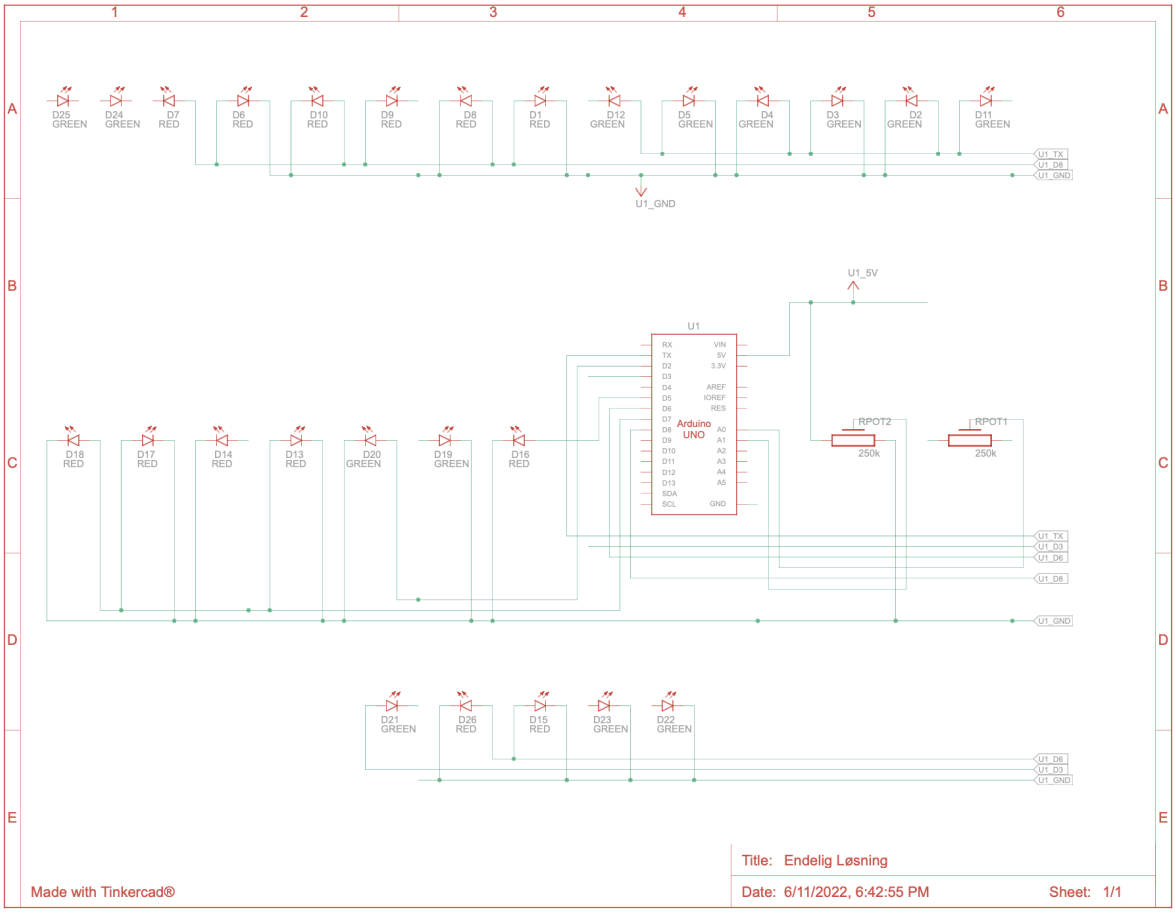


Figur 7: Endelig løsning koblet til strøm.

3.3.2. Kretstegning



Figur 8: Illustrasjon av kretstegning av endelig løsning lagd i Tinkercad.



Figur 9: Kretstegning av endelig løsning laget i TinkerCad.

3.4. Arduino-kode

3.4.1. Kodefil

INO.fil er vedlagt i innleveringen med navnet “CAPY_timglass_arduinokode.ino”.

3.4.2. Forutsetninger

En viktig forutsetning for at løsningen vår skal fungere er at artefaktet til enhver tid vet hva klokken er. Arduinoen har ingen mulighet til å vite hva klokken er, med mindre man bruker en ekstern komponent. Vi bestemte oss for å ikke gå til innkjøp av en klokkemodul. Dette førte til at vi måtte bestemme et tidspunkt som var konstant - dette tidspunktet satte vi til klokken 22:00.

Potensiometerne som følger med i Arduino-settet går ikke 360º rundt, de stopper etter 225º. Vi bestemte oss for å sette en viser på potensiometerne for å gi brukeren feedback om hvilket tidspunkt de setter. Tanken bak dette følger designprinsippet Affordance. Klokken gjør at brukeren enkelt forstår funksjonen. Funksjonen følger formen. Ettersom potensiometere ikke går hele veien rundt blir det ikke helt riktig å sette opp tiden rund som en klokke. Med løsningen vi har implemetert vil klokken forskyves litt hele veien.

1.1.1. Instansvariabler

Variabelnavn	Forklaring
potStaaOpp	Henter input fra potensiometer som er koblet til analoginputport A0.
potValStaaOpp	24-timersklokke og verdien potStaaOpp er derfor mappet i 24 deler. Potensiometerne vi har brukt har 1023 verdier, og ideelt sett hadde vi brukt potensiometer som gikk 365° rundt og dermed hadde flere verdier (samme gjelder for potValTimerSovn).
potTimerSovn	Henter input fra potensiometer som er koblet til analoginputport A1.
potValTimerSovn	Vi har satt maks 12 timer søvn, og verdien potTimerSovn er derfor mappet i 12 deler.
antTimerSovn	Lagrer input-verdiene bruker sender inn ved å skru på tilhørende potensiometer.
tidStaaOpp	Lagrer input-verdiene bruker sender inn ved å skru på tilhørende potensiometer.
tidLeggeSeg	Regner ut når brukeren må legge seg avhengig av nåværende klokkeslett (antagelse i koden vår), antTimerSovn og tidStaaOpp. Formelen er som følger: tidStaaOpp - antTimerSovn. Alle de mulige verdiene variabelen kan ha er illustrert i figur 10. Ved ugyldige verdier settes tidsIntervall til 0.
tidsInterval	Regner ut hvor lang tid det er til brukeren må legge seg basert på nåværende klokkeslett og tiden til tidLeggeSeg, se punkt 3.4.4. for formler.
antLysrekker	Konstant. Vi har åtte lysrekker der de øverste fire består av røde LED-lys og de fire nederste radene består av grønne LED-lys.

Tabell 4: Oversikt over variablene i koden vår og hva de gjør.

timer søvn	0	1	2	3	4	5	6	7	8	9	10	11	12
klokkeslett													
0	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12
1	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11
2	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
3	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9
4	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8
5	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7
6	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6
7	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5
8	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4
9	9	8	7	6	5	4	3	2	1	0	-1	-2	-3
10	10	9	8	7	6	5	4	3	2	1	0	-1	-2
11	11	10	9	8	7	6	5	4	3	2	1	0	-1
12	12	11	10	9	8	7	6	5	4	3	2	1	0
13	13	12	11	10	9	8	7	6	5	4	3	2	1
14	14	13	12	11	10	9	8	7	6	5	4	3	2
15	15	14	13	12	11	10	9	8	7	6	5	4	3
16	16	15	14	13	12	11	10	9	8	7	6	5	4
17	17	16	15	14	13	12	11	10	9	8	7	6	5
18	18	17	16	15	14	13	12	11	10	9	8	7	6
19	19	18	17	16	15	14	13	12	11	10	9	8	7
20	20	19	18	17	16	15	14	13	12	11	10	9	8
21	21	20	19	18	17	16	15	14	13	12	11	10	9
22	22	21	20	19	18	17	16	15	14	13	12	11	10
23	23	22	21	20	19	18	17	16	15	14	13	12	11
24	24	23	22	21	20	19	18	17	16	15	14	13	12

Figur 10: Oversikt over mulige verdier tidLeggeSeg kan ha, verdiene markert i rødt er ugyldige verdier.

1.1.2. tidsInterval

I utgangspunktet ønsket vi når én grønn lysrekke slo seg av slo også én rød lysrekke seg simultant på. Dette hadde derimot begrenset antall intervaller til kun fire. Vi valgte dermed å dele intervallene i åtte (antall lysrekker) for å spre tiden litt mer ut og gi brukeren feedback litt oftere. Intervallene har vi satt inn i enkel array.

Formelen for tidLeggeSeg er:

tidLeggeSeg = tidStaaOpp - antTimerSovn

Formelen for antTimerIgjen kommer an på leggetiden. For tidLeggeSeg fra og med kl 22 til og med kl 24 er formelen:

antTimerIgjen = tidLeggeSeg - 22.

For tidLeggeSeg fra og med kl 01 til og med kl 21 er formelen:

antTimerIgjen = tidLeggeSeg + 2

Formelen for tidsInterval blir dermed antall timer igjen multiplisert med 3 600 000 millisekunder, som tilsvarer én time.

tidsInterval = antTimerIgjen * 3 600 000

Når tidsInterval er lik 0 lyser alle de rød lysene, og de grønne er avskrudd.

1.1.3. Metoder

loop()

Kode som repeteres kontinuerlig. Her kalles hentInput() i starten av hver løkke. For så å regne ut tiden og begynne å telle ned.

millis()

Vi valgte å bruke millis() for å holde oversikt over tiden til tross for at det ikke er det mest ideelle å regne timer i millisekunder. Årsaken til dette var at dette var noe vi var kjent med, vi kunne kontrollere det og det gjorde testing av funksjonalitet enkelt underveis. Overflow skjer ikke før etter 50 dager (Arduino-dokumentasjon Arduino.cc), og vi så dermed ikke noe grunn til å ikke benytte oss av denne metoden.

hentInput()

hentInput()-metoden tar inn potensiometerverdier fra brukeren og disse verdiene brukes videre for å styre lyshandlingene.

2. Referanseliste

Dewey-Hagborg, H. (2022, 23. mai). Introduction to the Serial Peripheral Interface. Hentet 31.05.2022 fra https://docs.arduino.cc/tutorials/generic/introduction-to-the-serial-peripheral-interface/

Arduino (2022, 8. mai) Hentet 9. juni 2022 fra https://www.arduino.cc/reference/en/language/functions/time/millis/

Fitzgerald, S. & Shiloh, M. (2012). *The arduino projects book*. Arduino LLC.

Alle bilder er tatt av oss, og alle figurer er laget av oss.