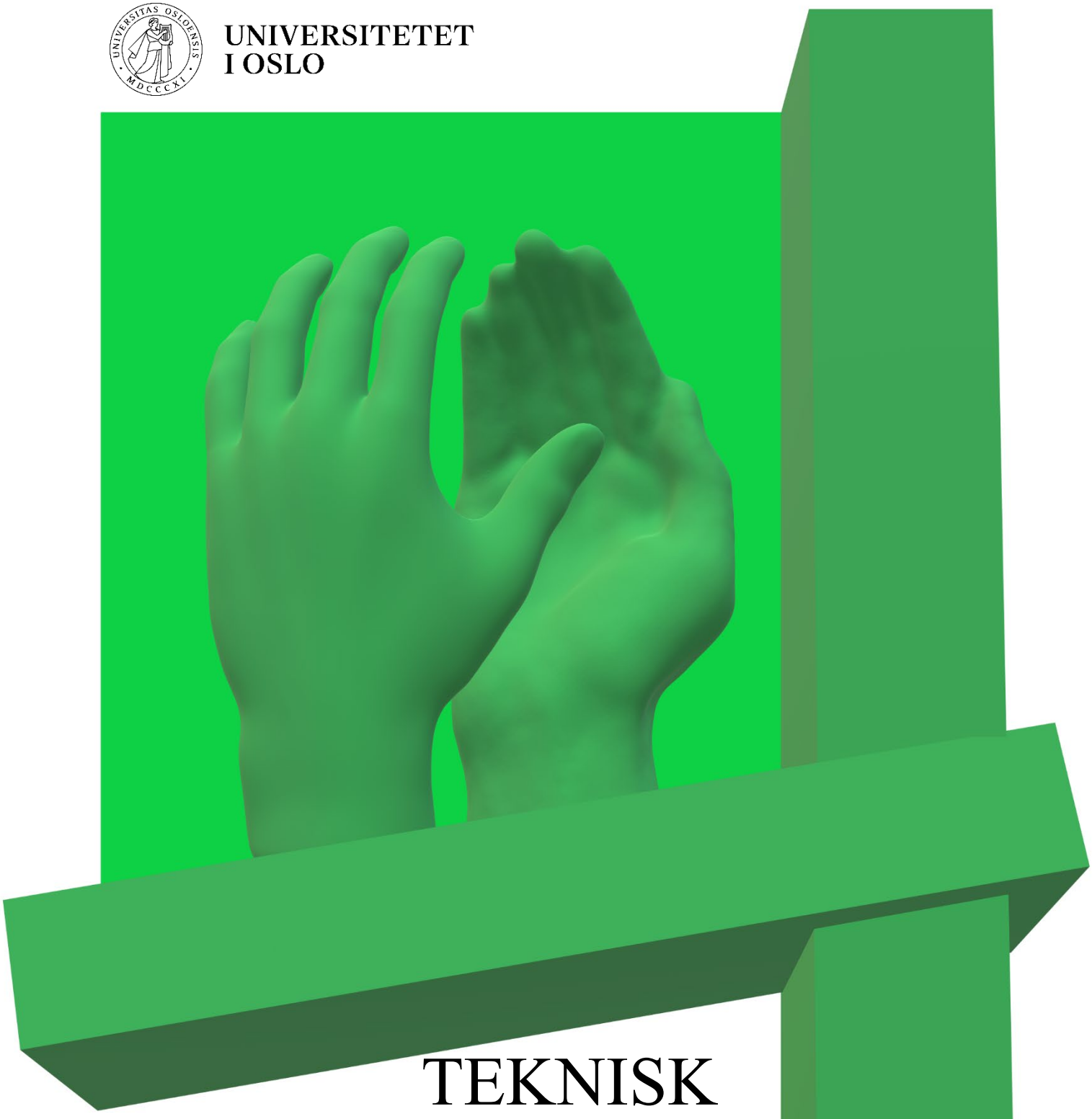
UNIVERSITETET
I OSLO



TEKNISK
RAPPORT

B-GJENGEN – IN1060 – V2023

August Brøyn, Tellef Berg Aasen, Kaja Mamelund Bradal, Helen Bugge og Lars Bjelland Brekke

Innholdsfortegnelse

Antall ord: 3927

I Innledning

I.I Om oss

August: Bakgrunn innen koding og kunst. Opptatt av sosial bærekraft. Hobby: Musikk

Tellef: Bakgrunn innen arkitektur og formidling. Hobby: Smiing

Kaja: Bakgrunn innen helse og arbeid med eldre og personer med funksjonsnedsettelser. Hobby: Strikking og TV-serier

Helen: Bakgrunn innen helseservice- og oppvekst. Glad i natur, reising, og ernæring. Hobby: Maling og trening.

Lars: Bakgrunn innen ikt og salg/service. Er utadvendt og trives med å snakke med mennesker. Hobby: Lego Technic

I.II Introduksjon

Formålet med denne tekniske rapporten er å dokumentere og presentere prototyper og løsningsforslag ved å demonstrere at vi har laget en løsning som virker, hvordan den virker, hvordan løsningen virker i bruk og hvor teknisk avansert løsningen er. Vi viser også til videoen gruppen har laget for å demonstrere disse punktene gjennom to scenarier.

Fra prosjektoppgavene valgte gruppa oppgaven, som vi har gjennomgående navngitt «å sanse det usanselige», som omhandler sansene. Vi har jobbet med døvblinde og menneskene rundt dem i vårt prosjekt. Prototypingen av funksjon, form og implementasjon skjedde gjennom en iterativ prosess. Prosessen inkluderte jevnlige brukermøter med døvblinde og ansatte på Conrad Svendsen Senter på Nordstrand og vi tok utgangspunkt i designtilnærmingen DMB (design med brukere, el. design for, med og av brukere). I B-gjengens grupperapport skriver vi mer om hvordan vi har kommet fram til målgruppe, konsept, formkonsept, krav og mye av det vi snakker om i denne tekniske rapporten. Vi beskriver i denne rapporten artefaktet vi har endt opp med ved slutten av prosjektet og dokumenterer innledende prototyper samt funksjon, brukbarhet og tekniske implementasjoner ved prototypene.

I prosjektoppgaven står det:

- Arduino skal være kjernen i den tekniske løsningen. Kommentar: en tommelfinger-regel er at løsningen ikke kunne ha vært en app – da er ikke Arduino utnyttet godt i designprosessen.

Vi har oppfylt kravspesifikasjoner, som ble formet i designprosessen, gjennom verktøyet Arduino og komponenter laget for det. Fra ordlyden av oppgaven vi valgte, var vi bestemte på å bruke sensorkomponenter til Arduino i prototypingen. Arduino er en liten datamaskin som er laget for kunst- og designstudenter, og som egner seg spesielt godt til prototyping. Vi har utviklingen av den tekniske løsningen dratt nytte av boken *The Arduino Projects Book* (Fitzgerald & Shiloh, 2012) og tilgjengelig materiale på Arduinos hjemmeside (Arduino Education, 2023).

I.III Visjon, konsept og formkonsept

Visjonen vår er *en kommunikasjonskanal, designet for døvblinde, som er hjelpsom for så mange folk med ulike forutsetninger som mulig.*
Konseptet er *kommunikasjon,* som kom fram gjennom fellestrekkene i de forskjellige behovene hos brukerne.
Formkonseptet er *hånden;* kontaktpunktet mellom døvblinde og omverdenen, i de fleste situasjoner.

I.IV Mål

Gjennom designiterasjoner og møter med brukerne kom det fram tre hovedtemaer, hva gjelder behov: Det sosiale, kontakt og informasjon. Fra disse kunne vi generalisere et konsept og en visjon. Samtlige oppdagede behov hos brukerne gjaldt kommunikasjon. Vi tror ikke det vil overraske mange, i lys av tilstanden til døvblinde. Mennesker med hørsel- og synsnedsettelse er stort sett ganske like alle andre og deler ofte mange av deres behov. Men noe av poenget med å jobbe med disse menneskene er å skape noe som hjelper med det som er spesifikt for deres tilstand. Dette står kontekstuelt med hele vårt prosjekt og målet med vår prototyping.

Fra behovene utviklet vi bestemte funksjonelle krav som bl.a. innebar følgende; vi skulle lage et system som kunne hjelpe i utfordringene rundt kommunikasjon i sosiale settinger, som fordret og støttet opp under god kontaktsøking blant målgruppa og som var et verktøy som gjorde nyttig informasjon tilgjengelig på en klar og konsis måte.

Løsningen vi gjennom iterasjonene kom fram til er et kart over rom i huset brukeren befinner seg i, der rommene er taktilt representert og signaliserer aktivitet ved vibrasjon. Dette oppnås ved hjelp av bevegelsessensorer plassert i rommene. Dette er kjernen i løsningen, men den har mange mulige implementasjoner.

I.V Video

Link til video: https://youtu.be/XF1_F_4-bI4

I videoen presenterer vi hvordan den tiltenkte bruken av artefaktet vil være. Dette gjøres gjennom to problemscenarioer med tilhørende løsninger. Alle i gruppa tok del som aktører i videoen. Den er delt i to deler. Den første delen er scenariene med løsning, her benytter vi oss av Wizard of Oz for å få situasjonen så lik tiltenkt bruk som mulig. I den andre delen viser vi hvordan løsningen faktisk fungerer slik prototypen er i dag. I denne delen benytter vi oss ikke av noen hjelpemidler, eller teknikker for å vise artefaktet i bruk, det vises akkurat slik det fungerer i praksis.

I det første scenariet viser vi hvordan noen ansatte og beboere driver med en aktivitet i et fellesrom på senteret et tiltenkt bo- og behandlingssenter. I et annet rom sitter det en beboer som ikke får med seg at det er en aktivitet i fellesrommet, siden brukeren er døvblind. I dette scenariet viser vi hvordan artefaktet vårt hjelper den døvblinde med å få med seg aktiviteten. En bevegelsessensor registrerer aktiviteten i fellesrommet og varsler brukeren på armbåndet hans. Han kan deretter føle på artefaktet hvor på senteret det er en aktivitet ved at bokstaven til rommet med aktivitet vibrerer. Dette er slik den siste iterasjonen av vår prototype fungerer.

I det andre scenariet utspiller det seg en situasjon på det tiltenkte bo- og behandlingsenteret der en ansatt ønsker å starte en aktivitet i et fellesrom. Den ansatte må selv gå til hver enkelt beboer og spørre om personen ønsker å bli med på aktiviteten eller ikke. I dette scenariet viser vi hvordan artefaktet ville løst dette ved at den ansatte kun trenger å trykke på en knapp på bevegelsessensoren i fellesrommet. Knappen varsler beboerne via artefaktet om at det er en aktivitet som skal starte, eller er i gang i fellesrommet. På denne måten kan beboeren selv velge om de ønsker å delta eller ikke, og de ansatte sparer tid og ressurser ved at de slipper å gå inn til hver enkelt beboer. Gjennom intervjuer og observasjon fant vi ut at dette kunne være ubehagelig for mange beboere, og at de aldri blir helt vant til at noen kommer inn og tar på de for å få kontakt. Dette scenariet er basert på innspill fra brukerne våre, men er ikke noe vi har fått implementert i prototypen. Holderen til bevegelsessensoren har knapp og hull til knapp, slik at det kan implementeres, men dette var dessverre ikke noe vi hadde tid til å prioritere i denne iterasjonen.

I den tekniske visningen av artefaktet viser vi hvordan prototypen ser ut og fungerer etter siste iterasjon. Selve planløsningen huser Arduino og breadboard, samt vibrasjonsmoduler til de to fungerende rommene. Ut av planløsningen går det ledninger til henholdsvis PC (for strøm), bevegelsessensorer, vibrasjonsmodul og knapp til armbåndet. Videoen viser hvordan aktivitet fra personen som filmer blir registrert og varsler dette via armbåndet. Når personen som filmer trykker på knappen på armbåndet begynner bokstaven til den tilhørende bevegelsessensoren å vibrere. Denne delen av videoen er med original lyd for å vise at artefaktet fungerer som beskrevet.

II Artefaktet «Vibber»

II.I Etablering av krav

Vi har samlet data, analysert dataen, utviklet krav og prototypet gjennom flere iterasjoner i prosjektet. Mer spesifikt itererte vi gjennom akkurat disse aktivitetene tre ganger. Designteamet, med de inviterte brukerne, har ved hver iterasjon stått for utvikling av nye krav til prototypingen. Til å begynne med handlet det mer om generelle funksjonelle krav og generelle rolleprototyper. Etter hvert som vi itererte og prototypet mer høyoppløselig, gikk vi mer inn i detalj. Under knytter vi samtlige etablerte krav til de stadiene der de først ble etablert. Vi vil også påpeke viktigheten av form i artefaktet. Noen av de viktigste kravene gjaldt nettopp form, da mennesker med hørsel- og synshemning har behov for forståelige taktile løsninger. Dette innebærer at vi fra første evaluering av de første ideene (rolleprototypene) så oss nødt til å prototype størrelser innen form. Formprototyping var altså et område vi hadde mye fokus på. Vi har vektlagt kravene (1, 2 og øverste vekt 3) og forsøkt å oppfylle samtlige.

II.II Spesifikasjon av krav

Beskrivelse	**Kommentar**
Arduino – Kjernen av artefaktet skal være Arduino.	Oppgaven påpeker at Arduino skal stå for funksjonene i løsningen.
Sensorer – Et ikke-funksjonelt krav: Artefaktet skal bruke sensorer for å innfri minst én funksjon.	Løsningen vi endte med bruker PIR-bevegelsessensorer.
Universell utforming – Artefaktet skal være universelt utformet etter heuristikk og retningslinjer. Ofte dreier det seg om formfaktorer når det gjelder døvblindhet.	Form: Størrelse, avstander, kontraster etc. finnes det ofte etablerte standarder av i feltet.
Oppnå selvstendighet – artefaktet skal gi den døvblinde bruker en økt selvstendighet i hverdagen.	Dette er etablert fra behov hos både døvblinde brukere og de ansatte brukerne.
Temaer: sosialt, kontakt og informasjon – artefaktet skal være et verktøy innen én eller flere av disse områdene.	Dette er behovsområdene som ble oppdaget i første iterasjon.

Varsler aktivitet – Artefaktet skal varsle brukeren at det er aktivitet i bygget brukeren befinner seg i.	Vi gikk for planløsning-idéen og dette er en av hovedfunksjonene.
Forskjellige rom – Artefaktet skal kunne vise til aktivitet i forskjellige deler av bygget.	Brukeren skal også kunne identifisere hvor det er aktivitet.
Manuell varsling – Brukere skal kunne manuelt varsle en aktivitet.	Dette fikk vi ikke implementert det tekniske rundt.
Plantegning – Artefaktet skal hovedsakelig bestå av en plantegning som beskriver en tenkt «layout» i en bygning brukeren befinner seg i.	Kunne også vært implementert med en liste over rom i bygget. Vi ville gå for plantegning da dette kom fram som mer intuitivt.
Vibrasjon – Artefaktet skal kommunisere aktivitet i andre rom ved hjelp av vibrasjon.	Andre implementasjoner ble utforsket, bl.a. avstander og til og med luktdispensere.
Bevegelsessensorer – Artefaktet skal bruke bevegelsessensorer for å merke aktivitet i forskjellige rom.	Andre sensorer er også mulig og de kan kombineres for å nå en satt aktivitetsterskel.
Knapp på bevegelsessensoren – Bevegelsessensorene skal ha knapper på seg som fungerer som manuell varsling.	Man kan også implementere automatisk varsling knyttet opp til en kalender med aktiviteter.
Klokke – Artefaktet skal fungere med en klokke/armbånd på håndleddet.	Varsling på kroppen er én av mange mulige implementasjoner.
Håndsansing – Artefaktet skal kunne være forståelig taktilt kun ved å føle med hånden.	Artefaktet skal være brukbart for fullstendig døvblinde.
Tilpasset hånden – Størrelse på plantegningen skal være tilpasset hånden (som er formkonseptet).	Hvis plantegningen er for stort kan det være vanskelig å få et klart bilde. Hvis det er for lite kan for mye informasjon ende opp for tett.
Tilpasset plassering av artefaktet – Vinkelen på brettet skal kunne endres	Gjennom evaluering fant vi ut at artefaktet gjerne kunne stå skrått for å lette interaksjonen.

Vegger – Veggene i planløsningen skal ikke være for høye.	Gjennom evalueringen av integrasjonsprototypen kom det fram at veggene kunne senkes.
Punktskrift – Artefaktet skal være universelt utformet også ved at rommene er markert med punktskrift.	«Stue», «Fellesrom», «Kjøkken», etc. i punktskrift.

III Teknisk løsning

Når vi skulle lage den tekniske løsningen måtte vi utforske, ta valg og inngå kompromiss. Vi vil nå si litt om denne prosessen.

III.I Bevegelsessensorer

Vi valgte å bruke kun to bevegelsessensorer i vår tekniske løsning, selv om artefaktet var laget for tre rom. Dette var fordi vi mente det holdt med to for å vise funksjonaliteten av at forskjellige rom skulle kunne vibrere uavhengig. Sensorene var ikke billige, som også var noe av grunnen.

Når vi skulle implementere de to sensorene, måtte vi først utforske hvordan de fungerte. Da vi først skrev koden for bevegelsessensorene, antok vi først at de fungerte som følger:

- At det ble sendt et kort HIGH-signal hver gang den merket aktivitet. Med andre ord trodde vi den kunne sende titalls individuelle HIGH-signaler i løpet av få sekunder.
- At den hadde en justerbar cooldown (som ble beskrevet på nettsiden til distributøren) som gjorde at den ikke sendte signaler i en gitt tid etter den sendte.
- At man kunne justere sensitivitet (som også ble beskrevet på nettsiden) som gjorde at det krevde mer (kraftigere) bevegelse for at den skulle sende HIGH-signal.

Utforskingen av bevegelsessensorene gikk hånd i hånd med utforskingen av koden som fikk de til å fungere, da man ikke kunne teste de uten kode (det var også mangelfull informasjon på distributørens nettsider). Det betydde at vi måtte ha et utgangspunkt. Utgangspunktet vårt i koden speilet våre antagelser om sensorene, som alle viste seg å være feil.

Vi hadde først tenkt at det skulle være x antall utløsninger (el. triggere) i sensorene innen en gitt tid. Dette var logisk vanskelig å se for seg hvordan skulle kodes. Vi kunne ikke ha en enkel variabel som

økte hver gang den sensoren ble utløst og deretter sette denne tilbake til 0 når en tid var gått. Dette er fordi det da kunne vært kort tid siden siste utløsning skjedde, uten å telle sammen med de utløsningene som hadde skjedd etter variabelen ble satt til 0.

Vi skjønte at hver utløsning måtte være lagret i den bestemte tiden, uavhengig. Vi løste dette ved å lage objekter for hver utløsning, som hadde en levetid lik tidsintervallet:

```
class Trigger {
  public:
    unsigned long startTime;
    bool erAktiv;

    Trigger() {
      erAktiv = false;
    }

    void aktiver() {
      erAktiv = true;
      startTid = millis();
    }

    void oppdater() {
      if (erAktiv && (millis() - startTid) > AKTIVITET_INTERVALL) {
        erAktiv = false;
      }
    }

    bool aktiv() {
      return erAktiv;
    }
};
```

```
// Knapp trykkes
if (lastsensorStatus == LOW && sensorStatus == HIGH) {
  // Legger til ny trigger-objekt
  for (int i = 0; i < MAX_triggere; i++) {
    if (!triggere[i].aktiv()) {
      triggere[i].aktiver();
      break;
    }
  }
}
```

```
// Oppdaterer alle triggere og teller de aktive
int aktivtriggere = 0;
for (int i = 0; i < MAX_triggere; i++) {
  triggere[i].oppdater();
  if (triggere[i].aktiv()) {
    aktivtriggere++;
  }
}
```

```
// EKSEMPEL PÅ SJEKK FOR AKTIVE TRIGGER-OBJEKTER
// Hvis det er 5 aktive triggere og det har gått
// SERVO_BEVEGELSE_INTERVALL ms siden siste bevegelse
if (aktivtriggere >= 5) {
  if (!klokkeVibrerer){
    klokkeStartTid = millis();
  }
  if (klokkeTimer < KLOKKE_VIBRASJON_TID){
    //Vibrasjonskode
  }

  klokkeTimer = millis() - klokkeStartTid;
}
```

Figur 1: Bruk av triggerobjekt i tidligere kode

Som sagt var antagelsene våre feil, så denne koden var ikke brukbar. Slik skjønte vi gjennom utforsking av kode at sensorene faktisk fungerte:

- Sensorene sendte et HIGH-signal i en bestemt cooldown-tid (som kunne justeres med et fysisk hjul på sensorene).
- De sendte altså alltid HIGH-signal i en bestemt tid etter den hadde blitt utløst kun én gang.
- Sensitiviteten kunne justeres (også med et fysisk hjul), men dette gjaldt avstanden sensorene kunne plukke opp bevegelse (maks 7 m).

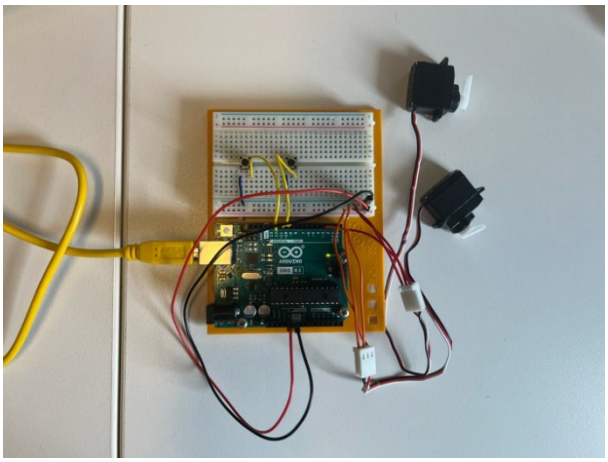
Etter denne utforskingen kastet vi idéen om utløsningsobjekter. Vi valgte heller å kode at om sensorene hadde sendt aktivitet i en bestemt tid, så ville det respektive rommet bli aktivt. Dette fungerte bra etter litt justering på tid-konstantene. Vi viser til den publiserte koden, samt beskrivelsen av koden videre i den tekniske rapporten, for endelig løsning av koden rundt sensorene.

III.II Vibrasjon

Vi fant tidlig ut at vibrasjon var noe som allerede ble brukt som varsling for både døve og døvblinde, og det ble naturlig å velge dette også i vår prototype. Helt i starten så vi for oss å bruke piezo-

elementer for å få vibrasjon i de forskjellige komponentene. Da vi fant ut at piezo-elementet ikke vibrerte i den forstand vi ønsket, gikk vi inn for å bestille vibrasjonsmoduler.

I forkant av evalueringen av den høyoppløselige prototypen hadde vi enda ikke fått vibrasjonsmodulene, og vi måtte finne et alternativ. Vi kom frem til at vi kunne skape en liten vibrasjon ved å bruke servomotorer, og vi valgte å implementere dette i den første iterasjonen av den høyoppløselige prototypen. Servoene satt montert under bokstavene på plantegningen, og roterte 5 grader frem og tilbake i høy hastighet. Servoen var i kontakt med bokstaven ved at vi festet en av de medfølgende armene i en vinkel som tillot dette. Denne løsningen gjorde at man kunne tyde at bokstaven beveget seg hvis man følte den med fingeren. Siden vi ikke ønsket å lage et armbånd som var stort og klumpete, lot vi servoen som representerte vibrasjonen i armbåndet ligge løst utenfor brettet. Dette fungerte godt nok for å vise den tiltenkte bruken, men var ikke optimalt. Koden vi brukte for å få disse til å fungere var heller ikke optimalt da vi måtte bruke delay(). Dette er en funksjon vi bruker for at servoene skulle få flytte seg før de fikk en ny instruks. Problemet er at funksjonen gjør at hele programmet venter denne tiden, som er lite ønskelig i et system som skal kunne interageres med.



```
klokkeServo.write(90);
delay(SERVO_BEVEGELSE_INTERVALL);
klokkeServo.write(85);
delay(SERVO_BEVEGELSE_INTERVALL);
```

Figur 2: Implementering av servoer, og koden som hørte til.

Etter evalueringen, i den siste prototypen fikk vi endelig implementert vibrasjonsmodulene. Disse var mye mindre enn servoene og ga oss mer plass å jobbe med. De hadde dobbeltsidig teip på seg, som gjorde det veldig lett å feste de på undersiden av bokstavene og innsiden av armbåndet. Ledningene var korte, så de måtte loddes til lengre ledninger for å få de koblet opp med Arduinoen. Med de nye modulene på plass var vibrasjonen tydelig, kanskje litt for tydelig. Vi kunne valgt å teste hvordan vi kunne variert vibrasjonen i koden, ved å føre mer eller mindre strøm til modulene, men vi valgte å ikke prioritere dette da de allerede fungerte for å vise den tiltenkte bruken av artefaktet. I vår implementasjon fungerer modulene slik:

- De blir aktivert ved at Arduinoen sender ut et HIGH signal i en gitt periode (ulik varighet på armbåndet og bokstavene), før den deretter sender LOW i den tiden vi ikke ønsker vibrasjon.

III.III Kode

Link til kode i GitHub: https://github.com/Gustus12/IN1060/blob/main/gpFerdig.ino

Under lister vi opp forskjellige konstanter og variabler. Standard er at man skriver konstanter i store bokstaver og med «underscore» som mellomrom. Dette for å skille dem fra variabler i koden. Forskjellen på konstanter og variabler er at konstanter ikke skal endre seg i løpet av koden.

Liste over konstanter

Konstanter	Beskrivelse
SENSOR1_PIN & SENSOR2_PIN	Pin-nummer for sensorene (flere kan legges til for implementasjon av flere rom)
KLOKKE_KNAPP_PIN	Pin-nummer for knappen i klokka
KLOKKE_VIBRASJON_PIN	Pin-nummer for vibrasjonselement i klokka
ROM1_VIBRASJON_PIN & ROM2_VIBRASJON_PIN	Pin-nummer for vibrasjonselementer i rommene (flere kan legges til for implementasjon av flere rom)

Konstanter	Beskrivelse
KLOKKE_VIBRASJON_TID	Hvor lenge klokken skal vibrere når noe skjer i systemet (i ms.)
ROM_VIBRASJON_TID	Hvor lenge vibrasjonselementene for rommene skal vibrere (i ms.)
ROM_AKTIVITET_TID	Hvor lenge et rom holdes aktivt (i ms.)
KLOKKE_COOLDOWN	Hvor lang tid som må gå for at klokken får lov til å vibrere igjen (i ms.)
SENSOR_TRESHOLD	Treshold (terskel) for at en sensor gjør et rom aktivt (i ms.)

KLOKKE_COOLDOWN

Klokka skal sende et slags push-varsel om noe skjer i systemet. Denne konstanten sørger for at det ikke sendes pushvarsel så ofte. Når en pushvarsel er sendt og en bruker velger å ignorere, er det nyttig at den ikke sender pushvarsel hele tiden.

For eksempel kan det tenkes at fler rom legges til og det vil da kunne hende at et rom blir deaktivert og aktivert mye hyppigere. Vi har gjennomgående kodet med tanke på mulig utvidelse.

Under testing har vi hatt denne konstanten på 60 sekunder (60000 ms.).

SENSOR_TRESHOLD

Denne konstanten avgjør om aktiviteten (bevegelsen som blir tatt opp av sensorene) er adekvat for å avgjøre at et rom er aktivt. Altså vil et rom gjøres aktivt om sensoren har sendt HIGH-signal i SENSOR_TRESHOLD ms.

Under testing har vi hatt denne konstanten på mellom 10 og 20 sekunder (10000 ms. – 20000 ms.).

Liste over variabler

Variabler	**Beskrivelse**
bool klokkeVibrerer	Blir satt til false ved initiering av variabelen. Gjøres true når klokken skal vibrere. Brukes til å sjekke om klokken vibrerer.
unsigned long klokkeStartTid	Blir satt til 0 ved initiering av variabelen. Settes til millis() når klokken starter å vibrere.
unsigned long klokkeTimer	Blir satt til 0 ved initiering av variabelen. Settes til millis() – klokkeStartTid ved hver gjennomkjøring av loopen for å ta tiden.
int sensor1Status & int sensor2Status	Holder en integer (et heltall) som i praksis bare er 1 eller 0. Dette kan også leses og endres med HIGH og LOW.
int forrigeSensor1Status & int forrigeSensor2Status	Settes til sensor1Status eller sensor2Status henholdsvis etter at all bruk av variablen er gjort.
unsigned long sensor1timer & unsigned long sensor2Status	Settes til millis() – sensor1timer (el. sensor2timer) ved hver gjennomkjøring av loopen for å ta tiden sensoren har vært aktiv.
unsigned long sensor1StartTid & unsigned long sensor2StartTid	Settes til millis() når sensoren starter å plukke opp bevegelse (når sensoren først sender HIGH-signal)
bool rom1Vibrerer & bool rom2Vibrerer	Settes til true når rommene vibrerer. Brukes i koden for å sjekke om rommet vibrerer (for å ta tiden).

bool rom1ErAktivt & bool rom2ErAktivt	Settes til true når et rom er aktivt. Brukes i koden for å sjekke om et rom er aktivt (for å ta tiden).
unsigned long rom1CooldownTimer & unsigned long rom2CooldownTimer	Settes til millis() – rom1CooldownStart (el. rom2CooldownStart) ved hver gjennomkjøring av loopen for å ta tid (ved å sjekke sammen med konstanten ROM_AKTIVITET_TID)
unsigned long rom1CooldownStart & unsigned long rom2CooldownStart	Settes til millis() når et rom først blir aktivt.

Variabel-typer

Foran alle variabel-initieringene kan man se variabel-typen. Dette er nødvendig i kodespråket C. Videre kan variablene brukes kun ved navnet de er gitt.

- Variabeltypen bool er en «boolean», en binær verdi som enten er true eller false (sann eller usann). Brukes som en slags av-på-bryter (rommenes aktivitet, vibrasjon i både klokke og rom).
- Variabeltypen int er en heltallsverdi. Denne lagrer gjerne antall, men brukes i koden også som output- og input-verdier med portene i arduinoen. Heltallet 1 er HIGH og heltallet 0 er LOW.
- Variabeltypen unsigned long er et lenger heltall. I kodespråket C skilles dette fra andre heltall for å effektivisere kjøring. Denne brukes i koden vår for de variablene som får en verdi basert på millis() da disse tallene ofte kan bli veldig store.

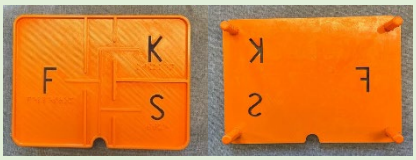
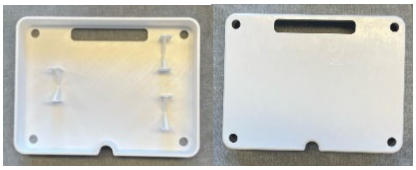
Innebygde funksjoner

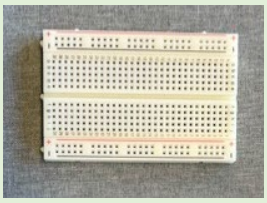
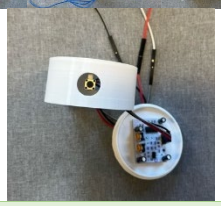
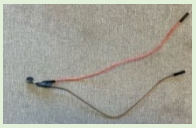
- Funksjonen millis() returnerer antallet millisekunder det har gått siden koden begynte å kjøre. Dette er en innebygd funksjon i Arduino-biblioteket. Vi bruker denne funksjonen mye gjennom koden da mye av funksjonaliteten til artefaktet baserer seg på tidsintervaller.
- Funksjonen begin() blir brukt på objektet Serial. Dette er for å senere kunne bruke funksjonen writeln() på samme objekt, som vi har brukt for å printe ut variabler i en monitor. Det har vært nyttig f.eks. for å se hvilke rom som er aktive.
- Funksjonen pinMode() definerer om en pin er ment for input eller output.
- Funksjonen digitalRead() tar inn en pin og returnerer signalet fra gitt pin. Blir brukt for å lese digitale signal fra sensorene.
- Funksjonen digitalWrite() tar inn HIGH eller LOW (digitale signal) og en pin og sender verdien ut til gitt pin.

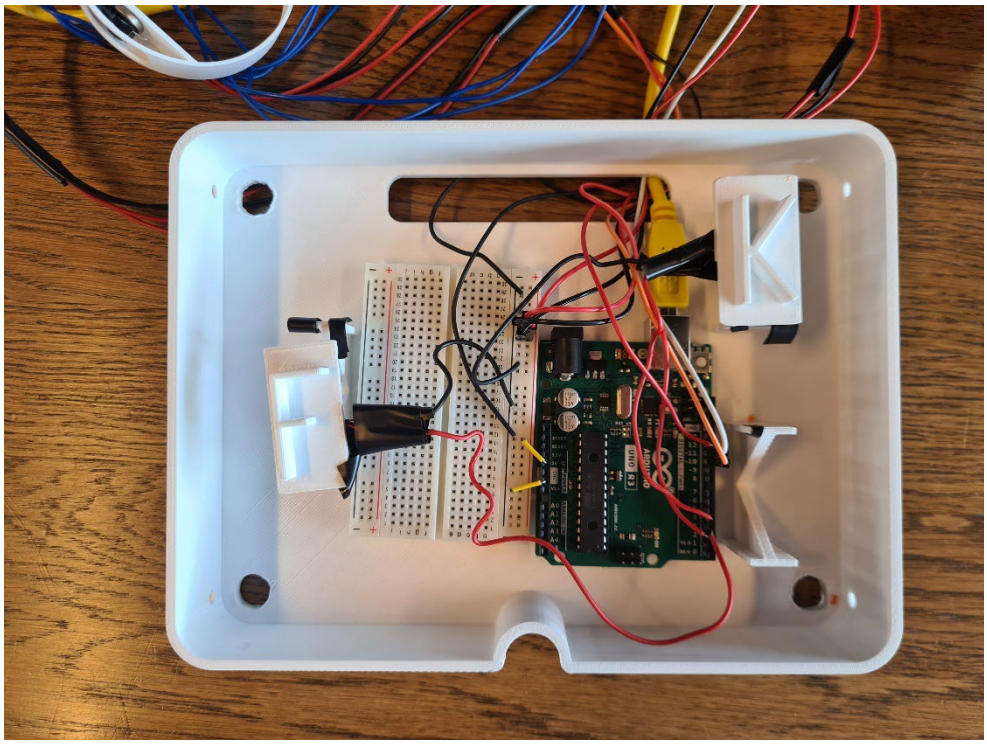
Vi har ikke skrevet egne funksjoner (el. metoder) i vår kode.

IV Komponenter

IV.I Komponentliste

3d-printet del med plantegning Vibrerende bokstaver blir tredd gjennom bokstav-formede hull. Fungerer også som lokk for teknologien som styrer systemet (arduino m. komponenter). Har også ben som tres gjennom hull i boksen under.	
Boks som holder Arduino og vibrerende bokstaver Boksen har plass til alt av komponenter og har stativ for bokstavene med plass til komponenter i midten. Stativene «paddes» med teip for å unngå overdreven vibrasjon-støy.	
Valgfrie ben som «tilter» artefaktet Når man bruker disse bena, blir artefaktet fremoverlent. På denne måten kan man gjøre det tydeligere fra hvilken vei plantegningen skal leses.	
Bokstaver Vibrasjonselementer festes til baksiden. «S» for stue, «K» for kjøkken og «F» for fellesrom (dette er eksempler, tenkte mulige rom.	
3d-printet armbånd/klokke Armbåndet simulerer en smartklokke som ved videre utvikling kan kobles med artefaktet (Apple Watch e.l.). Armbåndet har et lite rom med plass til vibrasjonselement og knapp. 3d-printet knapp limes på arduino-knapp.	 

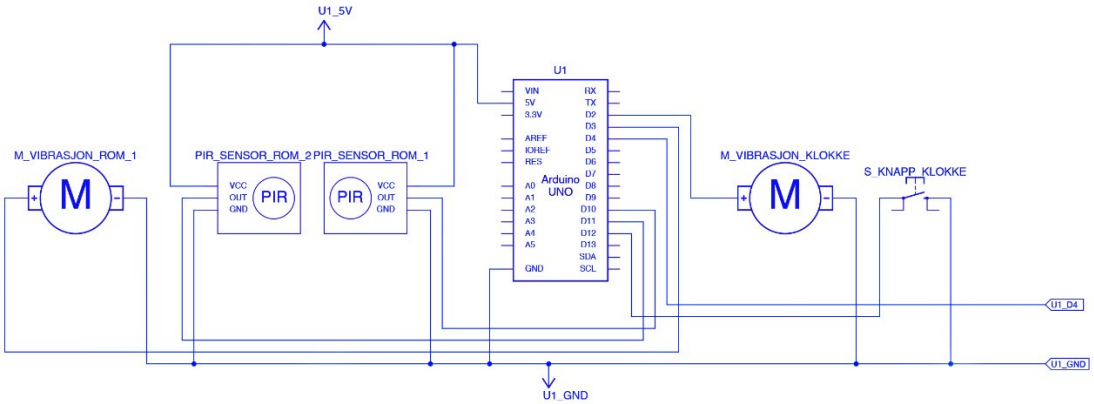
Arduino Kjernen (el. hjernen) av systemet. Her er koden lastet opp og alt av funksjoner styres med denne.	
Breadboard Mange av komponentene er koblet gjennom et breadboard som kom i settet Arduino Student Kit.	
PIR Bevegelsessensorer m. casing PIR (Passive Infrared) Motion Sensor fra Whadda settes inn i casing/stativ. Bevegelsessensor-delen av systemet har også knapp for å kunne manuelt varsle om aktivitet (men er ikke funksjonell i prototypen). Sensorene sender ut høyt digitalt signal i noen sekunder ved bevegelse.	 
Vibrasjonselement Vibrasjonselementer bestilt fra RS Norge. Starter å vibrere med digitalWrite(pin, HIGH) og slutter ved digitalWrite(pin, LOW).	 
Knapp Knapp fra Arduino Student Kit. Sender signal til Arduino ved LOW. Kodet som pullUp.	
Skruer Fire skruer med tilhørende muttere som holder bunnplate og topplate sammen. Mutterne er limt fast til bena i topplate. Skrus fast gjennom bunnplate.	



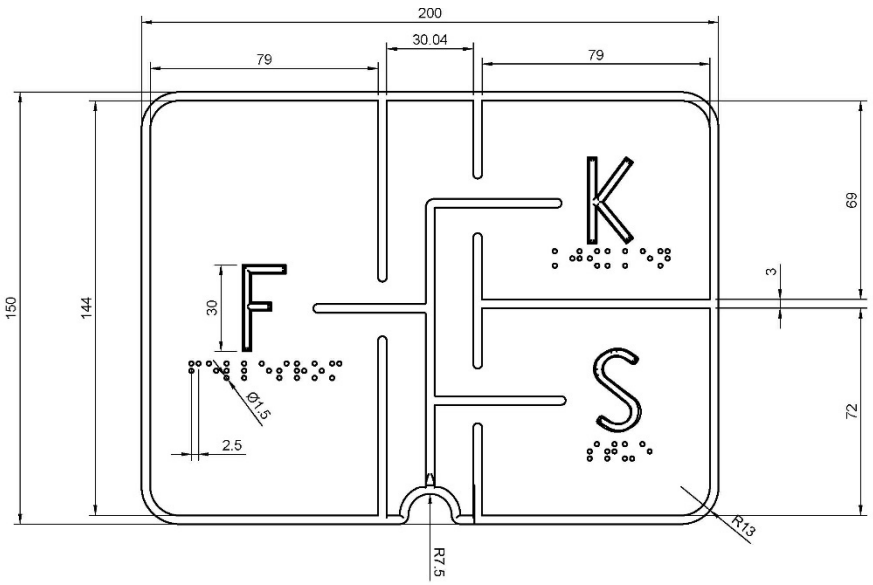
Figur 3: Komponentene inne i bunnplata

IV.II Tekniske tegninger

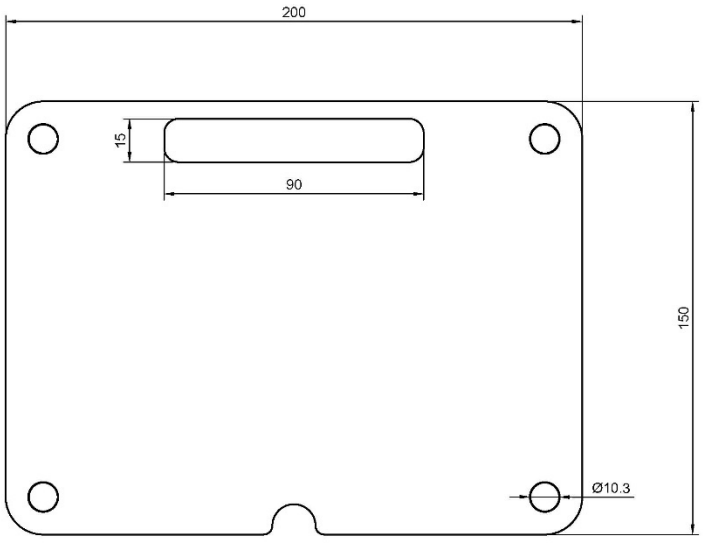
Under følger kretstegning og tekniske tegninger med dimensjoner av artefaktet og tilhørende komponenter.



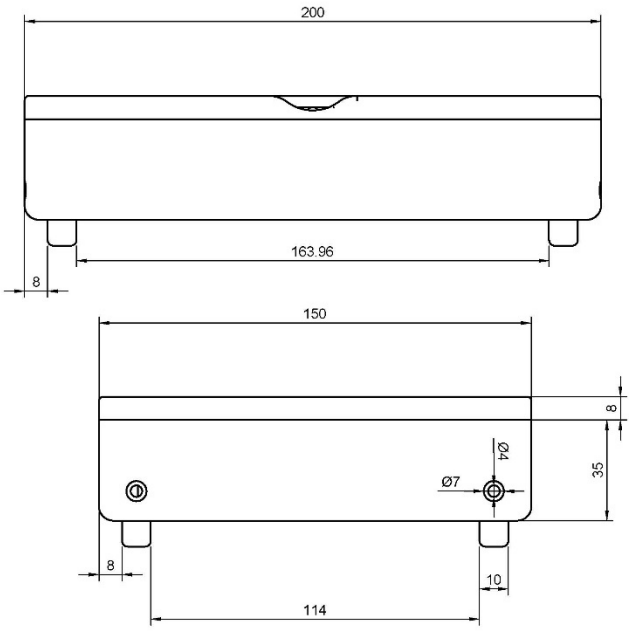
Figur 4: Kretstegning, siste iterasjon av prototype. Laget i Tinkercad



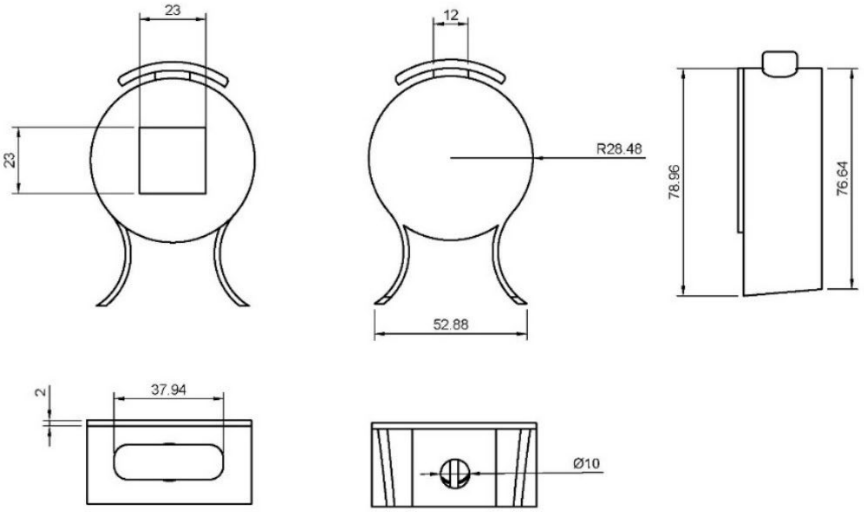
Figur 5: Teknisk tegning av artefaktet fra oversiden



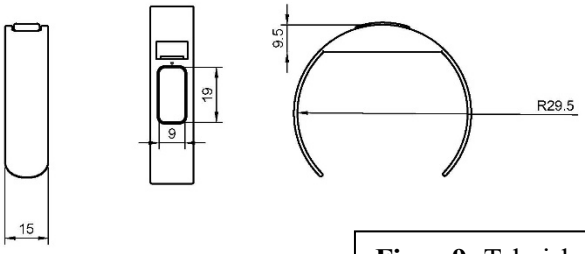
Figur 6: Teknisk tegning av artefaktet fra undersiden



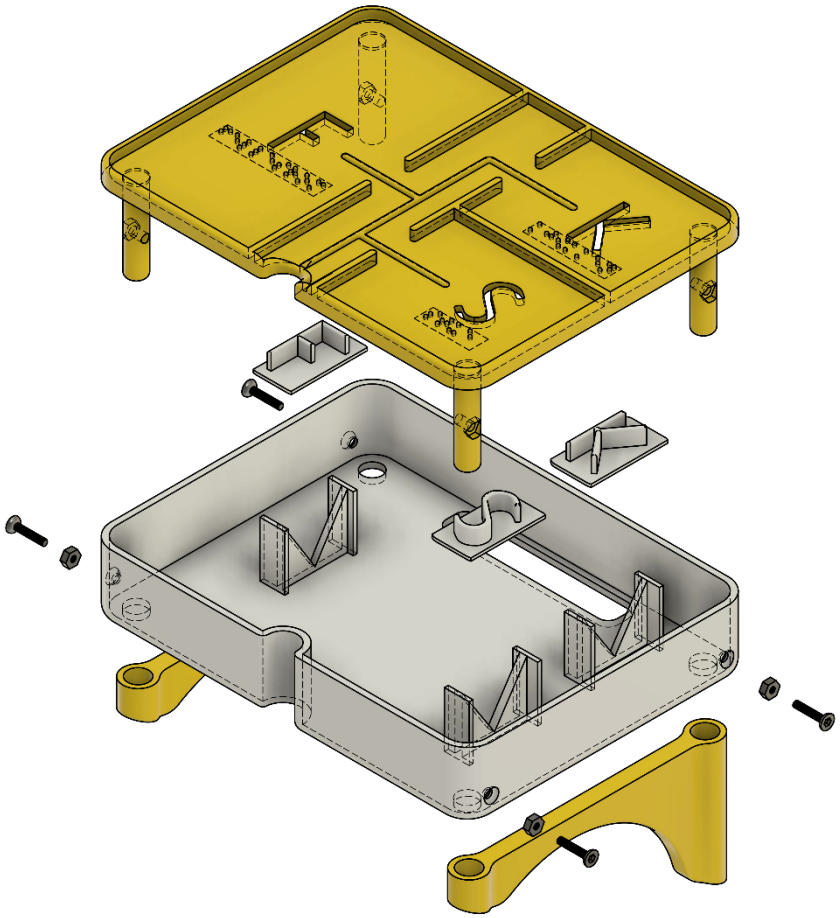
Figur 7: Teknisk tegning av artefaktet fra fremsiden og siden



Figur 8: Teknisk tegning av holder/stativ til bevegelsessensor.



Figur 9: Teknisk tegning av armbåndet.



Figur 10: Utvidet visning av bunn- og topplate uten elektronikk. Illustrerer hvordan de settes sammen.

Referanseliste

Arduino Education. (2023). *Education*. Arduino: https://www.arduino.cc/education

Fitzgerald, S., & Shiloh, M. (2012). *The Arduino Projects Book.*