Teknisk rapport

Den sjette sans

Av

Andrea Karlsen Rye

Marthe Seth

Lara Silberhorn,

Sigrid Strand Stiberg

Pernille Taraldsen Vannebo

Hanna Karsrud

Bruksorientert design
IN1060

Institutt for informatikk
Det matematisk naturvitenskapelige fakultet

UNIVERSITETET I OSLO

Våren 2023

Innhold

1. Presentasjon av prosjektets mål og prototype

Målet med prosjektet var å utvikle en velfungerende prototype, basert på brukerinvolvering og brukernes behov. Vår målgruppe var kontoransatte hos Kongsberg gruppen, som i hovedsak jobber i team. Vi valgte oppgave to, slik at fokuset vårt har vært på sensorteknologi.

Vi ønsket å lage en prototype som gjorde det mulig for brukerne å se kollegaenes tilgjengelighet, i tillegg til å kommunisere sin egen, på tvers av kontorer. Vårt formkonsept er bikake, og konseptet er tilgjengelighet. Artefakten plasseres på brukeren sin egen pult, og den inneholder både egen status og en oversikt over team-medlemmene og/eller andre kollegaer man ofte har kontakt med.

Vi har laget en prototype som implementerer følgende funksjoner:

1. Vise om man er ledig (grønn), opptatt (rød), ønsker å samarbeide (gul/oransje) eller ta pause (blå)
2. Vise om brukeren sitter foran skrivepulten/egen dataskjerm (hvitt lys hvis den ikke registrerer noen)
3. Vise en oversikt over status til team-medlemmer
4. Gi en påminnelse om man har stått på opptatt i en gitt periode
5. Sammensetningen av team-medlemmer kan endres på

Figur 1: Prototypen sett fra ulike vinkler

For å kunne implementere krav nr. 3, må artefaktene kommunisere/sende data til hverandre. Vi hadde ikke fokus på å få implementert dette på grunn av tid, kompleksitet og at det ikke var nødvendig for at få vist prototypens funksjoner til brukerne våre. Vi har også kun laget en prototype.

2. Video

2.1 Link til video

https://vimeo.com/831661028?share=copy

2.2 Beskrivelse av videoen

I videoen viser vi tiltenkt interaksjon og funksjonalitet gjennom et bruks-scenario og en gjennomgang av funksjoner. Videoen er delt inn i tre hoveddeler; introduksjon, bruksscenario(er) og beskrivelse av prototypen. Introduksjonen viser to av gruppemedlemmene som snakker om prototypen, noe som gir en naturlig overgang til bruksscenario. Bruksscenarioet viser en bruker som blir forstyrret selv om hun er opptatt, noe som kan være både irriterende og bryte konsentrasjonen. Videre får vi se samme bruker med artefakten, som etter en tid som "opptatt" vil ha en pause med andre. En annen kollega får med seg at lyset til brukeren endrer seg og inviterer teamet med på kaffe. Beskrivelse av prototypen gir en kort introduksjon til prototypens komponenter og funksjoner.

Vi fikk ikke vist at flere prototyper kommuniserer med hverandre, fordi dette ikke er implementert. Vi simulerer denne interaksjonen ved å ha statiske lys og RGB-lys som skifter farge etter et visst tidsintervall. Mulig interaksjon mellom prototyper kan gjøres via wifi-modul og blir forklart mer i 3.3.

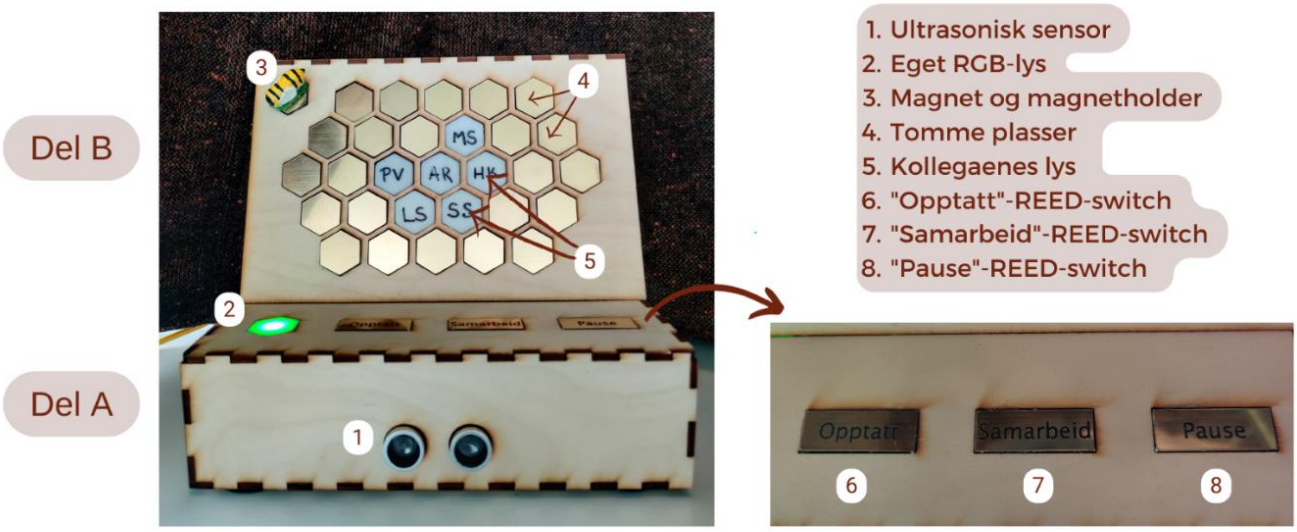
3. Teknisk løsning

3.1 Oversikt

Vår prototype består av to deler:

Del A består av eierens egen seksjon. Eieren har sitt eget lys, som vedkommende kan endre ved å flytte en magnet (formet som en bie) til tre ulike reed-switcher og en magnetholder. I bunnen er det en ultrasonisk sensor. Denne delen utgjør basen av prototypen, og alle funksjonene er riktig implementert.

Del B består av en oversikt over eierens teammedlemmer. Disse er representert med hvert sitt lys bak en hvit heksagon, der initialene til kollegaene er skrevet på. Det er mulig å endre initialene på brikkene, hvis bruker ønsker å endre teamet sitt. De heksagonene som ikke tilhører et teammedlem, er fylt med gull-heksagon. Denne delen utgjør toppen av prototypen og er designet for å vise funksjonalitet som ikke har blitt fullstendig implementert ennå.



Figur 2: En oversikt over komponentene i løsninger og plasseringen av dem

NB! Magneten oppe i venstre hjørne er kun en magnet som kan holde på bien. Grunnen er at man har et sted å plassere bien dersom man kun ønsker å være grønn/ledig. Det er ikke en REED-switch.

3.2 Komponenter

Tabell 1: En oversikt over komponentene brukt i løsningene

Komponenter	Antall	Funksjonalitet
Felles for del A og B		
Arduino Mega	1	For å skrive kode og koble de ulike komponentene sammen
Batteri (9V)	1	Strømkilde
Del A		
Ultrasonic distance sensor (HC-SR04)	1	Beregner avstanden fra et objekt ved å ta utgangspunkt i bølger den sender ut. Bølgen vil til slutt blir reflektert tilbake og registrert hos sensoren. Tiden det tar fra sensoren blir trigget til den registrerer bølgen igjen, brukes til å beregne avstanden. Brukes for å registrere om bruker er til stede
Reed switcher (MR2ME 150 04)	3	Kretsen fullføres hvis den merker at det er et magnetisk felt i nærheten. Brukes til å endre statusene (Se avsnitt 3.5.1 for videre informasjon)
RGB-lys (anode)	1	For å kunne definere egne farger på lyset ved hjelp av RGB-notasjon
Resistor (220 ohm)	3	For å hindre at RGB-LED-en blir ødelagt
Ledninger		Lede strøm
Del B		
RGB (1 anode, 2 katoder)	3	For å kunne vise og endre kollegaenes status for å vise tiltenkt funksjonalitet
Blå-LED	6	Vise et team-medlems status
Gul-LED	5	Vise et team-medlems status
Grønn-LED	6	Vise et team-medlems status
Rød-LED	9	Vise et team-medlems status
Trimmer potensiometer BAOTER3296	4	Variabel motstand koblet til de 4 ulike fargede LEDene. En kjapp løsning for å kunne justere lysstyrke slik at alle LEDene hadde lik lysstyrke.
Resistorer (220 ohm)	9	For å hindre at RGB-lysene blir ødelagt
8-pin female header	1	Header for å feste ledningene fra LEDene.
8-pin screw terminal	1	Kan lodde fast header i den ene enden, og skru fast ledningen i den andre. Gjør det enklere å koble sammen ledninger, uten å måtte lodde de fast.
Ledninger		Lede strøm

3.3 Tiltenkt funksjonalitet

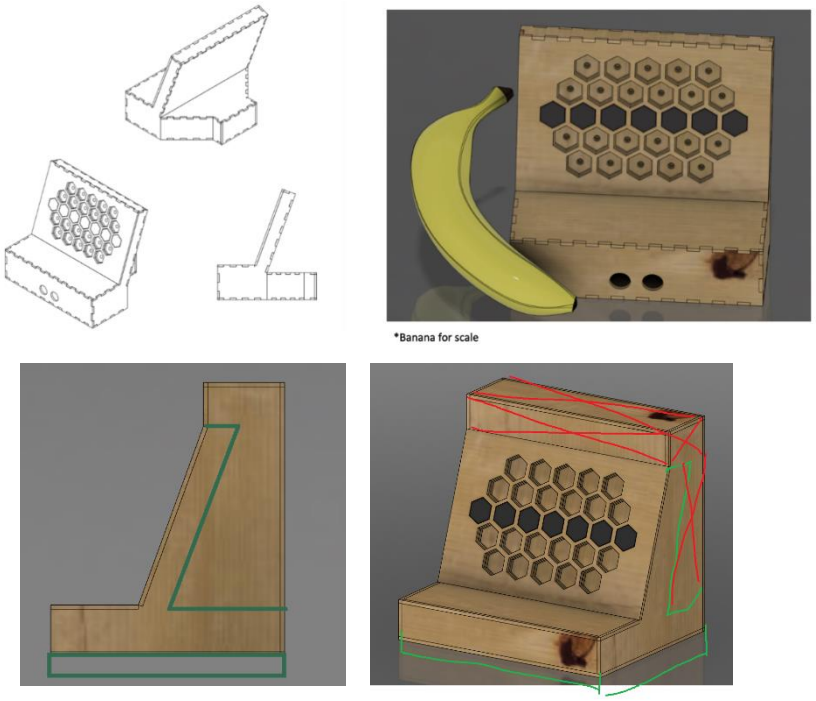
En viktig del av funksjonaliteten til prototypen er at den skal kunne kommunisere med andre like prototyper, men dette fikk vi ikke implementert. Vi har gjort oss opp noen tanker om hvordan vi ville gått frem for å implementere dette.

En tiltenkt plan er å bruke en ESP32 wifi-modul, slik at artefakten kobles til nettet (Getting Started with the ESP32 Development Board, u.d.) Siden løsningen vår er modulær, vil brukerne ha behov for en måte å konfigurere enheten på, etter hvert som teammedlemmer endres. Ved å ha en dedikert nettside for dette, kan brukerne gå inn og konfigurere sin egen enhet. Nettsiden og Arduinoen/ESP32 vil kunne kommunisere sammen via en server.

3.4 Fasade

3.4.1 Laserkutting

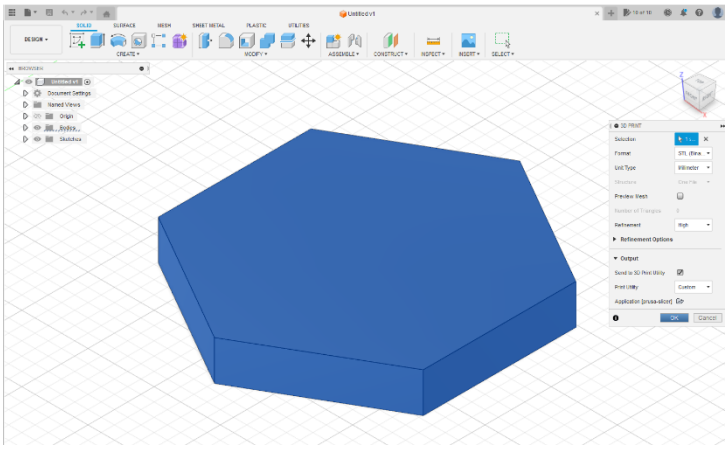
Formen på boksen ble utformet i Fusion 360 (CAD software) og laserkuttet i 3 mm finér. Delene som danner bunnen og sidene på artefakten ble limt sammen med trelim. Resten klikkes på plass, da vi har behov for å komme til inni boksen.



Figur 3: Alle bildene viser hvordan formen ble utformet i CAD

3.4.2 Hvite statusbrikker

De hvite statusbrikkene ble laget av hvit resin. Fusion 360 har en 3D-print funksjon som gjør det raskt og enkelt å konvertere 3D-modellen til en STL-fil. Denne lastet vi opp i Preform, som er Formlab sitt slicer software. Programmet lar oss konfigurere flere ulike parametere, før programmet lagrer dette som G-kode som 3D-printere forstår. Parameterne vi brukte i programmet kan ses i tabell under.



Figur 4: Modellering av statusbrikkene

Tabell 2: Forklaring av ord og valg i forbindelse med 3D-printing

Engelsk	**Forklaring**	**Valg**
Support	Siden modellen printes opp ned, så trenger man et "stillasverk" rundt modellen, slik at ikke modellen knekker eller faller av under print	Resin-printere skiller seg fra vanlig FDM-printere ved at modellen ikke burde røre byggeplattformen (plata modellen printes på). Man burde derfor bruke support i printingen. Om vi hadde brukt en FDM printer, hadde dette vært unødvendig, siden brikkene ikke har overheng e.l.
Layer thickness	3D-modellen blir bygd opp lagvis. Jo tynnere disse lagene er, jo mer detaljerte blir printen. Ulempen med tynn lagtykkelse, er at det tar lenger tid å printe	Vi valgte 0.1mm lagtykkelse (alternativene var 0.1, 0.05 og 0.025mm). Dette fordi brikkene ikke har mange detaljer på seg og det dermed ikke var noen grunn til at velge mer detaljert (noe som ville tatt lenger tid)
Material	Det finnes svært mange ulike typer resin, så man må velge utifra de egenskaper og utseende som er ønsket for modellen	Vi valgte hvit resin, fordi vi tenkte at den ville la LED lyset skinne gjennom, samtidig som det ble mer diffust.

3.4.3 Gullbrikkene

Brikkene i gull ble laserkuttet og gravert på skiltmateriale. For å laserkutte boksen, lagret vi CAD-modellen som en DXF-fil og importerte denne rett inn i laserkutte-programmet. De ulike statusbrikkene er også av samme materiale.

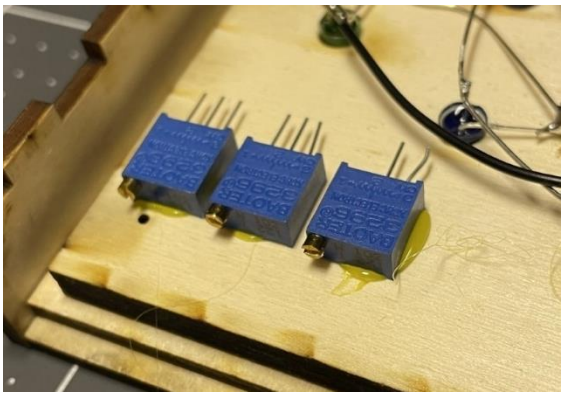
3.4.4 Magnet-bien

Magnet-bien ble lagd ved hjelp av to firkantet magneter og hvit milliputt. Etter at milliputten hadde herdet, ble bien malt med glass/porselens-tusjer.

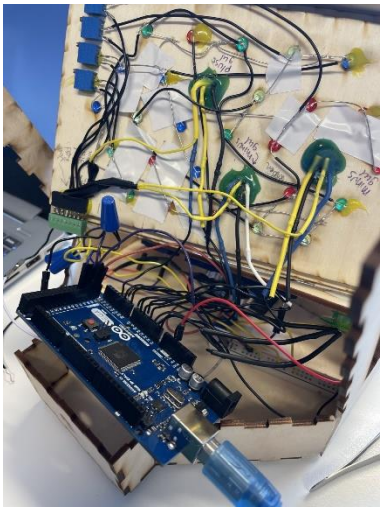
Figur 5: Biemagneten som brukes for å endre status og som fullfører kretsen til Reed-switchene

3.5 Oppkobling

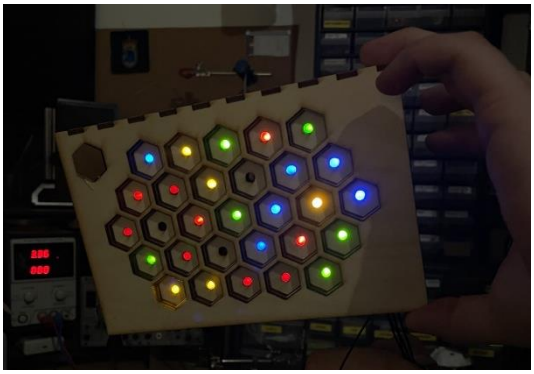
Det viste seg at LED-ene vi brukte hadde ulik styrkegrad. For å kompensere for dette ble LED-ene i hver farge parallellkoblet. Vi brukte derfor 4 trimmer potensiometere (variable motstandere), som man kun justerer motstanden med, ved å skru på en skrue. Siden dette var en prototype, koblet vi det opp slik at vi hadde mulighet til å koble alle LED-ene direkte på et batteri, eller inn i Arduino-brettet, hvis vi ønsket å styre når lysene skulle skrus av eller på. For å unngå alt for mange ledninger, brukte vi lodding og i noen tilfeller wirenett (ser ut som blå skru-korker) til å koble sammen flere ledninger som enten skulle til jord eller 5V. For å unngå kortslutning, brukte vi elektrikerteip, smeltelim og krympestrømper til å isolere der det var behov for dette.



Figur 6: Tre av resistorene som ble brukt til å regulere motstand til de vanlige LED-lysene



Figur 7: Baksiden av den øverste platen med alle lysene



Figur 8: De statiske LED-lysene som lyser

3.5.1 Reed-switchene

Figur 9: Nærbilde (overside) av reed-switch-typen som ble brukt til å endre status. Under er det flere pins på hver side.

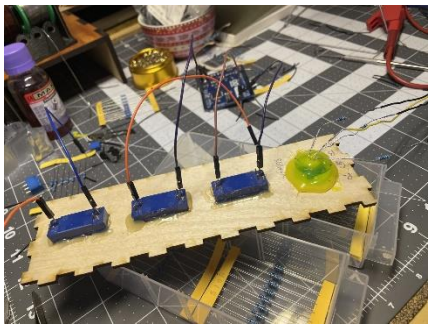
Valget på å bruke reed-switcher ble tatt fordi 1) det var en praktisk og enkelt løsning til formålet og 2) fordi samboeren til en i gruppa hadde svært mange liggende i en skuff. Vi fant ikke noe datablad til disse komponentene, så for å fastslå hvilke pinner som gjorde hva, målte vi motstanden på dem ved hjelp av et multimeter. For å finne ut av dette målte vi motstanden i hver pin, både med og uten en magnet.
Uti fra dette fant vi ut at vi kunne bruke pin nr. 3 og 4 (eller 5 og 6), da disse ikke hadde noen motstand når magneten var koblet til.

Uten magnet						
Pin nr.	1	3	4	5	6	7
1	0	0	0	0	0	6,8
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	6,8	0	0	0	0	0

Med magnet						
Pin nr.	1	3	4	5	6	7
1	0	0	0	0	0	6,8
3	0		X	0	0	0
4	0	X		0	0	0
5	0	0	0		X	0
6	0	0	0	X		0
7	6,8	0	0	0	0	0

Tabell 1.1
Målinger på reed-switch, med og uten magnet. 0 betyr motstand, X betyr ingen motstand, mens det ble målt 6,8 ohm på pin 1 og 7.

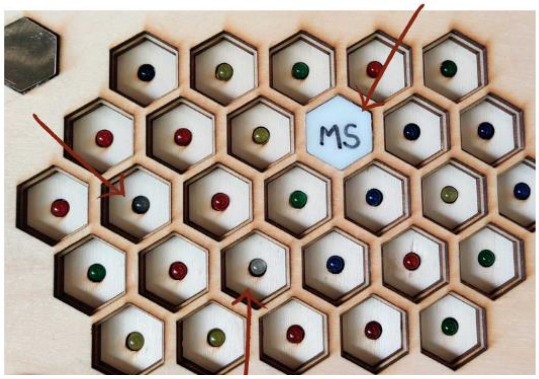
Vi brukte smeltelim til å feste gull-brikkene til reed-switchene, som igjen ble limt fast til plata.



Figur 10: Oppkoblingen av reed-switchene og RGB-lyset

3.5.1 RGB-lys

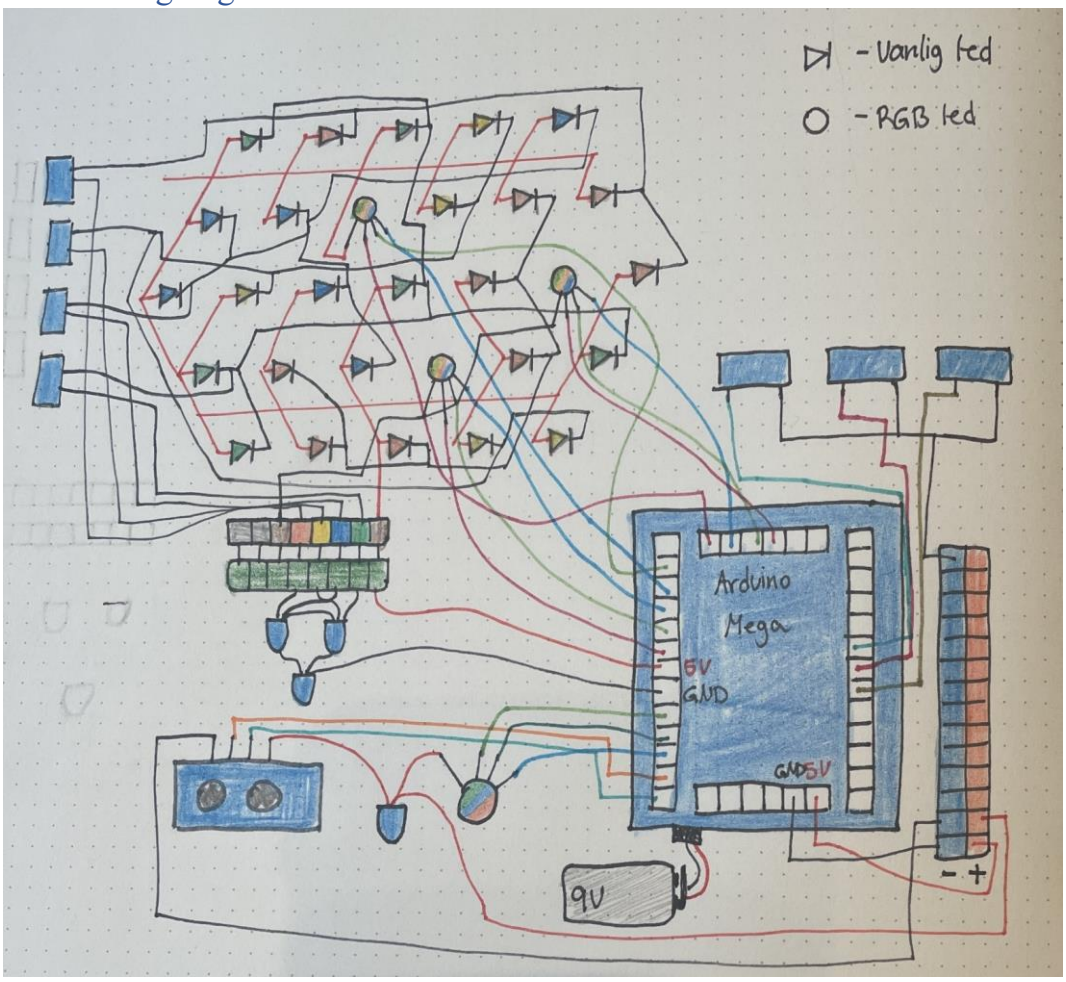
RGB-en som signaliserer brukerens status, ble festet ved hjelp av en undertråd-spole, som tilfeldigvis passer perfekt som en LED-holder.
Vi loddet på motstandere på de tre signalpinne på RGB-ledene. Utover i prosessen ble vi kjent med at det finnes to ulike typer RGB-led, katoder og anoder (How do RGB LEDs work?, 2019). Vi hadde brukt 2 av hver, noe som førte til at vi måtte være ekstra påpasselige med hvilke som skulle kobles til jord og hvilke som skulle ha strøm. Dette er noe vi selvsagt skulle ha fikset om vi hadde hatt mer tid, for det hadde blitt ryddigere både med tanke på oppkobling og Arduino-koden.



Figur 11: Plasseringen av RGB-lys

Ideen er at alle kollegaene har sine egne RGB-lys, men pga. tid og ressurser tok vi kun med tre for å vise tiltenkt funksjonalitet.

3.6 Kretstegning



Figur 13: Håndtegnet krets av hele oppsettet

3.7 Refleksjon og videreutvikling

Vi lærte mye gjennom prosessen av å lage prototypen. Akkurat som resten av designprosessen vår, har dette vært en iterativ prosess, preget av at vi har prøvd oss frem, feilet, rettet det opp, prøvd litt til og forbedret ting underveis. Vi har dermed noen refleksjon over hva vi hadde gjort annerledes, hvis vi hadde fortsatt å utvikle artefakten.

3.7.1 Kobling av LED-lysene

Et par av LED-lysene ble ødelagt underveis i prosessen. Dette tror vi skyldes at de fikk for mye strøm, fordi alle LEDene i samme farge ble parallellkoblet og hadde bare 1 motstander koblet til seg. Artefakten skal i utgangspunktet bestå av mange RGB-LEDs, så en bedre løsning hadde vært å bruke adresserbare LED-strips. På denne måten hadde hver enkelt LED hatt en motstander og det hadde vært enklere i koden å kontrollere lysene.

3.7.2 Oppkobling og komponenter

Oppkoblingen er preget av den utforskende og eksperimenterende tilnærmingen vi hadde når vi lagde artefakten. Vi ser i etterkant at det hadde vært lurt å planlegge og tegne opp alle komponentene i forkant, før vi begynte å lodde og koble alt på plass. Til en ferdig løsning hadde det vært naturlig å designe egne kretskort med monterte komponenter, som kunne vært klikket/skrudd på plass inni boksen. Dette hadde også gjort at vi kunne endret designet til å bli mindre, da komponentene hadde tatt mindre plass.

3.7.3 Fremtidsplaner

Flere av brukerne våre kom med ønsker om at artefakten kunne vært koblet opp mot outlook kalenderen deres. Flere av software-teamene i Kongsberg Gruppen jobber smidig, bruker canban boards og jobber i 3-ukers sprints. En av brukerne sa derfor at det hadde vært kult om løsningen vår på en eller annen måte kunne vært en del av denne prosessen. Ved å bruke nettside-server-ESP32 løsningen vår, kunne vi blant annet tatt i bruk outlook sine APIer, for å integrere dette inn i løsningen vår.

4. Kode

Link: https://github.com/Sig95/IN1060_Den_Sjette_Sans.git

Koden er også limt inn under:

```
// Den Sjette Sans
// av Andrea Rye, Marthe Seth, Lara Silberhorn, Sigrid Strand Stiberg, Pernille Vannebo og Hanna Karsrud

// Pins til ultrasonisk sensor
const int triggerPin = 50;                    // sender ut signal
const int echoPin = 52;                       // tar imot signal
long etappeTid;                               // tiden det tar for den ultrasoniske bølgen å gå fra trigger til echo
int avstand;                                  // avstand (i cm)som sensor måler
```

```
const int grenseAvstand = 80;                    // avstand i cm. Om ultrasonisk sensor detekterer
avstand større enn dette, settes eget rgb lys til hvitt
unsigned long startTid;                          // settes til millis() i det US sensor måler lengre avstand
enn grenseAvstand
const unsigned long grenseTid = 900 * 1000;      // hvis bruker ikke sitter ved pulten etter
grenseTid, vil status automatisk settes til "ikke tilstede".
                                                 // Angis i sekunder * 1000 (for å gjøre om til millisekunder)

// Pins til reed switcher
const int reed1 = 20;                            // kobles til pins med interrupt funksjon
const int reed2 = 21;
const int reed3 = 2;

// Pins egen RGB-led
const int r_egen = 4;
const int g_egen = 5;
const int b_egen = 6;
int statuskode;                                  // holder styr på brukers egen status,  1=opptatt
2=samarbeid   3=pause

// hvor lenge det skal gå før bruker får påminnelse om at de har statusen sin på opptatt
const unsigned long opptattPaaminnelse = 1800000;


unsigned long startTidOpptatt = 0;               // tidspunkt når bruker plasserer magnet på
opptatt-status

// Pins til de 3 rgb ledene som representerer team-medlemmer. Navn nord, sor og vest
korresponderer med plasseringer deres på artefakten
```

```
const int r_nord = A2;
const int g_nord = A1;
const int b_nord = A0;
const int r_sor = A3;
const int g_sor = A5;
const int b_sor = A4;
const int r_vest = A8;
const int g_vest = A6;
const int b_vest = A7;

unsigned long tidSkiftetFarge = millis();        // tidspunkt når en av rgb ledene skiftet farge sist
const unsigned long rgb_grensetid = 8 * 1000;    // lengde tid før ny rgb led skal skifte farge
int teller = 0;                                  // teller for fargemonster array i skiftFarge() metode
int forrigeRGB = 1;                              // holder styr på  hvilken rgb som byttet farge siste

void setup() {
  pinMode(triggerPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(reed1, INPUT_PULLUP);
  pinMode(reed2, INPUT_PULLUP);
  pinMode(reed3, INPUT_PULLUP);
  // om en av reed switchene aktiveres, vil programmet automatisk hoppe til korresponderende
  // funksjon (settOpptatt, settSamarbeid eller settPause) og utføre dette
  attachInterrupt(digitalPinToInterrupt(reed1), settOpptatt, CHANGE);
  attachInterrupt(digitalPinToInterrupt(reed2), settSamarbeid, CHANGE);
  attachInterrupt(digitalPinToInterrupt(reed3), settPause, CHANGE);

  pinMode(r_egen, OUTPUT);
```

```
  pinMode(g_egen, OUTPUT);
  pinMode(b_egen, OUTPUT);

  //Kode som skal utføres kun 1 gang (ved oppstart)
  // sjekker om magneten er plassert på en reed switch. Hvis ja, så settes statuskode til
korresponderende tall
  avgjorEgenStatus();
  //setter rgb lysene til å lyse
  skiftFargeVestSor(1, r_sor, g_sor, b_sor);
  skiftFargeVestSor(3, r_vest, g_vest, b_vest);
  skiftFargeNord(4, r_nord, g_nord, b_nord);
}

void loop() {
  // endrer statuslys basert på verdien til statuskode
  settEgenStatus();
  // ultrasonisk sensor trigges og måler avstand
  trigUltrasoniskSensor();
  // sjekker om bruker er tilstede eller ikke
  sjekkTilstedevaerelse();

  //sjekker om bruker har hatt magnet på "opptatt" over grensetiden. Hvis ja, starter pulsering.
  if ((millis() - startTidOpptatt) > opptattPaaminnelse){
    pulsering();
  }
  // sjekker om det har gått mer tid enn grensetiden, siden siste en rgb led skiftet farge.
  //Hvis ja, skifter farge og oppdaterer tiden dette skjedde
  if((millis() - tidSkiftetFarge) >= rgb_grensetid){
```

```
    skiftFarge();
    tidSkiftetFarge = millis();
  }
}

// De tre "sett"-metodene blir aktivert ved hjelp av interrupt funksjonaliteten på arduino brettet.
Når en av de trigges,
// sjekkes reed-swtichen sin status, statuskoden oppdateres og settEgenStatus() blir kallt på for
å oppdatere lyset
void settOpptatt(){
  if(digitalRead(reed1) == LOW){
    statuskode = 1;
  }
  else{
    statuskode = 0;
  }
  settEgenStatus();
}

void settSamarbeid(){
  if(digitalRead(reed2) == LOW){
    statuskode = 2;
  }
  else{
    statuskode = 0;
  }
  settEgenStatus();
}
```

```
void settPause(){
  if(digitalRead(reed3) == LOW){
    statuskode = 3;
  }
  else{
    statuskode = 0;
  }
  settEgenStatus();
}

// metoden sjekker statuskoden og endrer brukerens rgb led til riktig farge
// startTidOpptatt oppdateres til nåværende tid hvis statusen endres til opptatt, mens den
nullstilles
// hvis statusen endres til noe annet
void settEgenStatus(){
  // opptatt - rød
  if(statuskode == 1){
    analogWrite(r_egen, 0);
    analogWrite(g_egen, 255);
    analogWrite(b_egen, 255);
    if(startTidOpptatt == 0){
      startTidOpptatt = millis();
    }
  }
  // samarbeid - gul
  else if(statuskode == 2){
    analogWrite(r_egen, 0);
```

```
    analogWrite(g_egen, 0);
    analogWrite(b_egen, 255);
    startTidOpptatt = 0;
  }
  // pause - blå
  else if (statuskode == 3){
    analogWrite(r_egen, 255);
    analogWrite(g_egen, 255);
    analogWrite(b_egen, 0);
    startTidOpptatt = 0;
  }
  // tilstede - grønn
  else if (statuskode == 0){
    analogWrite(r_egen, 255);
    analogWrite(g_egen, 0);
    analogWrite(b_egen, 255);
    startTidOpptatt = 0;
  }
  // ikke tilstede - hvit
  else if (statuskode == 4){
    analogWrite(r_egen, 0);
    analogWrite(g_egen, 0);
    analogWrite(b_egen, 0);
  }
}

void trigUltrasoniskSensor(){
  // koden er basert på kode eksempelet på arduino sine nettsider
```

```
  /*****************************************************************
  * Tittel: Ping Ultrasonic Range Finder
  * Dato: 16.05.2023
  * Lenke: https://docs.arduino.cc/built-in-examples/sensors/Ping
  /*****************************************************************/
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delay(10);
  digitalWrite(triggerPin, LOW);
  //leser av tiden det tar for echoPin å registrere signalet
  etappeTid = pulseIn(echoPin, HIGH);
  // regner om til avstand i cm
  avstand = etappeTid * 0.034/2;
}

void sjekkTilstedevaerelse(){
  //nullstiller timer om personen har kommet tilbake til plassen før tidsgrensen har passert
  if (startTid > 0 && avstand < grenseAvstand){
    startTid = 0;
    avgjorEgenStatus();
  }
  // hvis nåværende avstand er over grense og timer ikke har startet
  // start timer (sett start tid til nåværende tid)
  else if (avstand > grenseAvstand && startTid == 0){
    startTid = millis();
    avgjorEgenStatus();
  }
```

```
  // hvis timer har startet og tidsgrensen er passert, sett egen status til ikke tilstede
  // og sjekk at avstanden ikke er endret (om personen har kommet tilbake igjen f.eks.)
  else if (startTid > 0 && (millis() - startTid) >= grenseTid){
    while(avstand > grenseAvstand){
      statuskode = 4;
      settEgenStatus();
      trigUltrasoniskSensor();
    }
    avgjorEgenStatus();
  }
  settEgenStatus();
}

// forskjellen på settEgenStatus() og avgjorEgenStatus er at førstnevnte endrer lyset basert på
statuskoden,
// mens sistenevnte endrer statuskode (ikke lysene) basert på lesing fra reed1 switchen.
void avgjorEgenStatus(){
  if(digitalRead(reed1) == LOW){
    statuskode = 1;
  }
  else if(digitalRead(reed2) == LOW){
    statuskode = 2;
  }
  else if (digitalRead(reed3) == LOW){
    statuskode = 3;
  }
  else{
    statuskode = 0;
```

```
  }
}

void pulsering(){
  // koden er basert på tutorialen funnet på Sparkfun sine nettsider
  /*********************************************************************
  * Tittel: Pulse a LED
  * Forfatter: MikeGrusin
  * Dato: 31.10.2011
  * Lenke: https://www.sparkfun.com/tutorials/329
  *********************************************************************/
  float in, out;
  while(digitalRead(reed1) == LOW){
    for (in = 0; in < 6.283; in = in + 0.001){
      out = sin(in) * 127.5 + 127.5;
      analogWrite(r_egen, out);
      analogWrite(g_egen, 255);
      analogWrite(b_egen, 255);
    }
  }
  // nullstiller timer
  startTidOpptatt = 0;

}

// metoden for å simulere at andre teammedlemmer endrer sin status.
// metoden itererer gjennom et array med tallverdier (som representerer farger) og for hver gang
et RGB lys skifter farge
```

```
// blir forrigeRGB variabelen endret. På denne måten varierer det både hvilken RGB led som
bytter farge, og hvilken farge
// det byttes til

void skiftFarge(){
  int fargemonster[] = {
    // kolonne 1 er rgb sor sitt lysmønster, kol 2 = rgb vest, kol 3 = rgb nord
    1, 3, 4,
    2, 1, 3,
    3, 3, 2,
    4, 1, 3};

    // hvis forrigeRGB var nord, bytt farge på sør og oppdatert forrigeRGB til sør
    if(forrigeRGB == 1){
      forrigeRGB = 2;
      skiftFargeVestSor(fargemonster[teller], r_sor, g_sor, b_sor);
      teller++;
    }
    // hvis forrigeRGB var sør, bytt farge på vest og oppdatert forrigeRGB til vest
    else if(forrigeRGB == 2){
      forrigeRGB = 3;
      skiftFargeVestSor(fargemonster[teller], r_vest, g_vest, b_vest);
      teller++;
    }
    // hvis forrigeRGB var vest, bytt farge på nord og oppdatert forrigeRGB til nord
    else if(forrigeRGB == 3){
      forrigeRGB = 1;
      skiftFargeNord(fargemonster[teller], r_nord, g_nord, b_nord);
```

```
      teller++;
    }
  // nullstiller teller når fargermønsteret har iterert gjennom fargermonster arrayet
  if (teller >= 11){
    teller = 0;
  }
  // oppdatert tiden når siste farge-skifte har skjedd
  tidSkiftetFarge = millis();
}

// metode endrer farge på RGB basert på fargekoden og pin nummer
// siden to av RGB-ledene (vest og sor) er "common cathode", så har de andre fargekoder enn
RGB nord (se egen skiftFargeNord funsjon)
void skiftFargeVestSor(int fargekode, int r, int g, int b){
  // 1: rød    2: gul    3: blå     4: grønn
  if(fargekode == 1){
    analogWrite(r, 255);
    analogWrite(g, 0);
    analogWrite(b, 0);
  }
  else if(fargekode == 2){
    analogWrite(r, 255);
    analogWrite(g, 255);
    analogWrite(b, 0);
  }
  else if(fargekode == 3){
    analogWrite(r, 0);
    analogWrite(g, 0);
```

```
    analogWrite(b, 255);
  }
  else if(fargekode == 4){
    analogWrite(r, 0);
    analogWrite(g, 255);
    analogWrite(b, 0);
  }
}

// metode endrer farge på RGB basert på fargekoden og pin nummer
void skiftFargeNord(int fargekode, int r, int g, int b){
  // 1: rød    2: gul    3: blå     4: grønn
  if(fargekode == 1){
    analogWrite(r, 0);
    analogWrite(g, 255);
    analogWrite(b, 255);
  }
  else if(fargekode == 2){
    analogWrite(r, 0);
    analogWrite(g, 0);
    analogWrite(b, 255);
  }
  else if(fargekode == 3){
    analogWrite(r, 255);
    analogWrite(g, 255);
    analogWrite(b, 0);
  }
  else if(fargekode == 4){
```

```
    analogWrite(r, 255);
    analogWrite(g, 0);
    analogWrite(b, 255);
  }
}
```

5. Referanser

Getting Started with the ESP32 Development Board. (u.d.). Hentet fra RandomNerdTutorials: https://randomnerdtutorials.com/getting-started-with-esp32/

How do RGB LEDs work? (2019). Hentet fra RandomNerdTutorials: https://randomnerdtutorials.com/electronics-basics-how-do-rgb-leds-work/