NTNU

Kunnskap for en bedre verden

MediCare

TTT4270 Elektronisk systemdesign, prosjekt (2024 VÅR)

Forfattere:
Elias Drøpping
Arya Raeesi
Jon Arne Lund
Sven Kristian Helland
Henrik Alstadhaug
Asil Zogby

Dato: 05.05.2024

Innhold

1 Problemstilling og bakgrunn

1.1 Bakgrunn

Fysikalsk medisinsk poliklinikk er St. Olavs hospitals avdeling for langvarige nakke, rygg og skulderplager [1]. Avdelingen er rettet mot personer som har et arbeidsforhold, men som har blitt eller risikerer å bli sykmeldt. Poliklinikken utfører en grundig utredning av plagene og setter deretter opp en behandlingsplan tilpasset pasientens behov. I visse tilfeller kan behandlingen kreve større inngrep hvor klinikken må benytte seg av lokalbedøvelse (lidokain) eller kortison. Legemidler som dette er viktig å kun benytte innenfor holdbarhetsdato og oppbevare på riktig måte. I en samtale med poliklinikken kom det frem at det var en del svinn av lokalbedøvelsen.

Ved nåværende ordning oppbevares det ca 15-20 hetteglass med lokalbedøvelse på 20ml i kjøleskapet. For at legemiddelet ikke skal bli ødelagt, er det viktig at temperaturen i kjøleskapet holder seg mellom 2 og 6 grader celsius. Uåpnede hetteglass har vanligvis holdbarhet 1-2 år frem i tid, og det er sekretærer som holder oversikt over lagerbeholdning, dato og bestiller opp dersom det begynner å bli tomt. En vanlig dose med lokalbedøvelse ligger på mellom 1,5 og 5ml per pasient, som fører til at det blir igjen 15-18,5 ml i det åpnede glasset. Disse åpnede hetteglassene kan ved riktig oppbevaring gjenbrukes frem til de har stått åpne i 3 dager. For å forsøke å minimere svinn, skrives åpningsdatoen for hånd på det gjeldende glasset, slik at de ansatte kan beregne om det kan brukes på nytt eller ikke. For å etterse at medikamentene ikke har blitt eksponert for feil temperatur, måles temperaturen inne i kjøleskapet lokalbedøvelsen oppbevareres i en gang om dagen. Deretter føres temperaturene på et ark slik at det kan dokumenteres at temperaturen er innenfor det bestemte området. Dersom legemidlene har blitt eksponert for feil temperatur eller har gått ut på dato, vil de ofte virke dårligere og kan i noen tilfeller også være farlige å bruke [2].

En stor del av svinnet kommer av at forbruket ikke er stort nok til å kunne bruke opp 20 ml med lokalbedøvelse innen 3 dager hver gang. Dette er imidlertid vanskelig å gjøre noe med uten å få produsenten til å levere hetteglass med mindre volum. Men det er også andre årsaker til svinn som det er lettere å gjøre noe med. «Påskrevet dato for hånd, huske riktig dato og regne seg frem til riktig utløpsdato bør ikke være så vanskelig, men kan være potensielle feilkilder i en hektisk klinisk hverdag og kan mulig stå for noe unødvendig svinn» skriver ansatt Lise Marita Aune på mail. I tillegg kan det oppstå svinn dersom temperaturen i kjøleskapet faller under 2 grader eller overskrider 6 grader, som medfører at lokalbedøvelsen må kastes.

1.2 Fokusområde

Det viktigste forbedringsområdet på poliklinikken er temperaturregisteringen og datooversikten, fordi den nåværende ordningen kan føre til problemer. For eksempel dersom døren på kjøleskapet ikke blir lukket helt eller kjøleskapet slutter å fungere etter at temperaturen har blitt registrert for dagen, vil man ikke finne ut av det før dagen etterpå, dette kan resultere i at legemidlene inne i kjøleskapet må kastes. I verste fall kan det tenkes at kjøleskapsdøren står delvis åpen i løpet av en hel natt, og noen åpner og lukker det igjen dagen etterpå uten

å kontrollere temperaturen. Dersom det går noen timer før temperaturen blir registrert kan det ha blitt kaldt på nytt og en ansatt kan feilaktig anta at alt er i orden. I dette tilfellet vil legemidlene i kjøleskapet potensielt være farlige å bruke, som vil kunne gå utover pasientenes helse. Det kan også tenkes at det er lett å glemme å sjekke datoen på uåpnede hetteglass i en hektisk hverdag ettersom holdbarheten er såpass lang, som også kan få store konsekvenser.

Problemstillingen til dette prosjektet er dermed å lage et system som forbedrer oppbevaringen av legemidler for å forhindre unødvendig svinn av legemidler og farlige situasjoner på poliklinikken.

1.3 Brukerkrav

En behovsanalyse av poliklinikken på St. Olavs universitetssykehus resulterte i følgende brukerkrav:

Krav	**Beskrivelse**
A	Systemet skal passe i kjøleskapet på avdelingen.
B	Systemet skal være enkelt å bruke.
C	Det skal være enkelt å få tak i legemiddelet som går ut på dato først.
D	Det skal gis informasjon om legemiddelet har gått ut på dato eller ikke.
E	Det skal gis informasjon om legemiddelet har stått åpent for lenge eller ikke.
F	Systemet bør se enkelt og ryddig ut.
G	Personalet skal varsles ved avvik i temperaturen i kjøleskapet
H	Det skal være enkelt å sjekke temperaturen i kjøleskapet.
I	Temperaturdata skal lagres slik at personalet kan dokumentere at temperaturen har vært innenfor det gitte området eller ikke.
J	Systemet skal kunne ha kontroll på alle legemidlene i beholdningen.

Tabell 1: Brukerkrav.

2 Konsept

Problemstillingen kan brytes ned i 3 hovedoppgaver:

1. Gi informasjon om temperatur
2. Indikere om et åpent legemiddel kan brukes eller ikke
3. Effektivisere håndteringen av uåpnede medikamenter

For å løse disse oppgavene er det valgt å lage 3 delsystemer som fokuserer på hver sin oppgave.

2.1 Gi temperaturinformasjon

For å kunne gi informasjon om temperaturen, skal det første delsystemet registrere temperaturen i kjøleskapet og sende denne informasjonen til en nettside som de ansatte kan lese av. Dersom temperaturen overskrider 6 grader eller faller under 2 grader skal noen bestemte ansatte få varsel på SMS. Dette vil sørge for at henholdsvis brukerkrav H og G er oppfylt. Systemet skal også lagre temperaturen i en database og beregne gjennomsnittstemperatur det siste døgnet, antall minutter temperaturen har vært over eller under det bestemte området, minimumstemperatur og maksimumstemperatur. Denne dataen skal bli presentert i en nedlastbar tabell fra nettsiden for å oppfylle brukerkrav I. Tabellen fylles gradvis opp slik at den kan lastes ned når som helst ved behov. Etter en uke vil tabellen være full og kunne akriveres, før det blir påbegynt en ny. Ved hjelp av denne oversikten kan man være helt sikker på hvordan temperaturen har utviklet seg og kan ta informerte beslutninger om legemiddelet kan brukes eller ikke.

Figur 1: Ide om hvordan temperatursystemet skal se ut generert av [3].

Figur 1 viser temperatursensoren og databasen (i midten) som laster opp dataen på en nettside som blir aksessert av en datamaskin (lengst til venstre). Temperaturen i dette tilfellet er for høy, så databasen sender en SMS til mobiltelefonene (lengst til høyre) som skal motta varselet.

2.2 Indikere om et åpent legemiddel kan brukes eller ikke

Det andre delsystemet skal ha en egen beholder til hvert av de åpne legemidlene. Systemet skal registrere tidspunktet det åpne medikamentet blir plassert i beholderen og starte en nedtelling tilpasset hvor lenge legemiddelet kan oppbevares åpent. Indikasjonen skal være i form av lys slik at det lyser grønt dersom legemiddelet kan brukes, og rødt dersom det må kastes. Legemiddelet kan dermed tas ut og brukes på nytt så mange ganger man ønsker frem til nedtellingen er ferdig og beholderen lyser rødt. Dette vil oppfylle brukerkrav E, og gjøre at de potensielle feilkildene beskrevet i 1.1 forsvinner, fordi de ansatte slipper å skrive på og beregne datoen selv.

Figur 2: Ide av hvordan stativet for åpne medikamenter kan se ut.

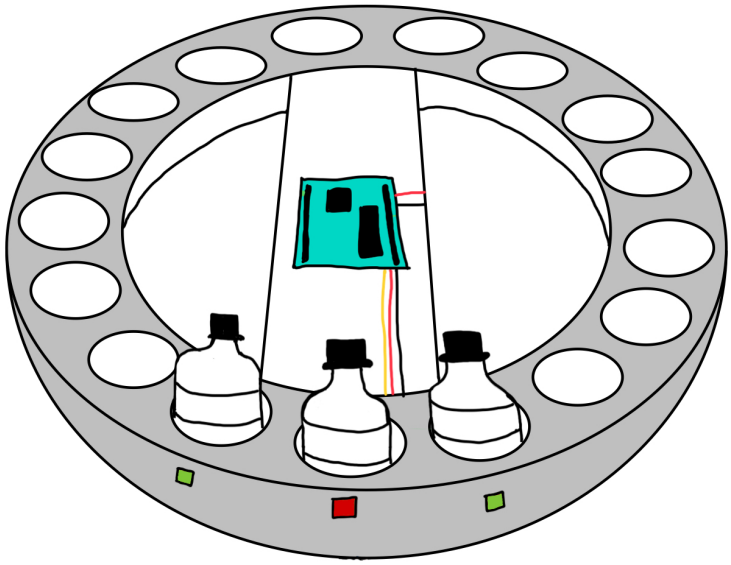
I figur 2 ser man hvordan glassene med lokalbedøvelse kan være plassert i et stativ. Stativet er utstyrt med lys som viser grønt dersom legemiddelet kan gjenbrukes (slik som de to lengst til høyre), eller rødt dersom legemiddelet må kastes (slik som det lengst til venstre).

2.3 Effektivisere håndteringen av uåpnede medikamenter

For å effektivisere håndteringen av medikamenter, skal det tredje delsystemet lese av utløpsdato fra QR-koden på legemiddelembalasjen, og tilordne datoen til beholderen som legemiddelet blir plassert i. Etter at medikamentene er plassert i stativet, lagres utløpsdatoen på de respektive medikamentene i en database, og et LED-lys plassert rett under medikamentet gir indikasjon på om det gjeldene legemiddelet har gått ut på dato eller ikke. Dette vil sørge for at brukerkrav

D er oppfylt. For å gjøre det lettere å få tak i legemiddelet som har kortest holdbarhet i henhold til brukerkrav C, skal systemet bevege seg slik at det gjeldene legemiddelet alltid blir plassert fremst. De ansatte vil dermed kun trenge å ta ut legemiddelet som står fremst hver gang og kunne være trygg på at det er det beste valget for at lagerbeholdningen skal holde lengst mulig. I likhet med det andre delsystemet skal beholderene ha et lys som indikerer om legemiddelet kan brukes (grønt) eller ikke (rødt). Dersom det gjeldene legemiddelet har gått ut på dato, lyser beholderen rødt og venter på at legemiddelet blir kastet før det gjennomfører en ny beregning av hvilke som har kortest holdbarhet. Dette vil forhindre at det oppstår farlige situasjoner ved at de ansatte glemmer å sjekke om legemiddelet fortsatt er holdbart, og gjøre det lettere å hente ut legemiddelet som har kortest holdbarhet.

Det ble valgt å ikke fokusere på brukerkrav J på grunn av økonomisk budsjett, tidsbegrensninger og mangelen på utstyr. Allikevel er det laget et utkast til et mulig design av et stativ som har plass til alle medikamentene til poliklinikken i seksjon 7.



Figur 3: Ide av hvordan stativet for uåpnede medikamenter kan se ut.

Figur 3 viser en skisse av hvordan stativet for de uåpnede hetteglassene kan se ut. Det er tenkt at hjulet skal rotere slik at lokalbedøvelsen som har kortest holdbarhet alltid befinner seg fremst. I denne skissen har lokalbedøvelsen med kortest holdbarhet allrede gått ut på dato, slik at beholderen lyser rødt.

2.4 Systemkrav

Basert på brukerkravene ble det satt opp systemkrav til en tilfredsstillende prototype:

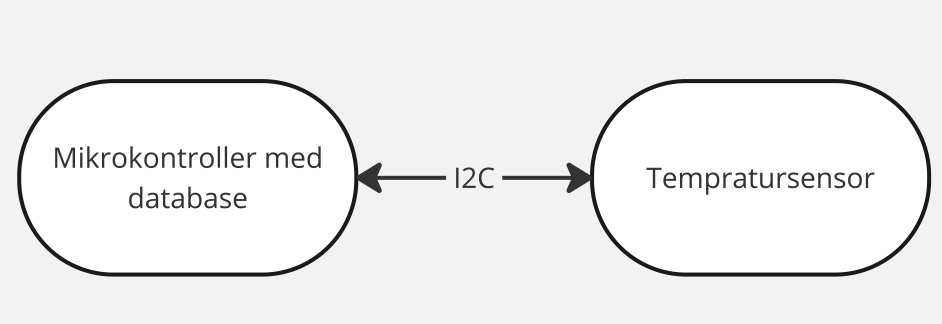
Systemkrav	Spesifikasjon	Aktuelle brukerkrav
1	**Temperatursensor**	
1.1	Systemet skal sende varsling til bruker dersom temperaturen i kjøleskapet går under 2 grader eller over 6 grader.	G
1.2	Systemet skal vise direkte gjennomsnittstemperaturen sist time på en nettside, både som tall og ved plot i en graf.	E, F, H
1.3	Systemet skal lagre ukesrapporter med informasjon om temperaturen har holdt seg mellom 2 og 6 grader i løpet av 24 timers perioder. Dersom temperaturen har vært utenfor området, skal det oppgis hvor lenge dette har vært tilfelle. I tillegg skal gjennomsnittlig temperatur per dag vises i ukesrapportene.	F, H, I
1.4	Skriften til temperaturen på nettsiden skal være rød eller blå dersom temperaturen har vært over 6 grader eller under 2 grader, og grønn ellers.	E, F
1.5	Temperaturmålingene fra temperatursensoren skal sendes til en database i 10 sekunds-intervaller.	H
1.6	Systemet må kunne gi resultater som har maksimalt et avvik fra den reelle temperaturen på 0,5 grader.	H
2	**Stativ for åpne medikamenter**	
2.1	Systemet for åpnede medikamenter skal ha kontroll på hvor lenge et medikament har stått nede i beholderen, og ha et lys som lyser rødt dersom medikamentet har stått åpent i 3 dager og grønt ellers.	E, B
2.2	Brukeren skal kun trenge å plassere det åpnede medikamentet i beholderen, og kaste det når holdbarheten har utløpt.	B
2.3	Systemet skal være under 20 cm i bredde og lengde og total høyde inklusivt medikament kan ikke overskride 60 cm.	A
2.4	Systemet bør være malt i nøytrale farger og elektroniske komponenter bør være skjult.	F
3	**Stativ for uåpnede medikamenter**	
3.1	Systemet skal være under 40 cm i bredde og lengde og total høyde inklusivt medikament kan ikke overskride 30 cm.	A
3.2	Systemet skal ha lys som indikerer om medikamentet som er fremst har gått ut på dato eller ikke.	D
3.3	Systemet for uåpnede medikamenter skal automatisk bevege seg slik at medikamentet som går ut på dato først er lettest tilgjengelig og ha et lys som indikerer om det medikamentet har gått ut på dato eller ikke.	B, C
3.4	Systemet bør være malt i nøytrale farger og elektroniske komponenter bør være skjult.	F

Tabell 2: Spesifikasjon av systemkrav.

3 Design

3.1 Temperatursensor

For å oppfylle systemkrav 1.1 i tabell 2 er det nødvendig å måle temperaturen i kjøleskapet kontinuerlig. Temperatursensoren må styres av en mikrokontroller. For å oppfylle systemkrav 1.2 og 1.3 i tabell 2 må temperaturdataen også lagres. Derfor må mikrokontrolleren lagre data fra temperatursensoren i en database. Deretter kan temperaturdata fra databasen brukes til å beregne en gjennomsnittstemperatur i henhold til systemkrav 1.2 i tabell 2. I tillegg kan temperaturdata fra databasen vises i ukesrapportene i henhold til systemkrav 1.3 i tabell 2. Systemet vises i figur 4. Mikrokontrolleren sender temperaturdata til databasen i 10 sekundersintervaller i henhold til systemkrav 1.5.

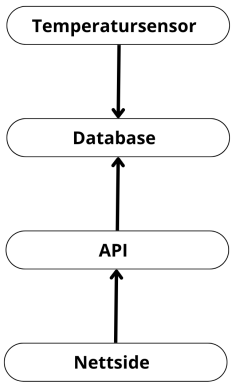


Figur 4: Kommunikasjonen mellom mikrokontroller og temperatursensor.

3.2 Nettside

Nettsiden skal bestå av 4 hovedelementer, en graf som modellerer gjennomsnittlig temperaturdata sist time sammen med en temperaturstatus, gjennomsnittstemperaturen sist time, knapp for nedlasting av temperaturdata sist uke i CSV format og automatisert SMS-varsling til ansatte på poliklinikken. Disse 4 hovedelementene skal sammen oppfylle systemkrav 1.1, 1.2, 1.3 og 1.4 i tabell 2.

En måte man kan sikre kommunikasjon mellom temperatursensoren og nettsiden er ved å benytte en database og API-er (application programming interface), som er et grensesnitt som gir direkte tilgang til data og funksjonalitet i et datasystem [4]. På denne måten kan temperatursensoren sende data direkte til databasen, samtidig som API-ene tar samme data fra databasen og sender det videre til nettsiden. På nettsiden skal dataen vises på en enkel og ryddig måte, for å oppfylle brukerkrav F fra tabell 1. Denne oppførselen fra temperatursensor til nettside er vist i figur 5.

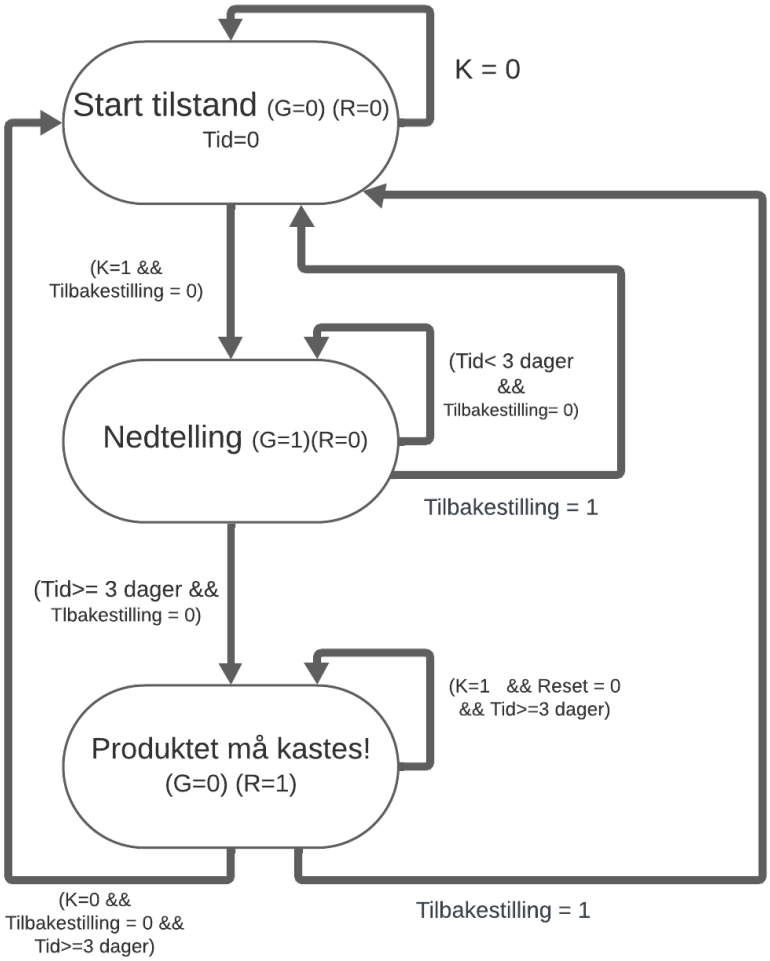


Figur 5: Diagram av kommunikasjon mellom temperatursensor og nettside.

Designet til selve nettsiden skal bestå stort sett av fargen grønn. Dette er bestemt ut ifra fargepsykologi, der fargen grønn kan gi en beroligende effekt, samtidig som det kan bidra med å redusere stress og fremme ro og harmoni [5]. Dette er spesielt praktisk for helsebransjen, der ansatte kan ofte oppleve stress i en hektisk hverdag [6].

3.3 Stativ for åpne medikamenter

I stativet for de åpne medikamentene er hver beholder designet som en tilstandsmaskin for å kunne oppfylle systemkravene. Oppførselen er vist i figur 6.



Figur 6: Tilstandsdiagram for delsystemet for åpne legemidler.

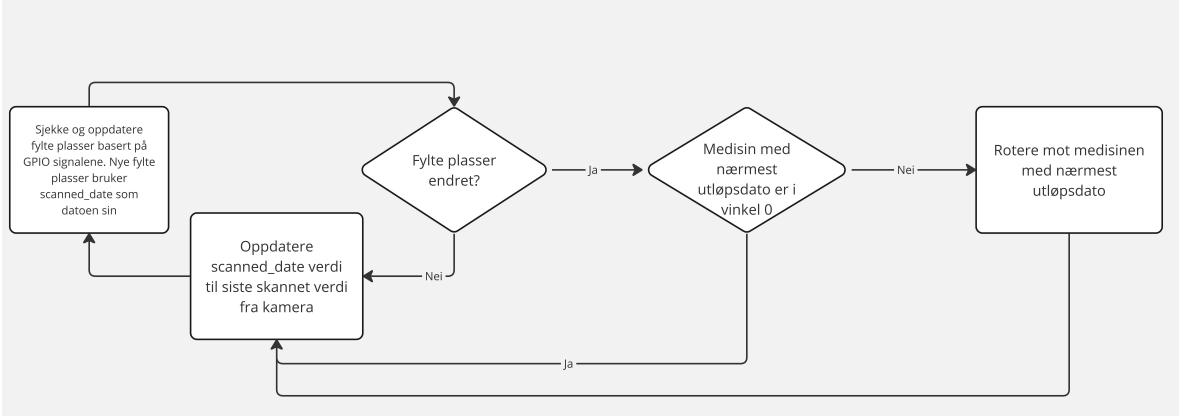
Figuren viser at delsystemet begynner i en starttilstand hvor signalet for grønt lys (G) og rødt lys (R) er logisk lave, og tidsvariabelen (tid) er lik 0. Denne tilstanden representerer en tom beholder. Når ett hetteglass med lokalbedøvelse blir plassert i beholderen, vil det aktivere et signal K (K=1) som tar maskinen til neste tilstand. I nedtellingstilstanden øker tidsvariabelen gradvis til den når 3 dager, som er holdbarheten til et åpnet hetteglass med lidokain. I denne tilstanden er signalet for grønt lys logisk høyt og signalet for rødt lys logisk lavt, noe som indikerer at lokalbedøvelsen kan gjenbrukes. Systemet forblir i denne tilstanden til tidsvariabelen når 3 dager, uavhengig av statusen til signalet K.

Etter 3 dager går systemet automatisk til den tredje og siste tilstanden, som indikerer at lokalbedøvelsen ikke lenger er holdbar og må kastes. I denne tilstanden vil signalet for grønt lys være logisk lavt, og signalet for rødt lys være logisk høyt, for å indikere at hetteglasset må kastes. Systemet forblir i denne tilstanden med rødt lys helt til hetteglasset blir fjernet,slik at signalet K blir logisk lavt. Da går systemet tilbake til starttilstanden der signalet for grønt lys (G) og rødt lys (R) er logisk lave, og tidsvariabelen (tid) er tilbakestilt til 0.

I tillegg er det implementert et tilbakestillings-signal, i tilfelle signalet K aktiveres utilsiktet. Dette tilbakestillings-signalet tar tilstandsmaskinen tilbake til startilstanden uavhengig av hvilke tilstand maskinen befinner seg i, eller statusen til de ulike signalene. Tilstandsdiagrammet gir en detaljert beskrivelse av hvordan systemet fungerer for hver enkelt beholder. Dersom alle beholderene følger dette tilstandsdiagrammet, vil systemkrav 2.1 og 2.2 fra 2 være oppfylt.

3.4 Design av stativ for uåpnede medikamenter

Figur 7 viser oppførselen til stativet. Stativet roterer alltid slik at medikamentet med nærmest utløpsdato er nærmest brukeren. Vinkel posisjonen som er nærmest brukeren er definert som vinkel 0 i figur 7.



Figur 7: Tilstandsdiagram for roterende medikamentstativ

4 Implementering

4.1 Implementasjon av temperaturkontrollsystemet

4.1.1 Komponentvalg

Designet krever en mikrokontroller som har evnen til å styre en temperatursensor, sende data til en database og kjøre en nettside på nettverket. En Raspberry Pi har evnen til å utføre alle disse oppgavene. Ulempen med Raspberry Pi er at den forbruker mye strøm, men siden gruppemedlemmene har erfaring med linux operativsystemet, er det mye enklere å bruke en Raspberry Pi. Derfor brukes Raspberry Pi som mikrokontroller for dette systemet.

4.1.2 Temperatursensor

Programvare
Sensoren bruker i2c, som nevnt i design delen. BME680 [7] sensoren benyttes siden den støtter og i2c og var lett tilgjengelig. For å bruke sensoren benyttes BME680 biblioteket [8]. I biblioteket finner vi en funksjon som leser data fra sensoren, denne funksjonen vises under i kodeblokk 1.

```
1  int8_t bme680_get_sensor_data(struct bme680_field_data *data, struct bme680_dev *dev)
2  {
3          int8_t rslt;
4
5          /* Check for null pointer in the device structure*/
6          rslt = null_ptr_check(dev);
7          if (rslt == BME680_OK) {
8                  /* Reading the sensor data in forced mode only */
9                  rslt = read_field_data(data, dev);
10                 if (rslt == BME680_OK) {
11                         if (data->status & BME680_NEW_DATA_MSK)
12                                 dev->new_fields = 1;
13                         else
14                                 dev->new_fields = 0;
15                 }
16         }
17
18         return rslt;
19 }
```

Kodeblokk 1: Lese funksjonen inkludert i BME680 biblioteket [8].

Funksjonen i kodeblokk 1 benytter `struct bme680_dev`, som vises i kodeblokk 2.

```c
struct        bme680_dev {
        /*! Chip Id */
        uint8_t chip_id;
        /*! Device Id */
        uint8_t dev_id;
        /*! SPI/I2C interface */
        enum bme680_intf intf;
        /*! Memory page used */
        uint8_t mem_page;
        /*! Ambient temperature in Degree C */
        int8_t amb_temp;
        /*! Sensor calibration data */
        struct bme680_calib_data calib;
        /*! Sensor settings */
        struct bme680_tph_sett tph_sett;
        /*! Gas Sensor settings */
        struct bme680_gas_sett gas_sett;
        /*! Sensor power modes */
        uint8_t power_mode;
        /*! New sensor fields */
        uint8_t new_fields;
        /*! Store the info messages */
        uint8_t info_msg;
        /*! Bus read function pointer */
        bme680_com_fptr_t read;
        /*! Bus write function pointer */
        bme680_com_fptr_t write;
        /*! delay function pointer */
        bme680_delay_fptr_t delay_ms;
        /*! Communication function result */
        int8_t com_rslt;
};
```

Kodeblokk 2: En struct som inkluderer alle BME680 konfigurasjoner.

Vi ser at `struct bme680_dev` har tre funksjonspekere. Funksjonene som funksjonspekerne skal peke til må implementeres (se kodeblokk 6, 7 og 8). Grunnen til at disse funksjonene er ikke implementert i biblioteket er fordi implementasjonen varierer basert på typen enhet som bruker sensoren. For eksempel er implementasjonen av disse funksjonene forskjellig på Arduino sammenlignet med Raspberry Pi.

```c
int8_t user_i2c_read(uint8_t dev_id, uint8_t reg_addr, uint8_t *reg_data, uint16_t len)
{
    int8_t rslt = 0; /* Return 0 for Success, non-zero for failure */

    /*
     * The parameter dev_id can be used as a variable to store the I2C address of the device
     */

    /*
     * Data on the bus should be like
     * |------------+---------------------|
     * | I2C action | Data                |
     * |------------+---------------------|
     * | Start      | -                   |
     * | Write      | (reg_addr)          |
     * | Stop       | -                   |
     * | Start      | -                   |
     * | Read       | (reg_data[0])       |
     * | Read       | (....)              |
     * | Read       | (reg_data[len - 1]) |
     * | Stop       | -                   |
     * |------------+---------------------|
     */

    return rslt;
}
```

Kodeblokk 3: Beskrivelsen av i2c lese funksjonen hentet fra BME680 biblioteket [8].

```c
int8_t user_i2c_write(uint8_t dev_id, uint8_t reg_addr, uint8_t *reg_data, uint16_t len)
{
    int8_t rslt = 0; /* Return 0 for Success, non-zero for failure */

    /*
     * The parameter dev_id can be used as a variable to store the I2C address of the device
     */

    /*
     * Data on the bus should be like
     * |------------+---------------------|
     * | I2C action | Data                |
     * |------------+---------------------|
     * | Start      | -                   |
     * | Write      | (reg_addr)          |
     * | Write      | (reg_data[0])       |
     * | Write      | (....)              |
     * | Write      | (reg_data[len - 1]) |
     * | Stop       | -                   |
     * |------------+---------------------|
     */
    return rslt;
}
```

Kodeblokk 4: Beskrivelsen av i2c skrive funksjonen hentet fra BME680 biblioteket [8].

For å lese fra i2c på Raspberry Pi, må vi lese fra filen '/dev/i2c-1'. '/dev/i2c-1' er i2c bussen til Raspberry Pi. For å skrive data til sensoren må Raspberry pi skrive til 'dev/i2c-1'. Vi begynner med å åpne '/dev/i2c-1' med read/write permissions og lagre 'file descriptor' i en variabel (se kodeblokk 5).

```c
int i2c_fd;

void i2cOpen()
{
        i2c_fd = open("/dev/i2c-1", O_RDWR);
        if (i2c_fd < 0)
        {
                perror("i2cOpen");
                exit(1);
        }
}
```

Kodeblokk 5: Koden åpner i2c bussen for data skriving og lesing.

Dataen fra sensoren kan hentes ved å lese fra `i2c_fd`. Fra beskrivelsen i kodeblokk 3 ser vi at vi må skrive register adressen `reg_addr` (1 byte) til i2c bussen. Etter at `reg_addr` er sendt, så leser vi `len` antall bytes fra i2c bussen. Implementasjonen vises i kodeblokk 6.

```c
int8_t user_i2c_read(uint8_t dev_id, uint8_t reg_addr, uint8_t *reg_data, uint16_t len)
{
        int8_t rslt = 0; /* Return 0 for Success, non-zero for failure */

        uint8_t reg[1];
        reg[0] = reg_addr;

        if (write(i2c_fd, reg, 1) != 1)
        {
                perror("user_i2c_read_reg");
                rslt = 1;
        }
        if (read(i2c_fd, reg_data, len) != len)
        {
                perror("user_i2c_read_data");
                rslt = 1;
        }

        return rslt;
}
```

Kodeblokk 6: Implementasjon basert på beskrivelsen i 3.

For å skrive data til sensoren implementeres en funksjon basert på beskrivelsen i kodeblokk 4. Først skrives registeradressen til sensoren, etterfulgt av dataen som skal skrives. Implementasjonen av skrivefunksjonen vises i kodeblokk 7.

```c
int8_t user_i2c_write(uint8_t dev_id, uint8_t reg_addr, uint8_t *reg_data, uint16_t len)
{
        int8_t rslt = 0; /* Return 0 for Success, non-zero for failure */

        uint8_t reg[16];
        reg[0] = reg_addr;

        for (int i = 1; i < len + 1; i++)
                reg[i] = reg_data[i - 1];

        if (write(i2c_fd, reg, len + 1) != len + 1)
        {
                perror("user_i2c_write");
                rslt = 1;
                exit(1);
        }

        return rslt;
}
```

Kodeblokk 7: Implementasjon basert på beskrivelsen i 4.

Implementasjonen av delay-funksjonen vises i kodeblokk 8.

```c
void user_delay_ms(uint32_t period)
{
        sleep(period / 1000);
}
```

Kodeblokk 8: Implementasjon av en delay-funksjon for Raspberry Pi.

Nå er det mulig å initialisere `struct bme680_dev` og sette funksjonspekerne til de implementerte funksjonene fra kodeblokkene 6, 7 og 8. I tillegg må blant annet riktig i2c addresse og oversampling konfigurasjoner settes. Data fra sensoren kan hentes ved å benytte funksjonen i 1.

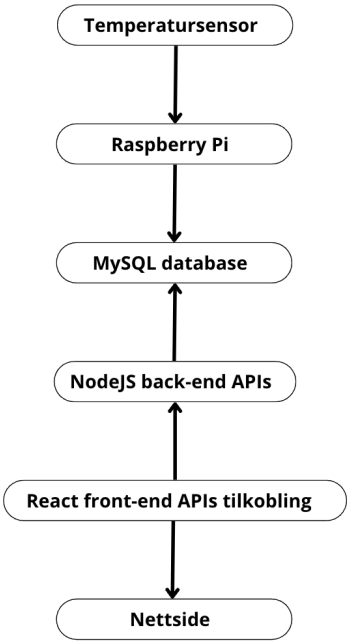
Data fra sensoren må sendes til MySQL databasen. For å skrive data til databasen benyttes en 'INSERT INTO' query. Dette vises i kodeblokk 9

```
char query[256];
            t = time(NULL);
            tm = *localtime(&t);

            sprintf(
                    query,
                    "INSERT INTO temperatur (id, temperatur, tid, dato) VALUES (%d, %.2f,
                    ↪  '%d:%d:%d', '%d-%d-%d')",
                    i,
                    data.temperature / 100.0f,
                    tm.tm_hour,
                    tm.tm_min,
                    tm.tm_sec,
                    tm.tm_year + 1900,
                    tm.tm_mon + 1,
                    tm.tm_mday);

            int mysqlStatus = mysql_query(conn, query);

            if (mysqlStatus)
            {
                    printf("mysqlstatus = %d\n", mysqlStatus);
            }
```

Kodeblokk 9: Koden temperaturdata med dato til MySQL databasen.

4.1.3 Implementasjon av nettside

Som tidligere nevnt, inneholder nettsiden 4 hovedelementer, en graf som modellerer temperaturdata, gjennomsnittstemperaturen, knapp for nedlasting av temperaturdata i CSV format og automatisert SMS-varsling til ansatte ved poliklinikken. For å få til denne konfigurasjonen er det benyttet av en temperatursensor, som sender temperaturdata til en Raspberry Pi. Videre overfører Raspberry Pi-en temperaturdataen til en MySQL database. Deretter er nettsiden satt opp med en back-end side og en front-end side, der back-end siden er lagd med NodeJS, mens front-end siden er lagd med React med Typescript. Back-end siden av en nettside refererer til server-siden, hvor nettsiden funksjonalitet og struktur er implementert [9]. Front-end siden derimot, refererer til brukersiden, hvor en bruker kan se og navigere gjennom innholdet til nettsiden [9]. I back-end siden er de aktuelle API-ene for å oppfylle systemkrav 1.1, 1.2, 1.3 og 1.4 fra tabell 2 satt opp og tar data fra MySQL databasen. Så består front-end siden av flere funksjoner, som ber om dataen fra back-end API-ene. Til slutt blir den mottatte dataen oppført på nettsiden på en enkel og ryddig måte, for å oppfylle brukerkrav F fra tabell 1. Denne oppførselen er vist i figur 8.

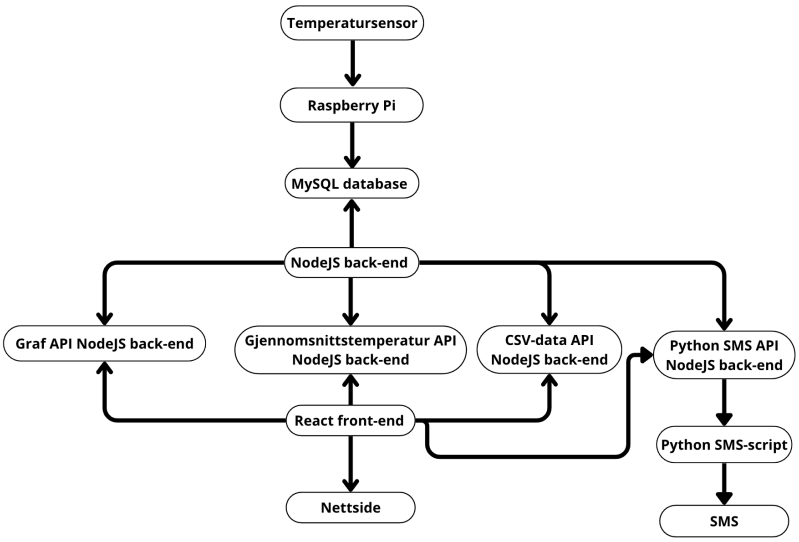


Figur 8: Diagram av kommunikasjon mellom temperatursensor og nettside.

Grunnen til at en MySQL databse, NodeJS og React med Typescript benyttes er som følger:

- MySQL database: Det finnes mange typer med databasetyper, for eksempel MongoDB, PostgreSQL og Oracle databaser. Vi valgte å benytte en MySQL database siden det er et databasesystem som vi har tidligere erfaring med.
- NodeJS: Siden nettsiden skal vise endringer kontinuerlig, er det nyttig å benytte et serversystem for å strukturere dataen som brukes i endringene. Det finnes andre type serversystemer, som for eksempel PHP, men vi valgte å bruke NodeJS siden det er et mer moderne system som blir brukt mer og mer i arbeidslivet [10].
- React med Typescript: Det finnes mange måter å lage et front-end system til en nettside på. For eksempel kan en front-end side lages med HTML og CSS. Vi valgte å bruke React med Typescript, siden det er et mer moderne system og kommuniserer godt med NodeJS. I tillegg brukes det mer og mer i arbeidslivet [10].

For å få til konfigurasjonen som er ønsket av systemkrav 1.1, 1.2, 1.3 og 1.4 i tabell 2, er det nødvendig med spesifikke API-er mellom back-end siden av nettsiden og databasen. Denne mer komplekse oppførselen er vist i figur 9.



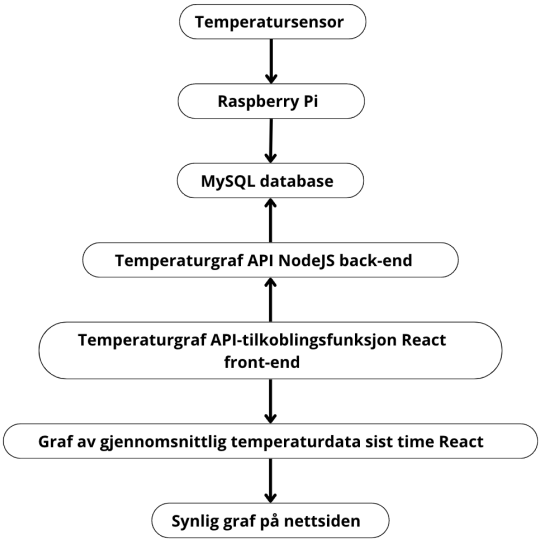
Figur 9: Diagram av kommunikasjon mellom temperatursensor og nettside med spesifkke API-er.

Dataen fra API-ene som er synlig på nettsiden, altså dataen til grafen og dataen til gjennomsnittstemperaturen, oppdateres hvert 10. sekund på front-end siden. Dette fører til at dataen på nettsiden endres på direkten, som hjelper med å oppfylle systemkrav 1.2 i tabell 2. Kodesnutten for oppdateringsmodellen til nettsiden vises i vedlegg A.

Nedenfor vil vi gjennomgå de 4 hovedelementene til nettsiden. Først vil diverse diagrammer (figur 10, 12, 16 og 19) vise hvordan Raspberry Pi-en funker som et mellomledd for temperatursensoren og MySQL databasen, slik at temperaturdata overføres til databasen på en konsistent og stabil måte. Denne utsendingen av data til databasen gjennomføres hvert 10. sekund. Deretter vises det hvordan vi har definert hver API i NodeJS. Videre vises det hvordan vi har koblet API-ene med front-end siden av nettsiden, lagd med React med Typescript. Så vises hvordan det ser ut på selve nettsiden. SMS-varslingssystemet har litt større konfigurasjon, men resten av hovedelementene vil følge rekkefølgen vi forklarte nå. Til slutt skal det forklares hvordan fargepsykologi er blitt implementert i nettsiden.

Graf

Først og fremst skal oppførselen til grafen gjennomgås.



Figur 10: Diagram av kommunikasjon mellom temperatursensor og nettside spesifikt for graf inne på nettsiden.

En API er blitt utviklet i NodeJS som henter sist times temperaturdata fra databasen som en array. Grunnen til at sist times temperaturdata hentes ut, er at systemkrav 1.2 i tabell 2, ber spesifikt om sist times temperaturdata i en graf.
I tillegg skal gjennomsnittlig temperatur i 5 minutters-intervaller plottes. Dette vil gi en bedre indikasjon til brukeren om temperaturen faktisk har vært for høy eller ikke, siden individuelle temperaturmålinger gjelder for små intervaller. Individuelle temperaturmålinger vil sannsynligvis ikke påvirke medikamentene på en negativ måte, om det skulle være over 6 grader eller under 2 grader. Større temperaturintervaller derimot, kan vise skade i medikamentene bedre. Grunnen til dette er at det viser temperaturen over lengre tid, som kan indikere til brukeren om medikamentene har vært utsatt for en ikke-gunstig temperatur for lenge. Dermed er det mer relevant for brukeren å se på lengre temperaturintervaller i grafen, enn det som måles av temperatursensoren.
Kort sagt vil grafen hjelpe til med systemkrav 1.2 i tabell 2, med tanke på at det modellerer endringen i temperatur, på en forståelig og intuitiv måte for brukerene på poliklinikken.

For å få til kommunikasjonen beskrevet i figur 10 er flere kodesnutter lagd. Først har vi utviklet API-en til grafen i NodeJS. Koden til denne API-en vises i vedlegg A. Det er flere utregninger som er blitt utført for å sette opp API-en, blant annet hvor mange målinger det utføres per time, og intervall-lengden til en 5 minutters intervall.

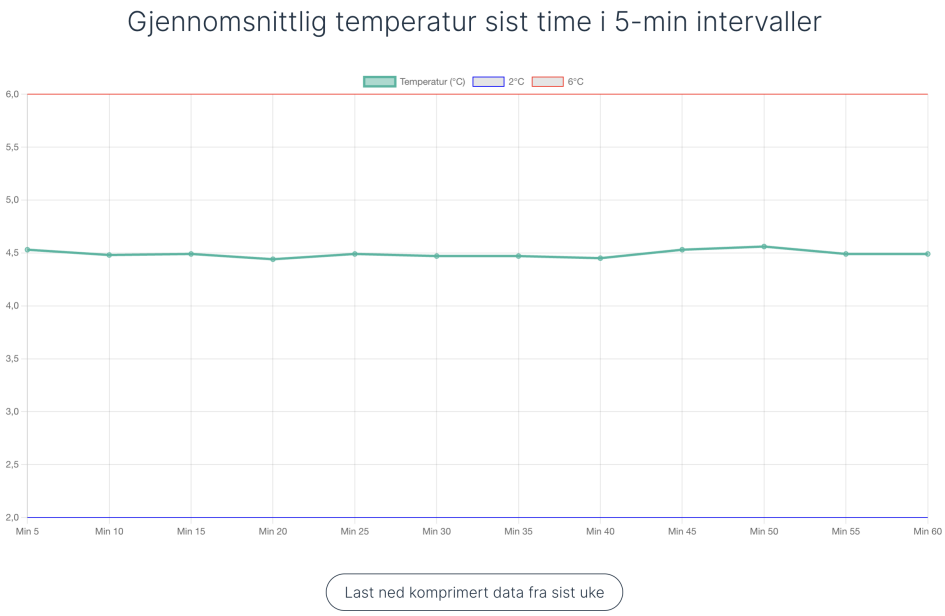
De siste 360 temperaturmålingene hentes ut i API-en. Grunnen til dette er at temperatursensoren måler 360 ganger i løpet av 1 time. Denne utregningen er mulig å se med likning 1 nedenfor.

$$\begin{aligned}\text{Antall målinger per time} &= \frac{\text{Antall sekund per time}}{\text{Antall sekund per måling}} \\ &= \frac{60 \times 60}{10} \\ &= 360 \end{aligned} \tag{1}$$

I tillegg splittes de 360 målingene i 12 intervaller, der hver intervall er 30 temperaturmålinger lang. Grunnen til dette er at 1 time består av 12 5-minutters intervaller. Denne utregningen er vist nedenfor i likning 2.

$$\begin{aligned}\text{Intervall-lengde til graftemperatur} &= \frac{\text{Antall målinger per time}}{\frac{\text{Antall minutter per time}}{\text{5 minutt}}} \\ &= \frac{360}{\frac{60}{5}} \\ &= 30 \end{aligned} \tag{2}$$

Deretter samler API-en alle målingene i en array (se vedlegg A). Denne arrayen blir bedt om av en funksjon på front-end siden av nettsiden (React med Typescript). Deretter blir arrayen satt opp i en graf inne på front-end siden (se vedlegg A), som blir til slutt vist på nettsiden. Under vises grafen på nettsiden i figur 11.



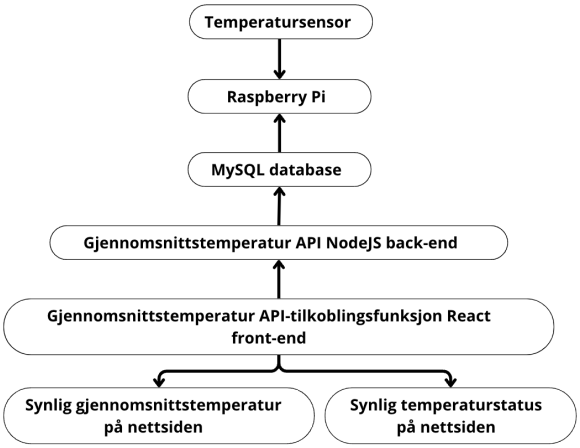
Figur 11: Illustrasjon av hvordan grafen ser ut etter temperaturdata fra back-end API-en utviklet med NodeJS er satt inn i den.

Som nevnt i design delen skal nettsiden baseres på fargen grønn, med tanke på fargepsykologi. Dette kan man se i figur 11, der grafen med temperaturdata består av fargen grønn. I tillegg har den øvre temperaturgrensen (6 grader) fargen rød, som ofte assosieres med varme, mens den nedre temperaturgrensen (2 grader) har fargen blå, som ofte assosieres med kulde.

Til slutt er det mulig å se i figur 11 at de individuelle temperaturmålingene er delt opp i 5 minutters-intervaller, der gjennomsnittet av hver 5 minutters-intervall er plottet.

Gjennomsnittstemperatur

Deretter skal oppførselen til gjennomsnittstemperaturen gjennomgås.

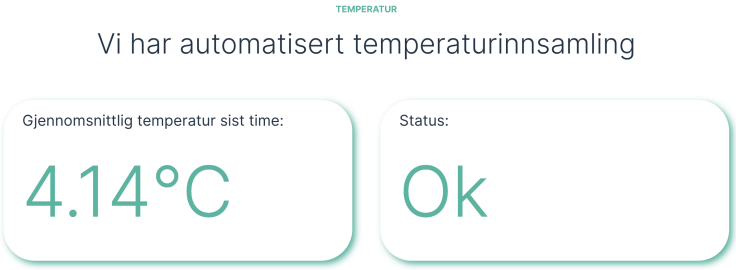


Figur 12: Diagram av kommunikasjon mellom temperatursensor og nettside spesifikt for gjennomsnittstemperatur og temperaturstatus.

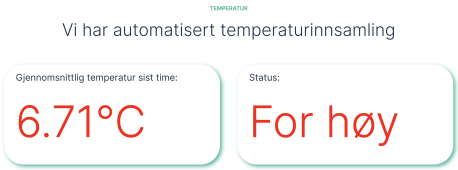
En API er blitt utviklet i NodeJS som henter alle temperaturmålinger fra sist time, som vil bli de siste 360 temperaturverdiene. Utregningen av dette vises i likning 1. Videre regner API-en gjennomsnittet av disse 360 verdiene, og sender ut det gjennomsnittet (se vedlegg A). Gjennomsnittet sendes ut for å gi en bedre indikasjon på om medikamentene er skadet, av samme årsak som det brukes i grafseksjonen av nettsiden. Så ber en funksjon i front-end siden om det gjennomsnittet. Denne funksjonen blir deretter brukt til å vise gjennomsnittstemperaturen, samt temperaturstatusen (se vedlegg A). Temperaturstatusen vil endre seg ut ifra om gjennomsnittstemperaturen er under 2 grader, mellom 2 og 6 grader og over 6 grader. I tillegg vil fargen på gjennomsnittstemperaturen endres ut ifra temperaturstatusen, da vil det være:

- Temperaturstatus 'For lav' (under 2 grader): blå farge
- Temperaturstatus 'Ok' (2 til 6 grader): grønn farge
- Temperaturstatus 'For høy' (over 6 grader): rød farge

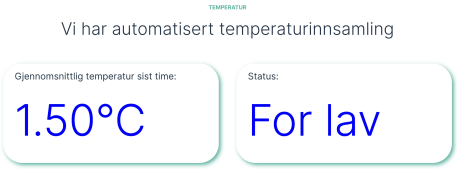
De forskjellige fargene og temperaturstatusene vises under i figur 13, 14 og 15.



Figur 13: Gjennomsnittlig temperatur sist time og temperaturstatus når temperaturen er passe (2 til 6 grader).



Figur 14: Gjennomsnittlig temperatur sist time og temperaturstatus når temperaturen er for høy.



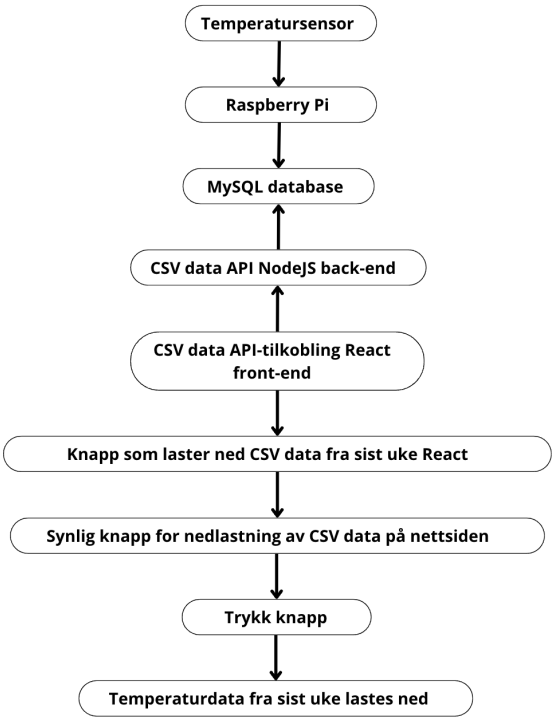
Figur 15: Gjennomsnittlig temperatur sist time og temperaturstatus når temperaturen er for lav.

Grunnen til at fargen rød er blitt benyttet når temperaturen er for høy, er på grunn av at fargen rød er assosiert med varme. Videre er fargen blå blitt benyttet når temperaturen er for lav, på grunn av at fargen blå er assosiert med kulde.

Kort sagt vil konfigurasjonen av gjennomsnittlig temperatur hjelpe til med systemkrav 1.2 og oppfylle systemkrav 1.4 i tabell 2, siden det viser gjennomsnittlig temperatur sist time, og fargelegger temperaturen rødt, blått og grønt, ut ifra om det er for varmt, for kaldt eller passe (2 til 6 grader).

CSV-nedlastning

Videre skal oppførselen til CSV-nedlasting gjennomgås.



Figur 16: Diagram av kommunikasjon mellom temperatursensor og nettside spesifikt for CSV-nedlasting.

En API er blitt utviklet i NodeJS som henter all temperaturdata fra sist uke, som vil bli 60480 temperaturmålinger (se vedlegg A). Utregningen av dette er vist under i likning 3.

$$\begin{aligned}\text{Antall målinger per uke} &= \frac{\text{Antall sekund per uke}}{\text{Antall sekund per måling}} \\ &= \frac{\text{Minutt per time} \times \text{Sekund per minutt} \times \text{Timer per dag} \times \text{Dager per uke}}{\text{Antall sekund per måling}} \\ &= \frac{604800}{10} \\ &= 60480 \end{aligned} \tag{3}$$

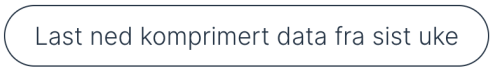
Deretter splittes de 60480 målinger i 7 for å finne antall målinger hver dag. Dette vil bli 8640 temperaturmålinger (se vedlegg A), som er blitt regnet ut under i likning 4.

$$\begin{aligned}\text{Antall målinger per dag} &= \frac{\text{Antall målinger per uke}}{\text{Antall dager i uken}} \\ &= \frac{60480}{7} \\ &= 8640 \end{aligned} \tag{4}$$

Videre vil API-en gjennomføre flere utregninger med hvert intervall på 8640 temperaturmålinger. De utregningene er:

1. Gjennomsnittstemperatur per dag.
2. Antall minutt med for høy temperatur per dag.
3. Antall minutt med for lav temperatur per dag.
4. Maksimal temperatur per dag.
5. Minimum temperatur per dag.

Så vil front-end siden av nettsiden be om disse verdiene fra API-en. Når verdiene ligger i front-end siden blir de formattert i en CSV fil, som kan lastes ned ved å trykke på en knapp på nettsiden (se vedlegg A). Under vises knappen i figur 17 og et eksempel av temperaturtabellen i figur 18.



Figur 17: Knapp på nettsiden man kan laste ned ukesrapporter fra.

komprimertData

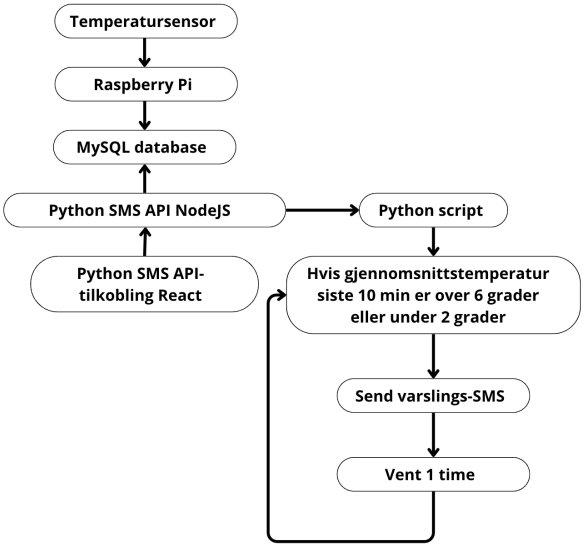
Dato	Gjennomsnittstemperatur	Tid med for lav temperatur (min)	Tid med for høy temperatur (min)	Maks temperatur	Minimum temperatur
Sat Apr 06 2024	4.15	0.00	0.00	4.5	3.8
Fri Apr 05 2024	3.30	222.00	0.00	5.2	1.4
Thu Apr 04 2024	4.07	0.00	0.00	5.9	2.2
Wed Apr 03 2024	3.69	167.00	0.00	5.9	1.5
Tue Apr 02 2024	5.11	0.00	227.50	6.4	3.8
Mon Apr 01 2024	6.02	0.00	733.00	7.4	4.6
Sat Mar 30 2024	5.52	0.00	489.33	7	4

Figur 18: Eksempel-ukesrapport.

Kort sagt viser figur 18 at systemkrav 1.3 i tabell 2 er oppfylt, siden det kan lagres ukesrapporter med:

1. Gjennomsnittstemperatur per dag.
2. Tid med for lav temperatur i minutter.
3. Tid med for høy temperatur i minutter.
4. Maksimal temperatur.
5. Minimal temperatur.

SMS-varsling
Så skal oppførselen til SMS-varsling gjennomgås.



Figur 19: Diagram av kommunikasjon mellom temperatursensor og nettside spesifikt for SMS-varsling.

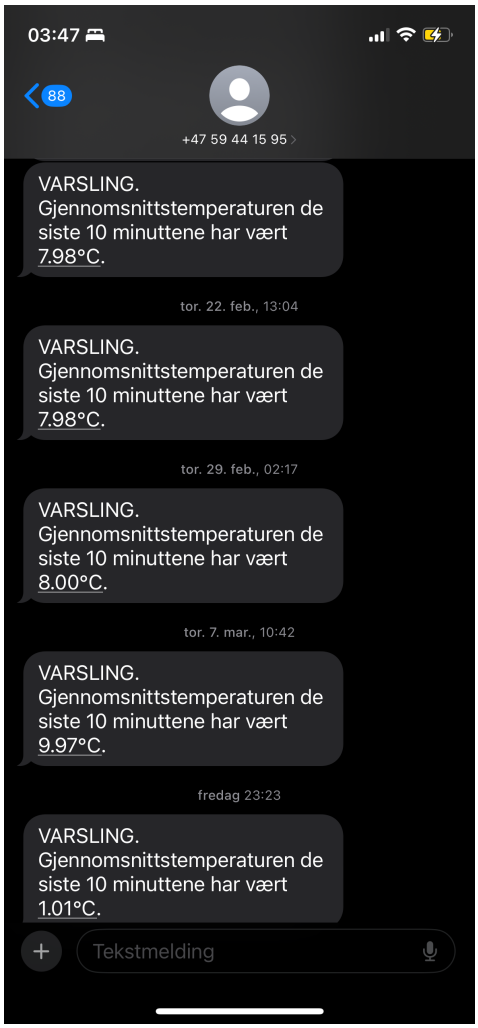
I NodeJS back-end siden er en API blitt utviklet, som kaller til et Python-script. Dette Python-scriptet vil hente de siste 10 minutters temperaturdata fra databasen. Dette vil totalt si 60 temperaturmålinger. Denne utregningen er vist under i likning 5.

$$\begin{aligned} \text{Antall målinger i 10 minutter} &= \frac{\text{Antall sekund i 10 minutt}}{\text{Antall sekund per måling}} \\ &= \frac{10 \times 60}{10} \\ &= 60 \end{aligned} \tag{5}$$

Deretter vil en SMS-varsling bli sendt til brukeren om gjennomsnittstemperaturen er over 6 grader eller under 2 grader (se vedlegg A).

Front-end siden utviklet med React skal kalle til API-en, slik at Python-scriptet kjører når nettsiden er oppe. Om en SMS-varsling blir sendt, skal Python-scriptet vente 1 time med å sende en ny eventuell SMS-varsling (se vedlegg A). Dette forhindrer SMS-spam til brukeren.

Eksempler av SMS-varslinger er vist under i figur 20.



Figur 20: Eksempel på SMS-varslinger som ansatte på poliklinikken vil motta, når temperaturen i kjøleskapet er utenfor 2 til 6 grader.

I utviklingsfasen sendes SMS-varslinger bare til 1 telefonnummer, men om denne løsningen skulle realiseres ville vi lagt inn flere telefonnumre, slik at flere mottar SMS-varslinger. Vi kom frem til denne beslutningen etter kommunikasjon med farmasøyten Gunn Lund, som sa: «Det er viktig at tilstrekkelig mange ansatte mottar disse varslene, slik at det alltid er noen til stede som kan handle raskt dersom det er nødvendig å redde legemidlene. Hvis kun en eller noen få personer mottar varsler, er det risiko for at de ikke oppdager situasjonen i tide, noe som kan få økonomiske konsekvenser.» (G. Lund, farmasøyt, personlig kommunikasjon, 27.03.2024).

Kort sagt vil denne konfigurasjonen oppfylle systemkrav 1.1 i tabell 2, siden systemkrav 1.1 ber om at systemet sender en varsling til brukeren om temperaturen i kjøleskapet er under 2 grader eller over 6 grader.

Fargepsykologi

Som nevnt tidligere i design delen er fargen grønn benyttet så mye som mulig på nettsiden,

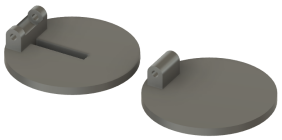
på grunn av fargepsykologi. Eksempler av implementering av denne fargebruken på nettsiden vises under i figur 21, samt tidligere figurer som 11 og 13, der fargen grønn er blitt benyttet så mye som mulig.

Figur 21: Eksempel av bruk av fargepsykologi i nettsideutvikling.

4.2 Design av etui til brytere

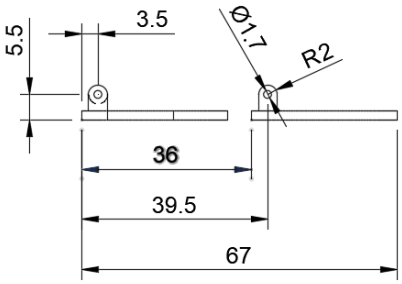
Bryterene i prosjektet blir brukt til å måle om et medikament står i et stativ eller ikke. Ved innkjøp, ble det registert at det var vanskelig å oppdrive brytere spesielt egnet til formålet da bryterene bør ha en stor kontaktflate mot hetteglasset. Det var essensielt at bryteren var lett å utløse da hetteglassene ikke veier spesielt mye. For å løse dette problemet, ble bryterene Zippy micro switch DF-series Type A [11] benyttet. Disse bryterene ble valgt da de var lett utløselige, var rimelige og lette å oppdrive.

Ved å 3D-printe et etui rundt bryteren, fikk bryteren større kontaktflate mot hetteglassene. Bryteren fikk også en lenger arm slik at kraftmomentet [12] mot bryteren blir større når et medikament settes ned på etuiet. 3D-modellen av det ferdige designet av etuiet kan sees i figur 22.

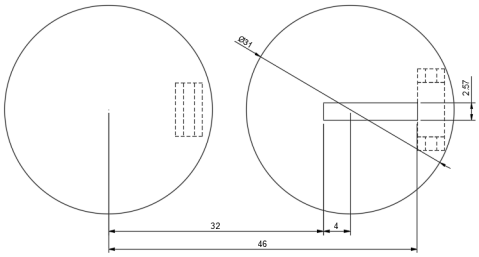


Figur 22: 3D-modell av etui-designet.

Figur 23: Bilde av bryteren benyttet. Bildet er hentet fra [11].

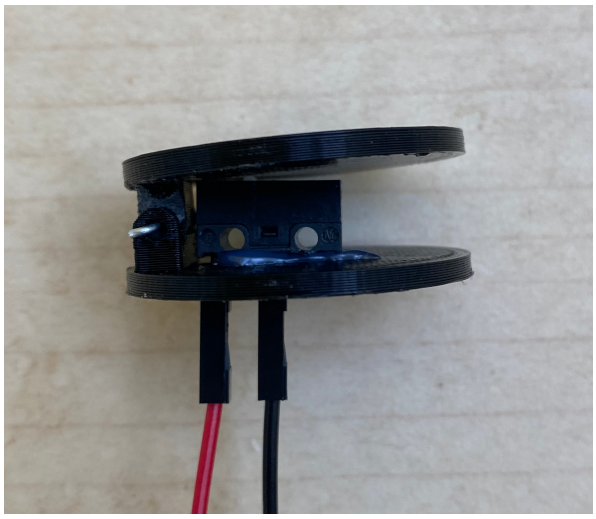


Figur 24: Mål av etuiet fra siden.



Figur 25: Mål av etui fra toppen.

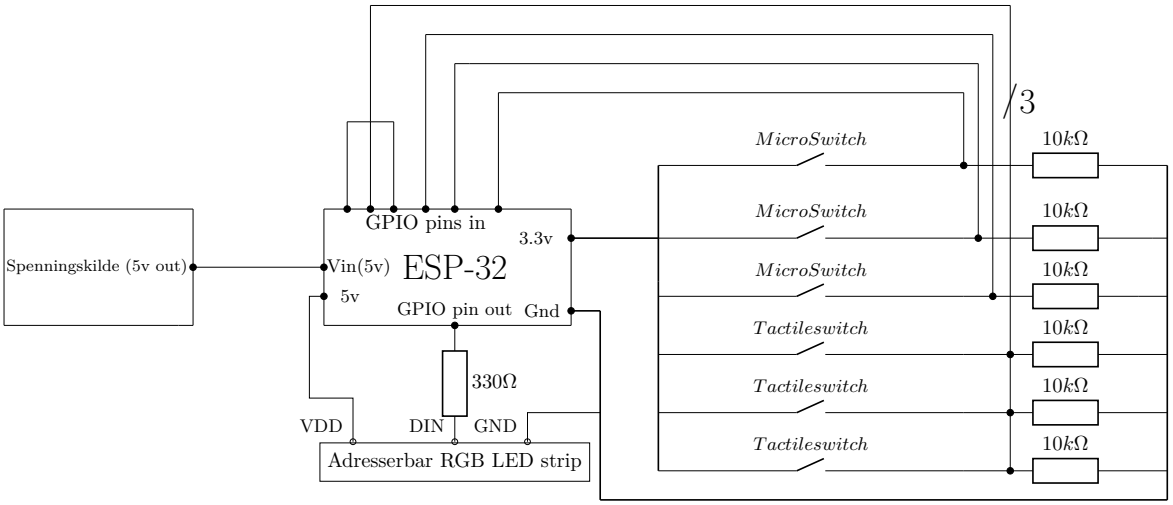
Når etuiet var ferdig printet, ble bryterene limt fast til etuiet med limpistol. Hullet på undersiden av etuiet er dimensjonert slik at vanlige jumper-kabler sitter fast til brytere uten at det trengs å limes. limpistol-lim ble valgt for at bryterene skal sitte fast, men fortsatt kunne fjernes og byttes ut ved eventuelle feil. En bit med ståltrå ble brukt for å holde hengselen på bryteren sammen. Bilde av det ferdige designet av etui med bryter og ledninger kan sees i figur 26.



Figur 26: Bilde av 3D-printet etui med bryter og tilkoplede ledninger.

4.3 Tidsstativ

For å implementere stativet for åpnede medikamenter, kalt *Tidsstativ* ble følgende krets designet:

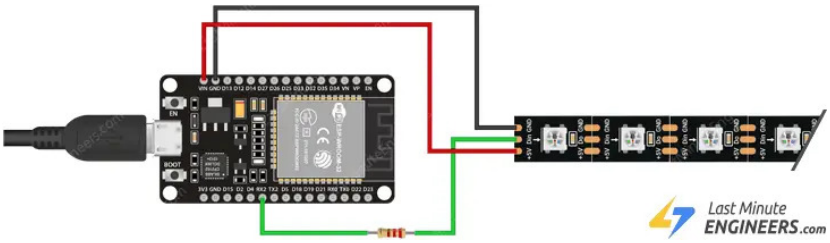


Figur 27: Kretsskjema for tidsstativ.

I figur 27 er mikrokontrolleren ESP-32 brukt til å styre logikken i systemet. Den får strøm fra en 5V spenningsskilde som er tilkoblet via micro-USB. ESP-32 er tilkoblet 3 brytere av typen MicroSwitch [11] og 3 Tactile Switches [13]. Inngangen til alle bryterne er forsynt med 3,3V fra mikro-kontrollerens 3,3V utgang. Utgangen fra hver bryter er koblet til en separat GPIO input pinne på ESP-32, sammen med en nedtrekksmotstand for å unngå flytende tilstander. Nedtrekksmotstandene er valgt til 10kΩ for å begrense strømforbruket, samtidig som man unngår for stort spenningsfall over motstanden. Dette er viktig for å unngå feilaktig lesing av

logiske nivåer på GPIO input pinnene [14]. GPIO pinnene leser dermed logisk høye signaler dersom den tilhørende bryteren er trykket ned og logisk lave signaler ellers.

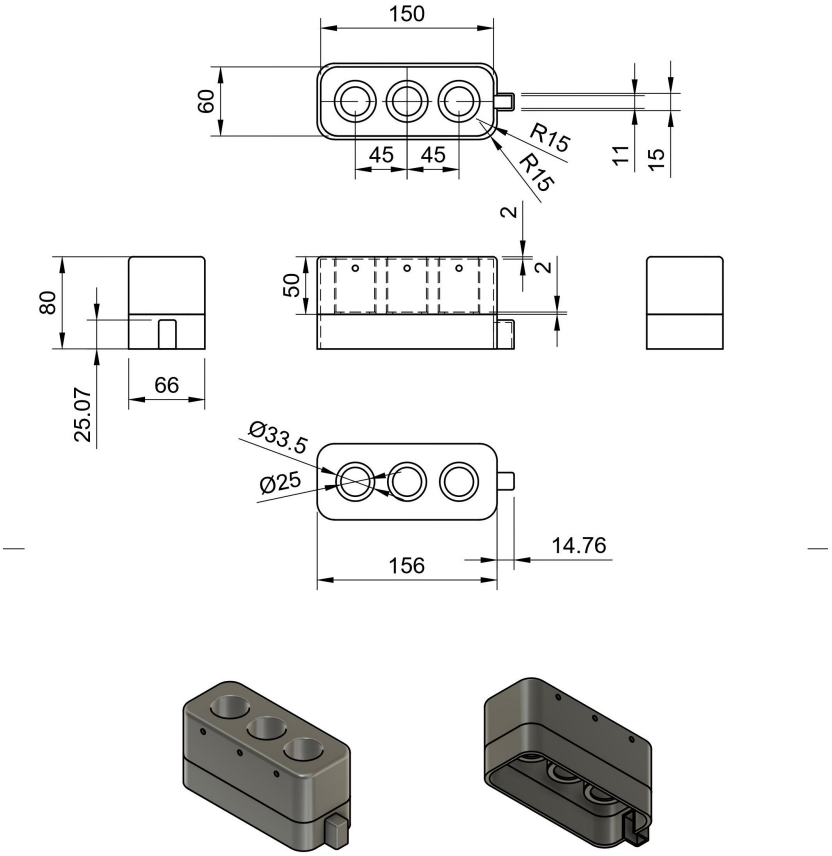
ESP-en er også tilkoblet til en adresserbar RGB LED-lenke, WS2812B [15] som består av 3 dioder. Lenken er plassbesparende og lyser kraftigere enn tradisjonelle LED-lys, som gjør designet lettere og indikasjonen tydeligere. LED-lenken er koblet til via 5v VDD, Gnd og en GPIO output pinne. Denne tilkoblingen er vist tydeligere i figur 28.



Figur 28: Kobling mellom ESP-32 og adresserbar RGB LED lenke hentet fra [16].

Figuren viser at den adresserbare lyslenken har tre tilkobligspunkter 5V (rød), GND (svart) og DIN (grønn). 5v og GND inngangene kobles direkte til mikrokontrollerens 5v og GND pinner. Mellom data inngangen (DIN) og ESP-ens GPIO-pinne, er det lagt inn en motstand på 330Ω for å beskytte data pinnen [16].

Designet av tidsstativet et vist i figur 29.



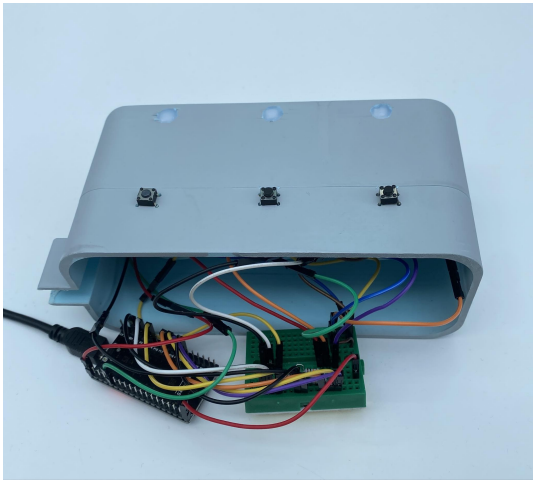
Figur 29: Design av tidsstativ, målene er oppgitt i millimeter

Stativet er utformet slik at beholderene skal ha plass til trykkknappene med etui (se figur [26]) i bunnen, og hetteglassene oppå. Beholderene er utstyrt med hull i bunnen, for å trekke ledningene fra bryterene gjennom. I tillegg er det integrert hull for diodene i den adresserbare LED-lenka. Diodene vil være rettet fremover slik at det er lett å identifisere hvilke av hetteglassene som har stått åpne for lenge.

For å skjule elektronikken best mulig i henhold til systemkrav 2.4, er det lagt til en støtteboks som hever stativet med 3 cm slik at elektronikken kan plasseres under. Dette gir stativet et ryddig utseende og gjør det mindre skremmende å bruke for personalet på poliklinikken.

Utformingen og oppkoblingen av det fysiske tidsstativet er vist i figur 30 og 31.

Figur 30: Fysisk utforming av stativet vist i figur 29.



Figur 31: Fysisk oppkobling av kretsen vist i figur 27 og 28.

I tillegg til den tidligere beskrevne utformingen, er det lagt til et silikonbelegg over diodene, og stativet er malt grått. Dette belegget holder LED-lenka på plass og bidrar til å gi stativet et mer nøytralt design for å oppfylle systemkrav 2.4. I tillegg er det boret hull til Tactile Swtich-knappene som tilbakestiller de respektive beholderene. Dette gjør det enklere for personalet på poliklinikken å se, og knytte knappene til hver enkelt beholder. I implementasjonen blir systemet forsynt av en batteribank. Siden dette begrenser levetiden til systemet, er det tenkt at en markedsklar versjon skal kunne forsynes via nettstrøm gjennom en kabel som trekkes gjennom kjøleskapet.

Logikken som er illustrert i figur 6 for hver av beholderene er anvendt i en evige løkke som består av 5 deler på ESP-32-en. Disse 5 delene gjentar seg for hver av de respektive beholderene. Det blir derfor kun vist kodeeksempel for den første beholderen. Den første delen er vist i kodeblokk 10. En oversikt over hvilke elementer som tilhører hver beholder er vist i figur 32.

```
1  //Registerer når knappen blir trykket ned og setter lyest til grønt.
2    if (digitalRead(knapp_Pin1) == HIGH and start_time1 == 0) {
3      start_time1 = millis(); // Registrerer starttidspunktet
4      leds[0] = CRGB::Green; // Setter LED til grønt
5      FastLED.show();  // Oppdaterer lyslenken
6    }
```

Kodeblokk 10: Registrering av knappetrykk (hopp fra tilstand 1 til tilstand 2)

Koden benytter mikrokontrollerens *digitalRead*-funksjon for å lese den logiske verdien til beholderens *knapp_Pin*. Dersom denne er logisk høy og *start_time* er lik 0 (som indikerer at ingen tidligere knappetrykk er registrert), settes *start_time* til det daværende tidspunktet ved hjelp av *millis()*-funksjonen. Deretter blir LED-en satt til grønt, og lyslenken oppdateres for å vise endringen.

```
//Sjekker om de respektive medikamentene har overgått holdbarheten eller ikke.
  currentMillis = millis(); // henter gjeldene tid fra mikrokontrolleren

  // Sjekk om millis() har rullet over
if (currentMillis < start_time1) {
    // Rullering har skjedd
    // Beregn tiden siden start_time1 ved å legge til den maksimale verdien til unsigned long
    unsigned long elapsedTime = (ULONG_MAX - start_time1) + currentMillis;
    //Sjekker om tiden siden start_time1 har overskredet holdbarheten
    if (elapsedTime >= duration) {
        leds[0] = CRGB::Red; // Setter LED til rødt
        FastLED.show(); // Oppdaterer lyslenken
    }
} else {
    // Ingen rullering
    //Sjekker om tiden siden start_time1 har overskredet holdbarheten
    if (currentMillis - start_time1 >= duration and start_time1 > 0) {
        leds[0] = CRGB::Red; //Setter LED til rødt
        FastLED.show(); //Oppdaterer lyslenken
    }
}
```

Kodeblokk 11: Sjekk av holdbarhet

Kodeblokk 11 viser del 2 av den uendelige løkka. Koden starter med å sette den nåværende tiden *currentMillis* ved hjelp av *millis()* funksjonen til mikrokontrolleren. *millis()* er en funksjon som henter ut antall millisekunder det har gått siden oppstart i form av en unsigned long variabel [17]. Ulempen med dette er at unsigned long elementer har en maksimalverdi på $2 \exp 32 - 1 = 4,294,967,295$ [18]. Dette gjør at *millis()* vil begynne på 0 igjen etter omtrent 49 dager [17]. Dersom et åpent hetteglass blir plassert i beholderen på for eksempel dag 48 etter oppstart av mikrokontrolleren vil det oppstå problemer. Da vil *currentMillis* aldri nå en verdi som er 3 dager større (i millisekunder), og systemet vil aldri nå den tredje tilstanden 6. For å unngå dette problemet sjekker koden om millis funksjonen har rullet over ved å se om *currentMillis* er mindre enn *start_time*. Dersom dette er tilfellet, vil mikrokontrolleren beregne tiden som har gått siden *start_time* ved å ta maksimalverdien til et unsigned long element, trekke fra *start_time* og legge til *currentMillis*. Dersom denne tiden er like lang eller lenger enn en gitt mengde millisekunder (*duration*), vil beholderens LED bli satt til å lyse rødt.

Dersom *millis()* ikke har rullet over, sjekker koden om tiden som har gått (*currentMillis* - *start_time*) er større eller lik *duration* gitt at *start_time* er større en null. Dersom dette er tilfelle, vil beholderens LED også lyse rødt. Hvis tiden som har gått siden *start_time* er mindre enn *duration*, vil ingenting skje.

Den tredje delen av den evige løkka er vist i kodeblokk 12.

```
1
2  // Gradvis fargeendring frem til holdbarheten går ut.
3  if (start_time1 > millis()) {
4      // Millis har rullet over
5      unsigned long elapsedTime = (ULONG_MAX - start_time1) + millis();
6      if (elapsedTime < duration) {
7          // Beregner forholdet mellom elapsedTime og duration for gradvis fargeendring
8          float proportion = (float)elapsedTime / (float)duration;
9          // Beregner indekser for blått og grønt lys basert på forholdet
10         int blueIndex = proportion * 255;
11         int greenIndex = 255 - blueIndex;
12         // Oppdaterer LED-lyset med gradvis fargeendring
13         leds[0] = CRGB(0, greenIndex, blueIndex);
14         FastLED.show();
15     }
16 } else {
17     // Millis har ikke rullet over
18     unsigned long elapsedTime = (millis() - start_time1);
19     if (elapsedTime < duration) {
20         // Beregner forholdet mellom elapsedTime og duration for gradvis fargeendring
21         float proportion = (float)elapsedTime / (float)duration;
22         // Beregner indekser for blått og grønt lys basert på forholdet
23         int blueIndex = proportion * 255;
24         int greenIndex = 255 - blueIndex;
25         // Oppdaterer LED-lyset med gradvis fargeendring
26         leds[0] = CRGB(0, greenIndex, blueIndex);
27         FastLED.show();
28     }
29 }
```

Kodeblokk 12: Gradvis fargeendring frem til holdbarheten går ut.

I Kodeblokk 12 håndteres gradvis fargeendring på LED-lyset basert på hvor lenge det har gått siden *knapp_Pin* ble aktivert. Koden begynner med å sjekke om *millis()* har rullet over eller ikke, og beregner deretter *elapsedTime* basert på dette. Deretter sjekkes det om *elapsedTime* er mindre enn holdbarheten *duration*. Hvis dette er tilfelle beregnes forholdet mellom *elapsedTime* og *duration* for å oppnå gradvis fargeendring. Dette forholdet blir benyttet for å beregne variabelen *blueindex* som angir mengden blått lys LED-en skal emmitere. Etterhvert som forholdet blir større vil *blueindex* bli større, samtidig som mengden grønt lys *green index* blir mindre. Dette resulterer i at beholderen vil lyse gradvis mer blått jo *elapsedTime* kommer *duration*. Dette gjør at personalet i tillegg til å enkelt kunne skille mellom legemidlene som må kastes og ikke, nå også kan identifisere medikamentet som har stått åpent lengst, og dermed få mest mulig ut av holdbarheten på de åpne medikamentene.

Del 4 av den evige løkken er vist i kodeblokk 13.

```
//Tilbakestiller diodene og start time.
 currentMillis = millis();

 // Sjekk om millis() har rullet over
if (currentMillis < start_time1) {
    // Rullering har skjedd
    // Beregn tiden siden start_time1 ved å legge til den maksimale verdien til unsigned long
    unsigned long elapsedTime = (ULONG_MAX - start_time1) + currentMillis;
      // Sjekker om tiden siden start_time1 har overskredet holdbarheten og knappen er trykket ned
    if (elapsedTime >= duration and digitalRead(knapp_Pin1)==LOW) {
        leds[0] = CRGB::Black; // Setter LED til å ikke lyse
        FastLED.show(); //Oppdaterer lyslenken
        start_time1 = 0; //Tilbakestiller start_time1
    }
} else {
    // Ingen rullering
    // Sjekker om tiden siden start_time1 har overskredet holdbarheten og knappen er trykket ned
    if ((currentMillis - start_time1) >= duration and digitalRead(knapp_Pin1)==LOW and and start_time1 >
    ↪  0) {
        leds[0] = CRGB::Black; // Setter LED til å ikke lyse
        FastLED.show(); // Oppdaterer lyslenken
        start_time1 = 0; // Tilbakestiller start_time1
    }
}
```

Kodeblokk 13: Automatisk tilbakestilling

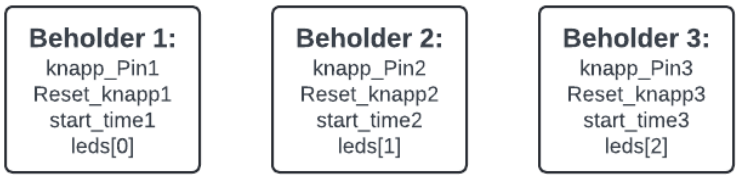
På samme måte som i kodeblokk 11, blir det tatt hensyn til om *millis()* har rullet over eller ikke. Dersom tiden som har gått er større en *duration*, og verdien på *knapp_pin* er logisk lav (som indikerer at hetteglasset har blitt tatt ut og kastet), vil beholderens LED settes til svart (ikke lys) og *start_time* settes til null. På den måten går systemet tilbake til første tilstand.

Den siste delen av den evige løkken lagt til i tilfelle brukerfeil.

```
//Manuell tilbakestilling:
if (digitalRead(Reset_knapp1) == HIGH){
    leds[0] = CRGB::Black; //Setter LED  til å ikke lyse
    FastLED.show();  // Oppdaterer lyslenken
    start_time1 = 0; // Tilbakestiller start_time1
  }
```

Kodeblokk 14: Manuell tilbakestilling

Kodeblokk 14 legger til muligheten for manuell tilbakestilling i tilfelle bryteren blir aktivert ved en feil, eller brukeren ønsker å avslutte oppbevaringen før tiden overstiger *duration*. Koden sjekker om beholderens *Reset_knapp* har logisk høy verdi. Dersom dette er tilfelle vil beholderes LED bli satt til svart (ikke lyse) og *start_time* blir satt til 0 slik at systemet er tilbake i første tilstand.



Figur 32: Oversikt over hvilke elementer i Kodeblokk som tilhører hver beholder.

knapp_pin og *Reset_knapp* er variabler som avhenger av om henholdsvis bryteren og tilbakestillingsknappen til den tilhørende beholderen er trykket ned eller ikke. *start_time* er en variabel som registrerer tidspunktet *Knapp_pin* får logisk høy verdi, og *leds[]* er beholderens tilhørende LED-enhet.

Den fullstendige koden er gitt i vedlegg A.

4.4 Implementasjon av roterende medikamentstativ for uåpnede medikamenter

4.4.1 Utforming

Designet av det roterende medikamentstativet er inspirert av skissen i figur 3. I utgangspunktet vil det å få to separate fritt roterende deler til å samhandle elektronisk være utfordrende uten en form for trådløs kommunikasjon. En aktuator fastspent i basen vil måtte hente informasjon fra medikamentflaske-sensorene på overdelen av det roterende hjulet. I tillegg er begge delene avhengig av en strømkilde, og en kablet løsning mellom de roterende delene vil fort gi designet større begrensninger enn ønskelig.

Dette ble løst ved å benytte et design der all elektronikken er montert i den øvre roterende delen av stativet. En aktuator festet i overdelen vil kunne kobles til et tannhjul på basen og dytte seg selv og hele toppdelen rundt uten å ha elektronisk kontakt med basen.

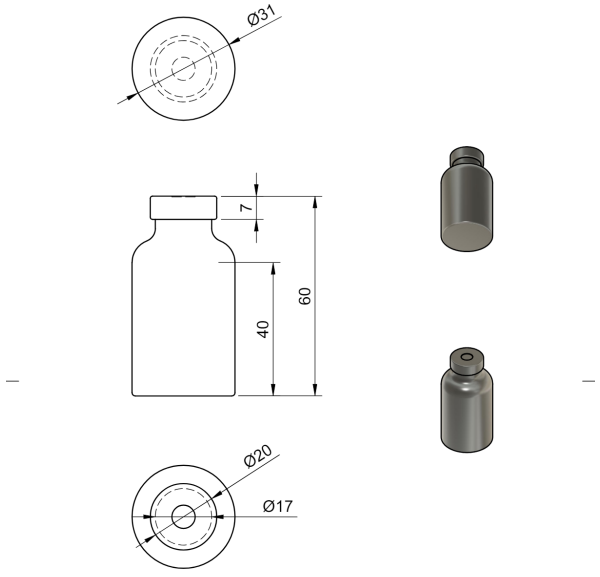
Det ble tidlig i implementeringsfasen moddelert og realisert en enklere og mindre prototype med 6 medikamentholdere. Denne prototypen ga mulighet for å utforske konseptet fysisk med muligheter for enkle designendringer og rom for prøving og feiling. I tillegg til få antall medikamentholdere, rommet ikke modellen nok plass til innmaten og den tilhørende elektronikken. Det ble dermed moddelert en større prototype med små designforbedringer og tilstrekkelig rom for alle de elektroniske komponentene.

Alle fysiske deler utenom elektronikken er designet i programvaren fusion 360. [1] En dekomponert modell av det komplette roterende stativet (første og andre prototype) er vist figur 33 og 34. Ved design av stativet var det viktig å gjøre gode målinger av elektronikk og hetteglass for å sørge for at alt passet fint inn på sin respektive plass ved montering.

[1] Fusion 360 er en programvare for design og 3D-modellering. Programvaren kan man finne her.

Figur 33: Fullstendig dekomponert modell av første prototype

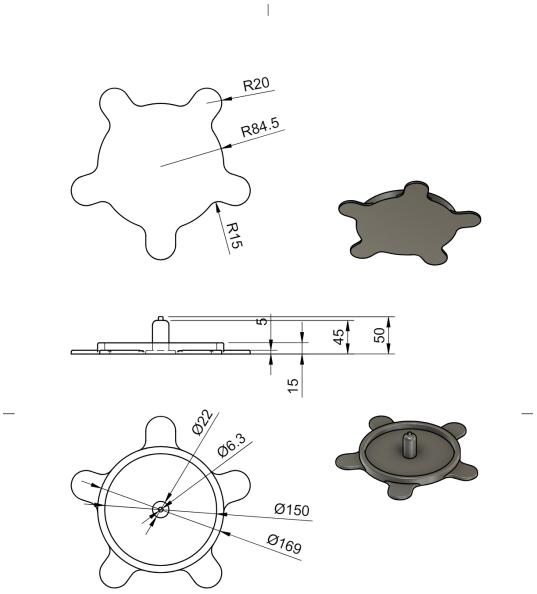
Figur 34: Fullstendig dekomponert modell av andre prototype.



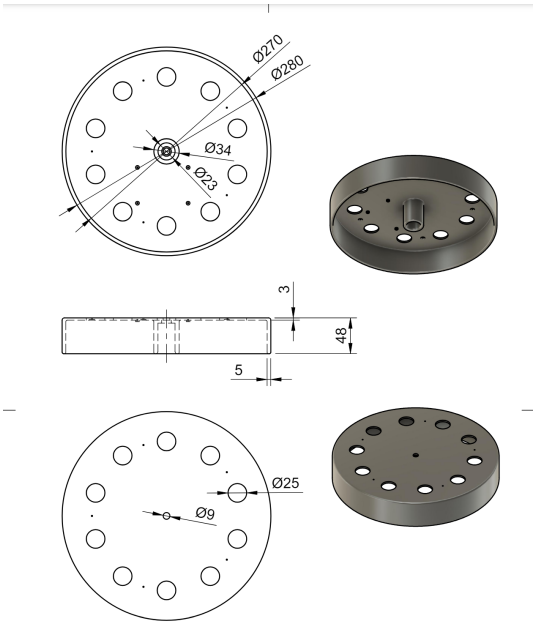
Figur 35: Medikamentflaske (hetteglass), med mål (mm)

Et eksemplar av det faktiske hetteglasset som skulle bli brukt, ble benyttet til remodellering av flaskene i fusion som vist i figur 35. Dermed kunne disse 3d-printes og bli brukt som test-medikamenter.

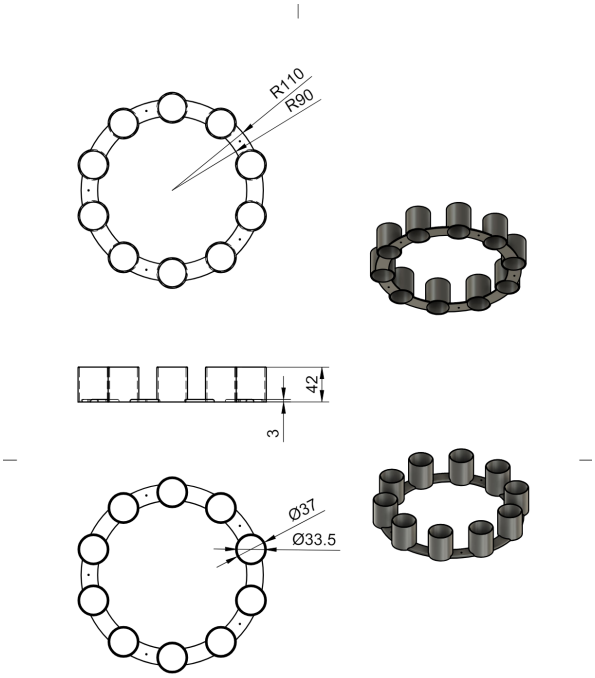
Det roterende medikamenthjulet er designet med 3 hoveddeler; basen, den roterende overdelen, og medikamentholderne, illustrert i figur 36, 37 og 38 henholdsvis.



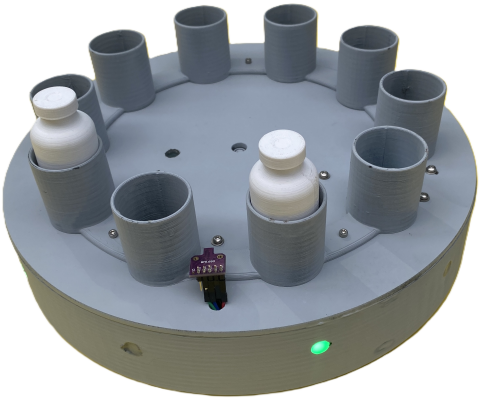
Figur 36: Medikamenthjul base, med mål (mm).



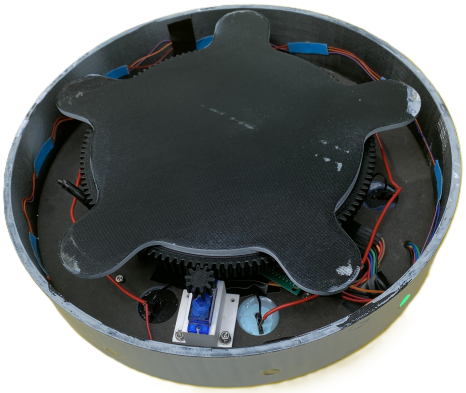
Figur 37: Medikamenthjul overdel, med mål (mm).



Figur 38: Medikamentholdere, med mål (mm)



Figur 39: Bilde av det realiserte medikament-stativet ovenfra etter printing og maling.

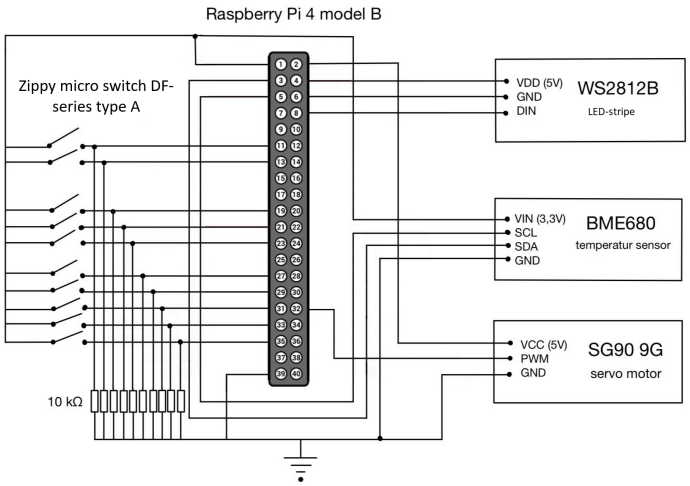


Figur 40: Bilde av det realiserte medikament-stativet undenfra.

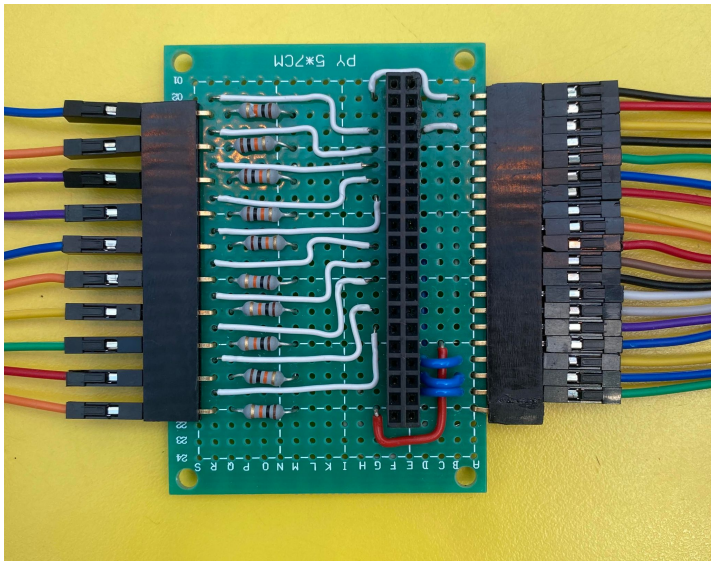
4.4.2 Elektronikk

På det roterende stativet benyttes mikro-kontrolleren Raspberry Pi 4 Model B [19] for all kommunikasjon med databasen i tillegg til å styre alle elektriske komponenter. Dette innebærer LED-lys, brytere, servo-motor og temperatursensor. LED-lysene benyttet til det roterende stativet, er identiske med de på det statiske stativet, og er av typen WS2812B. Servo-motoren benyttet har komponentnavn SG90 9G [20].I tillegg er det koplet opp en temperatur-sensor av typen BME680[7]. Sensoren kan sees i figur 39, og er koplet opp som vist i figur 41.

For å kople bryterene, led stripen, servomotoren og temperatursensoren til mikrokontrollen, ble det designet et kretskort på et eksperimentkort. For å sørge for at komponenter kan byttes ut, er alt festet med headere eller skruer. Pin-out til kretskortet kan sees i figur 41. Realiseringen av kretskortet og oppkopling til komponenter kan sees i figur 42 43.



Figur 41: Koplingsskjema til kretskortet designet til Raspberry Pi model 4B.

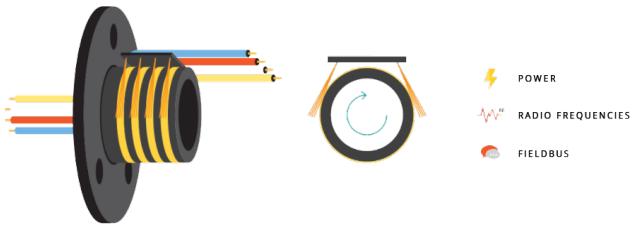


Figur 42: Realisering av kretskort for kopling mellom komponenter og mikrokontroller i det roterende stativet.

Figur 43: Bilde av kretskortet koplet opp til mikrokontroller og komponenter i medikamentstativet.

For å registrere om et medikament er plassert i en beholder på stativet benyttes bryterene på samme måte som beskrevet i 4.2. Slik man kan se av kretsskjemaet i figur 41, forsynes hver bryter av 3.3V spenningsforsyningen på mikrokontrolleren. Utgangen på hver bryter er igjen koplet på en GPIO-pin med en 10 kΩ nedtrekksmotstand koplet til jord.

I prototypen bygget for visning, er det benyttet en batteribank som spenningsforsyning. Dette er i utgangspunktet hverken en nødvendighet eller del av den videre planen for utvikling grunnet at det gir systemet begrenset batteritid. Ved videre utvikling er det lagt planer for innkjøp av en roterende kopling som vist i figur 44, slik at stativet får en kontinuerlig spenningsforsyning. En slik løsning er nødvendig fordi stativet roterer, og dermed ville en vanlig ledning tvinnet seg. Dette er til dags dato ikke kjøpt inn på grunn av lang leveringstid, og det er heller strengt tatt ikke nødvendig for å vise at systemet fungerer.



Figur 44: Illustrasjon av en roterende kopling eller slepering. Illustrasjonen er hentet fra [21].

4.4.3 Programvare

Koden består av en while løkke som roterer til riktig led lys basert på GPIO signalet fra knappene og lagret utløpsdato. Implementasjonen vises i kodeblokk 15. QR kode skanning som nevnt i subseksjonen 2.3 er ikke implementert av tidsmessige årsaker.

```
        while True:
            now = datetime.datetime.now()-START_TIME
            next_expired = 0
            for i in range(LED_COUNT):
                if(GPIO.input(buttons)): #button is released
                    strip.setPixelColor(i, Color(0,0,0))
                    continue
                if(dates[i] > now): #Not expired
                    strip.setPixelColor(i, Color(0,0,255))
                    if(dates[next_expired] > dates[i]):
                        next_expired = i
                else:
                    strip.setPixelColor(i, Color(255,0,0))
            rotate(next_expired)
            print ('Color wipe animations.')
```

Kodeblokk 15: En while løkke som er basert på designet fra 7.

5 Validering, verifikasjon og test

For å utføre verifikasjoner av systemet har vi gjennomgått flere aktiviteter. Den totale planen er vist i tabell 3.

Systemkrav	Aktivitet	Hvem	Hvor	Når	Godkjent
Temperatursensor					
1.1	Teste om automatisk SMS-varsel funker.	Arya, Henrik	Koopen	18.4.2024	✓
1.2	Teste om gjennomsnittstemperaturen og grafen endrer seg når nye temperaturmålinger kommer i MySQL databasen.	Arya	Koopen	18.4.2024	✓
1.3	Teste om ukesrapportene lagres i et fint format med riktig data.	Arya	Koopen	18.4.2024	✓
1.4	Se om teksten på temperaturen og temperaturstatusen endres til blått og rødt som følge av gjennomsnittstemperaturen, og ellers er grønt.	Arya	Koopen	18.4.2024	✓
1.5	Se om nye målinger oppstår i databasen hvert 10. sekund.	Asil, Henrik	Koopen	18.4.2024	✓
1.6	Tester temperaturavviket ved bruk av en normalfordeling.	Asil	Koopen	15.4.2024	✓
Stativ for åpne medikamenter					
2.1	Teste om det lyser rødt etter et tidsintervall på 3 dager, og teste presisjon i korte intervaller.	Jon Arne	Koopen	15.4.2024	✓
2.2	Fysisk test av funksjonaliteten til stativet	Jon Arne	Koopen	15.4.2024	delvis
2.3	Måling med linjal.	Jon Arne	Koopen	15.4.2024	✓
2.4	Visuell testing.	Jon Arne	Koopen	15.4.2024	✓
Stativ for uåpnede medikamenter					
3.1	Måling med linjal.	Elias, Sven	Koopen	11.4.2024	✓

3.2	Se om lyset endrer seg etter et tidsintervall, tester med spesifisert dato.	Elias, Sven, Asil	Koopen	18.4.2024	✓
3.3	Se om medikamentet med kortest utløpsdato kommer fremst i stativet.	Elias, Sven, Asil	Koopen	18.4.2024	✓
3.4	Visuell testing.	Elias, Sven, Asil	Koopen	18.4.2024	✓

Tabell 3: Verifikasjonplan basert på systemkrav fra tabell 2.

5.1 Verifikasjon av temperatursensor

5.1.1 Verifikasjon av temperaturmåling

Først og fremst testet vi om nye temperaturmålinger sendes til databasen hvert 10. sekund ved bruk av 2 metoder. Først sjekket vi grafen på nettsiden, og så at grafen endret seg hvert 10. sekund, som følge av nye målinger i databasen. For å være sikker sjekket vi selve databasen også, der det oppsto nye målinger hvert 10. sekund. Derfor godkjennes systemkrav 1.5 i tabell 3.

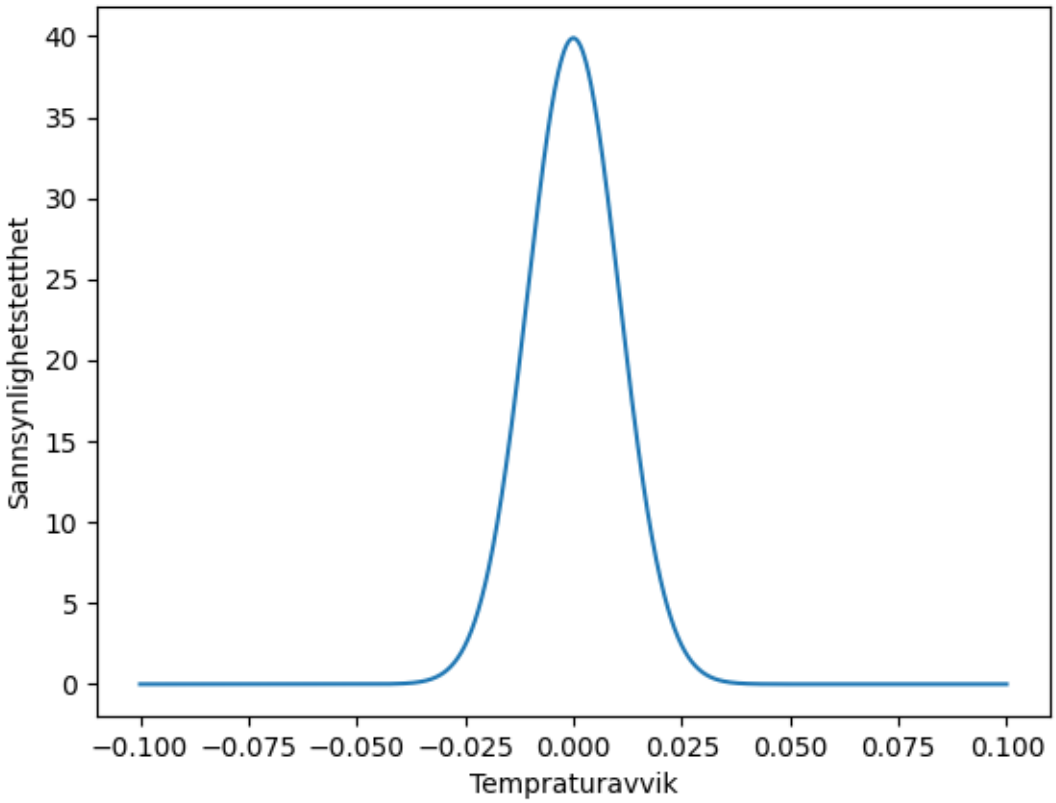
Videre måler ikke tempratursensoren temperaturen med full nøyaktighet. Målingene til temperatursensoren er uniformt fordelt med følgende parametere

$$\begin{aligned} b &= T + 1 \\ a &= T - 1 \\ \mu &= \frac{b+a}{2} \\ \sigma^2 &= \frac{(b-a)^2}{12} \end{aligned} \tag{6}$$

Siden vi tar 3000 målinger per datapunkt er det mulig å tilnærme sannsynlighetsfordelingen for temperaturmålingene til en normalfordeling ved bruk av sentralgrenseteoremet.

$$Z = \frac{\bar{X} - \mu}{\sqrt{\frac{\sigma^2}{n}}} = N(0,1) \tag{7}$$

Figur 45 viser sannsynligheten for et temperaturavvik. Ved bruk av grafisk avlesning kan det observeres at sannsynligheten for et temperaturavvik som er større enn 0,025 grader er ekstremt liten. Siden temperaturen i et kjøleskap er oftest mellom 2 og 6 grader, er temperaturavviket på 0,025 grader neglisjerbar.



Figur 45: Sannsynligheten for temperaturavvik

Med tanke på det neglisjerbare avviket, godkjennes systemkrav 1.6 i tabell 3.

5.1.2 Verifikasjon av nettside

For verifisering av nettsiden må systemkrav 1.1 til 1.4 fra tabell 2 verifiseres.

For å teste om brukeren mottar varslinger når gjennomsnittstemperaturen de siste 10 minuttene blikker over 6 grader eller går under 2 grader, ble en fysisk gjennomføring av prosessen utført. Først ble det sendt inn falsk temperaturdata til MySQL databasen, som lå utenfor det gunstige området (2 til 6 grader). Deretter gikk vi inn på nettsiden, og mottok en varsling tilsvarende SMS-ene fra figur 20. En problemstilling her er at varslingene sendes bare når nettsiden er oppe, som realistisk sett ikke alltid vil være tilfellet. Kort sagt funker varslingssystemet, derfor godkjennes systemkrav 1.1 i tabell 3, men samtidig er det rom for forbedring.

Videre testes det om gjennomsnittstemperaturen og grafen endres når nye målinger kommer i MySQL databasen, ved å la temperatursensoren sende målinger til databasen, og se om nettsiden korrigeres ut ifra de nye målingene. Her viste det seg å funke helt fint, derfor

godkjennes systemkrav 1.2 i tabell 3.

Deretter testes ukesrapportene ved visuell testing av om de lagres i et fint format med riktig data. Testene viste at temperaturen lagres i et fint format (se figur 18) og all data fra sist uke vises i tabellen. En problemstilling her er at koden som henter ut temperaturdata for tabellen, henter ut nøyaktig de siste 60480 temperaturverdiene. Dette medfører at tabellen vil bestå av temperaturdata fra nøyaktig det tidspunktet du er i, og til 1 uke bak. For eksempel om du laster ned tabellen en mandag klokken 14, vil tabellen bestå av all temperaturdata fra sist uke, til og med mandag klokken 14 uken før. Med andre ord vil temperaturdataene før klokken 14 uken før ikke være med i tabellen, som fører til falske tall til mandagen uken før. Kort sagt funker tabellen stort sett fint, og derfor godkjennes systemkrav 1.3 i tabell 3, selv om det er rom for forbedring.

Til slutt testes det om teksten på temperaturen og temperaturstatusen endrer seg som følge av temperaturendringer. Vi utførte denne testen ved å fylle MySQL databasen med falsk temperaturdata, og observerte på nettsiden at fargen ble rød når gjennomsnittstemperaturen sist time var over 6 grader, blå når det lå under 2 grader og grønn ellers. På grunn av disse resultatene godkjennes systemkrav 1.4 i tabell 3.

5.2 Verifikasjon av stativ for åpnede medikamenter

For å verifisere systemkrav 2.1 ble stativet for åpnede medikamenter testet ved å plassere 3D printede hetteglass i beholderene og sette på en alarm 3 dager frem i tid. Når alarmen gikk, ble tidstativet sjekket, og alle LED-ene til de respektive beholderene lyste rødt. I tillegg ble systemets nøyaktighet testet gjentatte ganger med 5,10 og 30 sekunders intervaller, ved hjelp av en stoppeklokke. Resultatene til de forskjellige tidsintervallene er vist i tabell 4.

Tidsintervall (sekunder)	Måling 1 (sekunder)	Måling 2 (sekunder)	Måling 3 (sekunder)
5	5.19	5.35	5.13
10	10.28	10.21	10.15
30	30.26	30.22	30.27

Tabell 4: Målinger av tiden det tok før beholderenes LED begynte å lyse rødt med forskjellige tidsintervaller.

Tabell 4 viser at avviket fra det opprinnelige tidsintervallet er rundt 0.25 sekunder, uavhengig av lengden på tidsintervallet. En del av dette avviket kommer trolig av reaksjonstiden, ettersom resultatene er avhengig av hvor raskt testgjennomføreren stopper klokka etter å ha oppdaget at beholderene lyser rødt. Basert på dette, samt det faktum at det tiltenkte tidsintervallet er på 3 dager og at det dermed ikke kritisk at nøyaktigheten har noen millisekunders avvik, er det rimelig å konkludere med at systemkrav 2.1 er oppfylt.

Systemkrav 2.2 ble evaluert ved å plassere 3D printede hetteglass i beholderene til stativet. Etter hetteglassene ble plassert, kunne de tas ut og inn så mange ganger som ønsket før tiden utløp og lyset ble rødt. Etter å ha fjernet hetteglassene fra beholderen når tiden hadde utløpt, ble systemet automatisk tilbakestilt. Imidlertid opplevdes tidvis dårlig kontakt mellom noen av ledningen i oppkoblingen, noe som kunne føre til udefinert oppførsel. Videre var det ikke

forberedt for potensielle strømbrudd i systemet, slik at da strømforsyningen ble koblet fra og til igjen, startet systemet i en udefinert tilstand. Man måtte derfor starte en ny nedtelling ved å plassere hetteglass i beholderene. Hvis systemet skal tas i bruk er dette nødt til å utbedres. Systemkrav 2.2 er dermed kun delvis oppfylt.

Tidsstativet ble målt til en lengde på 16.5cm, en høyde inklusiv hetteglass på 10.4 cm og bredde på 6.2 cm. Dermed er systemkrav 2.3 oppfylt.

Stativet er malt i en nøytral lys grå farge og all elektronikken er skjult inni stativet. Dermed er systemkrav 2.4 oppfylt.

Det ble også sjekket og bekreftet at tilbakestillings-knappene og den gradvise fargeendringen fungerte som det skulle.

5.3 Verifikasjon av stativ for uåpnede medikamenter

Vi har benyttet oss av en verifikasjonsplan innen sorteringen av medikamenter, som er basert på systemkravene i (tabell 2) til løsningen. Først og fremst målte vi dimensjonene til løsningen. Diameteren lå på 28 cm og høyden lå på 12,5 cm. Dette er innenfor systemkrav 3.1 fra tabell 2. Med andre ord vil løsningen passe fint i kjøleskapet til medisinsk fysikalsk poliklinikk. En mulig feilkilde her er at vi bruker øyemål sammen med en linjal, men siden løsningen er langt innenfor systemkrav 3.1 fra tabell 2 (40 cm diameter, 30 cm høyde), har vi vurdert at denne feilkilden har lav sannsynlighet til å være utslagsgivende med tanke på den tilgjengelige plassen i kjøleskap. Derfor godkjennes systemkrav 3.1 i tabell 3.

Videre er det blitt testet om lyset endres etter et tidsintervall, der det testes med spesifiserte dato. Etter flere runder med testing viste det seg at systemet funker, og derfor godkjennes systemkrav 3.2 i tabell 3.

Til slutt er det blitt testet om medikamentet med kortest utløpsdato kommer fremst i stativet. Ved testing av dette er det blitt konkludert at systemet utfører dette kravet, og derfor godkjennes systemkrav 3.3 i tabell 3.

5.4 Validering

Brukerkrav	I hvilken grad...	Score (1-100)
A	passer systemet i kjøleskapet?	100
B	er det enkelt å bruke?	89.5
C	er det enkelt å få tak i medikamentet som har gått ut på dato først?	86.75
D	gir systemet informasjon om medikamentet har gått ut på dato?	94
E	gir systemet informasjon om medikamentet har stått åpent for lenge?	87
F	ser systemet enkelt og ryddig ut?	92
G	varsles brukeren dersom temperaturen i kjøleskapet faller under 2 grader eller stiger over 6 grader?	100
H	blir det enkelt å sjekke temperaturen i kjøleskapet?	97.8
I	lagres sist ukes temperaturdata i en tabell, der brukeren kan dokumentere at temperaturen har vært innenfor 2 til 6 grader eller ikke?	78
J	har systemet kontroll på alle medikamentene i beholdningen?	82

Tabell 5: Valideringsplan basert på brukerkrav fra tabell 1 (Score=100: perfekt, score=1: dårlig)

For å utføre valideringsplanen fra tabell 5 har vi spurt 5 personer fra helse-sektoren, og brukt tilbakemeldingen deres til å sette verdier fra 1 til 100 på valideringsplanene våres. 4 av personene var i Trondheim under spørreundersøkelsen, mens 1 sykepleierstudent var ikke fysisk tilgjengelig. Derfor fikk den ene sykepleierstudenten bare vurdert nettsiden, og ikke hele systemet. Under står det kommentarer angående hver valideringsplan.

Validering av brukerkrav A
Poliklinikken på St. Olavs Hospital benytter et ordinært kjøleskap for oppbevaring av medikamenter. Derfor testet vi systemet i et slikt kjøleskap, der det viste seg å passe med god margin. Med tanke på dette resultatet har vi satt score til 100 her.

Validering av brukerkrav B
For å validere om systemet er enkelt, viste vi det fysiske systemet til 4 personer innenfor helsebransjen, samt 1 sykepleiestudent som testet nettsiden. Alle ga oss tilbakemeldinger og score, der gjennomsnittsscoren ble 89.5, som er det vi satt i tabell 5.

Validering av brukerkrav C
Videre validerte vi om systemet gjør det enkelt å få tak i medikamentet som har gått ut på

dato først, ved å teste det med 4 personer innenfor helsebransjen. Vi fikk tilbakemeldinger og score fra de, der gjennomsnittsscoren ble 86.75. Dette satt vi i tabell 5.

Validering av brukerkrav D
Deretter validerte vi om systemet gir informasjon om medikamentene har gått ut på dato, ved å vise hele systemet til 4 personer innen helsebransjen, og nettsiden til en sykepleierstudent. Alle ga oss egne scores til valideringen av brukerkrav D, der gjennomsnittsscoren ble 94. Denne scoren satt vi inn på tabell 5.

Validering av brukerkrav E
Systemet gir deg informasjon når medikamentet har stått åpent for lenge. Et mulig problem her er om strømkoblingen til systemet ikke lengre kommer, som fører til at systemet slås av. Dette kan forhindres ved å benytte batterier som strømkilde, men vi valgte å bruke en vanlig strømkobling på grunn av tidsmessige årsaker. Med tanke på dette satt vi scoren til 87, siden brukerkrav E oppfylles, men har et mulig problem.

Validering av brukerkrav F
Så validerte vi om systemet ser enkelt og ryddig ut ved å vise systemet til alle 5 personene i helsebransjen, der vi fikk gode tilbakemeldinger og scores fra de. Gjennomsnittsscoren ble 92, som vi satt opp i tabell 5.

Validering av brukerkrav G
Her testet vi om man får SMS-varslinger selv, som man fikk om gjennomsnittstemperaturen de siste 10 minuttene lå under 2 grader eller stiger over 6 grader. Dermed satt vi scoren til 100, siden systemet oppfyller brukerkrav G fint.

Validering av brukerkrav H
Vi testet om det er enkelt å sjekke temperaturen i kjøleskapet ved å teste brukervennligheten til nettsiden med de 5 personene i helsebransjen. Her fikk vi gode tilbakemeldinger og scores, der gjennomsnittsscoren lå på 97.8, som vi satt i tabell 5.

Validering av brukerkrav I
Så testet vi om sist ukes temperaturdata vises i en tabell, der brukeren kan dokumentere at temperaturen har vært innenfor 2 til 6 grader eller ikke, ved å teste det selv internt. I tabellen får du oppført all data som nevnt over, som er positivt. Det negative er at systemet kan føre feil data siden all temperaturdata fra nøyaktig sist uke fra det tidspunktet du er i vises i tabellen. Dette ble forklart mer nøye tidligere i verifikasjonsdelen av nettsiden. Med tanke på det positive og det negative her, har vi valgt å sette scoren til 78, siden det er forbedringsområder i valideringen av brukerkrav I.

Validering av brukerkrav J
Systemet har kontroll på alle medikamentene i beholderen, og sorterer de basert på utløpsdato. En mulig feilkilde her er om koblingene vi benytter mister koblingene deres sammen, som følge av fysiske årsaker som bevegelse mot ledningene, generell bevegelse mot systemet, osv. Dette kan medføre til at systemet slutter å funke, som er ikke ideelt. En måte vi kan forhindre denne feilkilden på, er ved å benytte egne PCBs, men på grunn av det begrensede budsjettet vi hadde tilgjengelig, valgte vi å ikke gjøre dette. Til slutt valgte vi å sette scoren til 82, siden systemet funker helt fint, men har en feilkilde som kan påføre negative konsekvenser til systemet.

6 Anbefalinger

6.1 Forbedringer av temperatursensorsystem og nettside

Programmet som leser data fra temperatursensoren har ingen feilhåndtering. Hvis kommunikasjonen mellom mikrokontrolleren og sensoren stopper midlertidig, vil programmet fryse og aldri starte på nytt. Dette fører til at temperaturdata på nettsiden vil aldri oppdateres.

Deretter har nettsiden 2 forbedringsområder, som vi ikke fikk korrigert på grunn av tidsmessige årsaker. De er:

1. Ikke-ideell datauthenting for temperaturtabell
2. Varslingssystem som kjører bare når nettsiden er oppe

Som nevnt tidligere i verifikasjonen til nettsiden henter API-en til tabellen i NodeJS de siste 60480 temperaturmålingene. Dette fører til at tabellen består av temperaturmålinger nøyaktig fra 1 uke siden, som medførte til tapt data. Vi fikk ikke korrigert denne ikke-ideelle konfigurasjonen, på grunn av tidsmessige årsaker. En mulig korrigering ville vært å hente data fra databasen basert på dagen data-en oppsto i databasen, i stedet for de siste 60480 temperaturmålingene. På denne måten vil all data fra de siste 7 dagene lastes opp, uavhengig av når på dagen data-en lastes opp, som korrigerer problemet. I tillegg kan det utvikles en funksjonalitet som tillater nedlasting av tabeller fra tidligere uker, men dette fikk vi ikke utviklet på grunn av tidsmessige årsaker.
Deretter er varslingssystemet med SMS-varslinger avhengig av at nettsiden er oppe, ellers kjører ikke varslingssystemet. Vi kunne korrigert dette ved å for eksempel kjøre systemet i en ekstern server, som er alltid oppe. Som den ikke-ideelle datauthentingen for temperaturtabellen, korrigerte vi ikke dette forbedringsområdet på grunn av tidsmessige årsaker.

6.2 Forbedringer av stativ for åpnede medikamenter

Implementasjonen av stativet for åpnede medikamenter har noen svakheter. Det første er at kablene ikke er godt festet i koblingsbrettet og ESP-en. Dette kan løses ved å designe kretsen på en PCB istedet, som også gjør at man slipper alle ledningene under stativet. En annen og betydelig svakhet er at systemet ikke legger til rette for potensielle strømbrudd. Dersom systemet ikke får strømtilførsel vil alle variablene bli slettet og systemet vil begynne i starttilstanden når strømmen starter opp igjen. Dette er en betydelig usikkerhetsfaktor og det er noe som må vurderes før produktet eventuelt blir tatt i bruk. I tillegg tar ikke systemet hensyn til fargeblinde, ettersom funksjonaliteten baserer seg på indikasjoner med lys i forskjellige farger. En mulig løsning til dette er å inkludere skjermer til hver beholder som gir informasjonen som trengs, som åpningsdato, holdbarhet i forhold til de andre beholderene og om den har utløpt eller ikke.

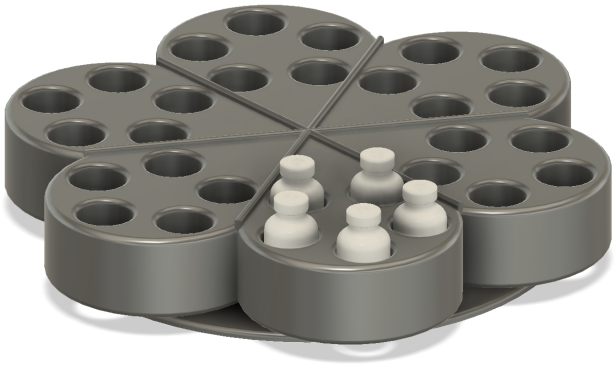
6.3 Forbedring av stativet for uåpnede medikamenter

For å videreføre prosjektet har gruppen funnet noen punkter for et forbedret design ved en eventuell ny prototype for å gjøre designet enda bedre skreddersydd til St. Olavs behov.

Etter tilbakemelding fra poliklinikken, fikk gruppen vite at medikamentene kommer i pakninger med 5 hetteglass, hvor gjeldende 5 medikamenter har samme utløpsdato. Ved et eventuelt nytt design kunne det derfor vært nyttig å samle disse 5 hetteglassene i samme område slik at det blir lettere å sette dem inn, og lettere å holde oversikt for bruker. Et forslag til en slik implementasjon er vist i figur 46.

I ettertid ble det også tenkt på at systemet kan utvides til å kommunisere med nettsiden, slik at sekretærene som har ansvar for medikamentbeholdningen enklere kan få oversikt over utløpsdatoene og antall medikamenter som gjenstår. Dette vil bidra til å gjøre arbeidsoppgavene enda enklere, men har ikke blitt implementert som en følge av tidsbegrensinger.

Til slutt er rotasjonen til stativet basert på tid, som kan medføre en lite avvik under rotasjonen. En mulig korrigering av dette er å utvikle et reguleringssystem, som kan ta hensyn for posisjonsavviket.



Figur 46: Illustrasjon av ide for forbedret design.

7 Konklusjon

Systemet som er utviklet, representerer et betydelig skritt fremover av legemiddeloppbevaringen ved poliklinikken på St. Olavs universitetssykehus. Begge stativene gir en tydelig indikasjon i form av rødt lys for utgåtte medikamenter, samt overvåkning av temperatur gjennom en sensor og tilhørende nettside som formidler dataen og varsler ved avvik. I tillegg viser systemet en mulig måte å effektivisere medikamenthåndteringen på, ved at stativet for uåpnede medikamenter roterer og at tidsstativet har en fargegradient slik at man alltid enkelt får tak i medikamentet med kortest holdbarhet.

Imidlertid har systemet visse mangler som hindrer det i å være markedsklart. Stativet for uåpnede medikamenter har kun plass til 10 medikamenter, som vil si to pakker med lokalbedøvelse. Da 5 og 5 hetteglass har samme utløpsdato vil ikke systemet oppnå sin fulle nytte før det blir utvidet til en større skala som nevnt i 6.3. I tillegg er ikke systemet sikret for potensielle strømbrudd og har ikke tilstrekkelig pålitelig kontakt i den fysiske oppkoblingen slik den er nå. På en klinikk hos et universitetssykehus er det viktig å prioritere sikkerheten til pasientene, og et system som ikke er helt sikkert kan derfor ikke tas i bruk. Dersom temperatursensoren sikres mot strømbrudd, vil den allikevel kunne være tilnærmet markedsklar og en solid forbedring av det nåværende temperaturregistreringen. Selv om stativene som er laget ikke kan tas i bruk direkte, er de allikevel et godt utkast på hvordan man kan redusere svinn, farlige situasjoner og gjøre hverdagen til de ansatte på poliklinikken lettere.

8 Takk

Takk til sykepleierstudentene Marthe Høgbråt og Haldis Kathrine Skogslund, tidligere ambulansefag student Mariell Harby, medisinstudenten Khalil Zogby og tidligere lege Abdullah Zogby, for gode tilbakemeldinger på prosjektet vårt.

Takk til Andrew Perkis, som ga oss god veiledning under prosjektarbeidet.

Takk til studentassistentene for god hjelp under prosjektet.

Takk til gruppe 1, 3 og 4 for å komme helt hit i gjennomlesningen av rapporten og forhåpentligvis gode tilbakemeldinger til vårt rapportutkast.

Referanser

[1] St. Olavs hospital, *Fysikalsk medisinsk poliklinikk*, https://www.stolav.no/avdelinger/rehabiliteringsklinikken/Fysikalsk-medisinsk-poliklinikk/, 1, 2024.

[2] Apotek 1, *Holdbarhet på legemidler*, https://www.apotek1.no/bruk-av-legemidler/holdbarhet-paa-legemidler, 1, 2024.

[3] OpenAI, *Dalle-e Image Generator*, https://chat.openai.com/g/g-2fkFE8rbu-dall-e, 1, 2024.

[4] Store norske leksikon, *API*, https://snl.no/API, 22, 2024.

[5] verywellmind, *What Does the Color Green Mean?*, https://www.verywellmind.com/color-psychology-green-2795817, 1, 2023.

[6] Centers for Disease Control and Prevention, *Healthcare Workers and Work Stress*, https://www.cdc.gov/niosh/topics/healthcare/workstress.html, 1, 2023.

[7] Bosch, *BME680*, https://cdn-shop.adafruit.com/product-files/3660/BME680.pdf, 4, 2024.

[8] Bosch Sensortec, *BME680_SensorAPI*, https://github.com/boschsensortec/BME680_SensorAPI, 4, 2024.

[9] Computerscience.org, *Front-End vs. Back-End: What's the Difference?*, https://www.computerscience.org/bootcamps/resources/frontend-vs-backend/, 1, 2023.

[10] Saranya S Kumar, *The Node.js and React.js Revolution: A Deep Dive into the Job Market Trends*, https://medium.com/@saranyasasikumar06/the-node-js-and-react-js-revolution-a-deep-dive-into-the-job-market-trends-71ca285c07 1, 2023.

[11] Zippy, *MICRO SWITCHES*, https://www.zippy.com/files/20231211/DF.pdf, 4, 2024.

[12] J. R. Kristiansen «Kraftmoment» SNL.no. Hentet fra: https://snl.no/kraftmoment (Lastet ned: 26.04.2024).

[13] Omron, *Tactile Switch B3F*, https://eu.mouser.com/datasheet/2/307/en-b3f-13826.pdf, 4, 2024.

[14] Texas Instruments, *[FAQ] How do I size pull-up or pull-down resistors?*, https://e2e.ti.com/support/logic-group/logic/f/logic-forum/751472/faq-how-do-i-size-pull-up-or-pull-down-resistors#:~:text=Pull-up%20and%20pull-down%20resistors%20are%20required%20in%20many,10k%CE%A9%20resistor%20and%20adjust%20if%20you%20have%20problems., 1, 2024.

[15] Worldsemi, *WS2812B Intelligent control LED integrated light source*, https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf, 4, 2024.

[16] Last Minute Engineers, *Control WS2812B Addressable LEDs with ESP32 and WLED*, https://lastminuteengineers.com/esp32-wled-tutorial/, 1, 2024.

[17] Arduino, *millis()*, https://www.arduino.cc/reference/en/language/functions/time/millis/, 4, 2024.

[18] Arduino, *unsigned long*, https://www.arduino.cc/reference/en/language/variables/data-types/unsignedlong/, 4, 2024.

[19] Raspberry Pi, *Raspberry Pi 4 Model B* https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf, 3, 2024.

[20] MPJA, SERVO MOTOR SG90 DATA SHEET", datablad, nov. 2021. https://lastminuteengineers.com/esp32-wled-tutorial/, 2024.

[21] Slipring, *What is an electrical slip ring?*, https://slip-ring.com/what-is-an-electrical-slip-ring/, 1, 2024.

A Vedlegg kode

All kode brukt i prosjektet ligger i URL-en under.
https://gitlab.stud.idi.ntnu.no/asilz/elsys-gruppe-2