# Oppgave 1

## a)

> V modellen har en direkte sammenheng mellom spesifikasjonskrav og implementasjon. Man kan lett se direkte sammenheng mellom implementasjon og spesifikasjonskrav. Vanskelig å benytte under store og objektorienterte prosjekter.

## b)

> FAT er Factory Acceptance Test og er en test på om produktet når kravspesifikasjonene. SAT er Site Acceptancy Test er en hard test som "stresstester" produktet. I FAT testes produktet for et krav av gangen, mens i SAT kan en kombinasjon av krav bli stillt samtidig til produktet og føre til at det ikke fungerer slik det skal. I SAT kan for eksempel moduler faile på radiokommunikasjon hvis det er mye støy i radio rommet.

## c)

1) En 70dB tone på 20kHz spilles av og skal oppdages av babycalleren når den er innen rekkevidden. Babycalleren skal være innstillt på å kunne måle 70dB.

2) Produktet skal kunne måle lyd via hardware for lydopptak og kunne sende det til ADC for å gi en digital representasjon av lydbildet.

3) Produktet skal plasseres i et 9m^2 rom med en skrikende baby og skal kunne sende representasjon av lyden til babycaller innen rekkevidden sin.

# Oppgave 2

## a)

> Vi skal håndtere lyd med høyeste frekvens på 20kHz. Vi bør derfor sample med minst 40kHz sampling rate for å unngå nedfolding. Jeg tror det er gangske mye for et system og derfor må samplingen skje ganske fort. Samplingen trenger ikke å være veldig presist da det ikke er musikk vi skal høre på. Derfor ville jeg ha valgt en flash ADC. Det jeg absolutt ikke ville ha valgt hadde vært en servoomsetter. Da hadde ADC aldri funnet riktig verdi å sende videre til radioenheten fordi fluksjoasjonene hadde vært for raske.

## b)

> Da kan man bruke multiplexer til å "interupte" lydsamplingen.

## c)

> For å kunne representere 60 forskjellige grader med oppløsning på 0.5 grad per verdi trengs det 120 forskjellige verdier. Derfor er det nødvendig med en oppløsning på minst 7 bit gitt at referansespenningen på ADC skal holdes konstant (antar her 10V). Siden en mikrofon skal også

> benytte denne ADC, så må vi liniært skalere antall verdier vi trenger helt til vi klarer å representere referansespenningen på 10V. Dermed blir endelig krav en ADC med minimum 9 bits oppløsning. Men det er vanligvis mye lettere å finne 10 bits ADCs på markedet enn 9 bits. Legger også merke til at mikrofonen gir ut en spenning mellom -5v og 5v. Denne skalerer vi bare opp til å passe inn i ADC.

### d)

> Antar at vi sampler 2 ganger høyeste frekvens som et "worst case scenario". Da har vi en samplingsrate på 40kHz og vi skal ha 9 bits per samle. Dermed har vi 360 000 bits/s.

### e)

> HW tiltak:
>
> 1. Legge inn aliasfilter slik at kun bølger i det ønskede område kommer frem.
>
> Tiltak til familien:
>
> 1. Kjøpe ny mikrobølgeovn
> 2. Passe på at mikrobølgeovnen er koblet til en stikkontakt med jording. Hvis man har ei hylle av metall så kan man plassere mikrobølgeovnen der.
> 3. Unngå å holde babycalleren og mikrobølgeovnen i nærheten av hverandre. Når begge er i drift.

## Oppgave 3

> spi.h

```
#ifndef SPI_H
#define SPI_H
void spi_init();
void spi_send_byte(char letter);
void spi_read_byte();
#endif
```

man tar differansen mellom to adresser og deler på `sizeof(uint32_t)` og ganger med 8 som tilsvarer størrelse på en byte. Etter på trekker man fra 1 for å ikke overskrive det nye elementet.

> spi.c

```
#include "spi.h"

#include <stdint.h>
#define SPI0 ((NRF_SPI_REG*)0x40003000)
#define SPI1 ((NRF_SPI_REG*)0x40004000)
#define PIN_DISABLED 0xFFFFFFFF
```

```
#define PIN_MOSI 21
#define PIN_SIMO 22
#define PIN_CLOCK 23

typedef struct {
    volatile uint32_t RESERVED1[66];
    // events
    volatile uint32_t READY;
    volatile uint32_t RESERVED2[125];
    // registers
    volatile uint32_t INTEN;
    volatile uint32_t INTENSET;
    volatile uint32_t INTENCLR;
    volatile uint32_t RESERVED3[125];
    volatile uint32_t ENABLE;
    volatile uint32_t RESERVED4[1];
    volatile uint32_t PSELSCK;
    volatile uint32_t PSELMOSI;
    volatile uint32_t PSELMISO;
    volatile uint32_t RESERVED5[1];
    volatile uint32_t RXD;
    volatile uint32_t TXD;
    volatile uint32_t RESERVED6[1];
    volatile uint32_t FREQUENCY;
    volatile uint32_t RESERVED7[11];
    volatile uint32_t CONFIG;
} NRF_SPI_REG;

void spi_init(){
    SPI->ENABLE = 0;
    SPI->PSELSCK = PIN_CLOCK;
    SPI->PSELMOSI = PIN_MOSI;
    SPI->PSELSIMO = PIN_SIMO;
    SPI->FREQUENCY = 0x02000000; // 125 kbs
    SPI->ENABLE = 1;
}

void spi_send_byte(byte letter){
    SPI->TXD = letter;
}


void spi_read_byte(){
    while(!SPI->READY);
    return SPI->RXD;
}
```

main.c

```
#include "spi.h"
int main(){
```

```
    spi_init();

    ...

    return 0;
}
```

## b)

siterer databladet:

> Only one peripheral can be assigned to drive a particular GPIO pin **at a time**, failing to do so may result in **unpredictable behavior**.

Altså med andre ord så kan vi ikke kommunisere med flere enheter samtidig. Dette kan skape litt trøbbel, men man kan bare avslutte kommunikasjonen med en enhet og starte med en annen. Alle SPI enheter kan være koblet på samme linje, men flere kan ikke være selected av "slave select". Dermed bruker man kun en GPIO pin per ny enhet man ønsker å koble til CPUen.

## c)

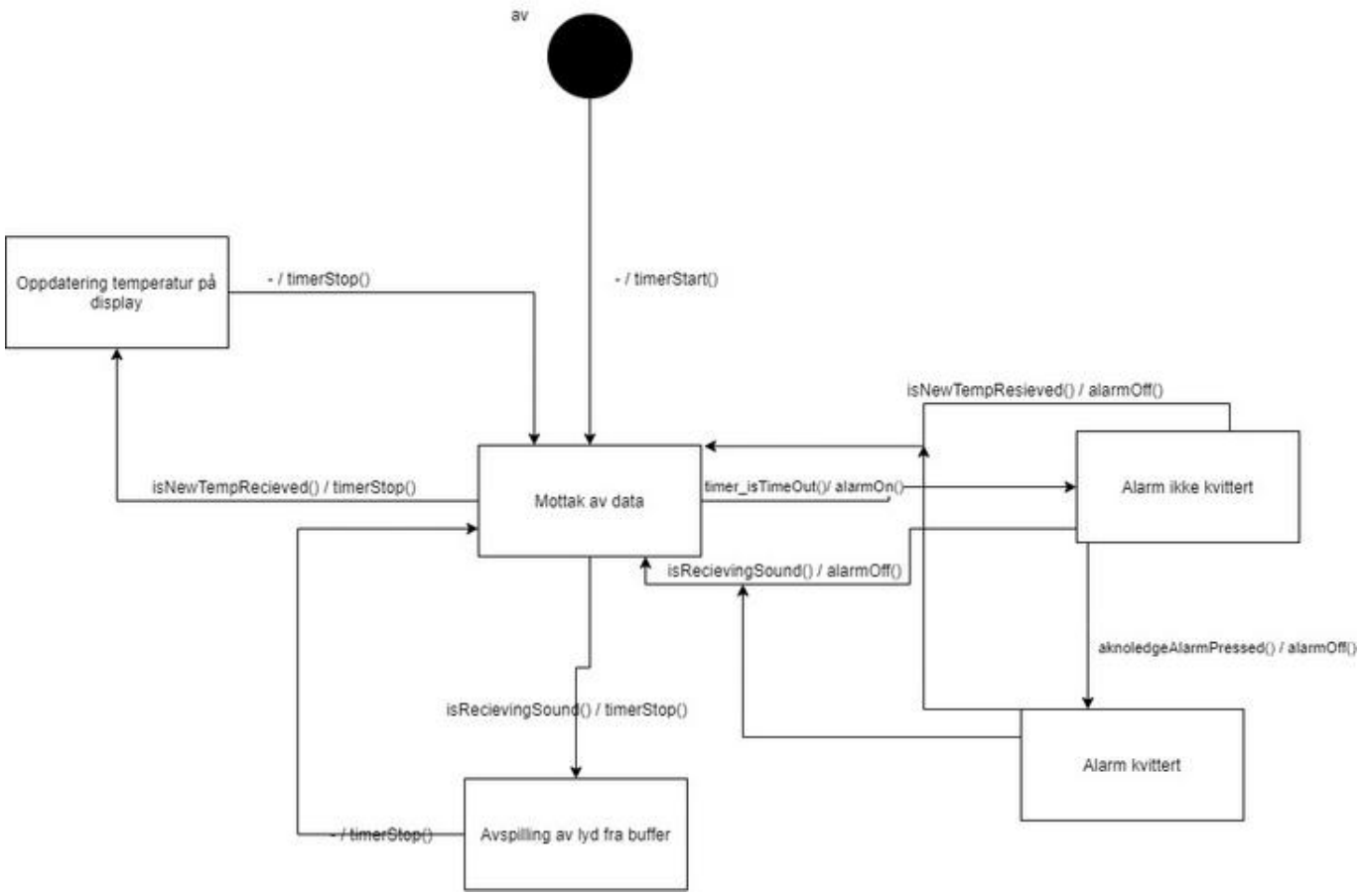
Har besvart oppgaven i a). under init funkjsonen

## d)

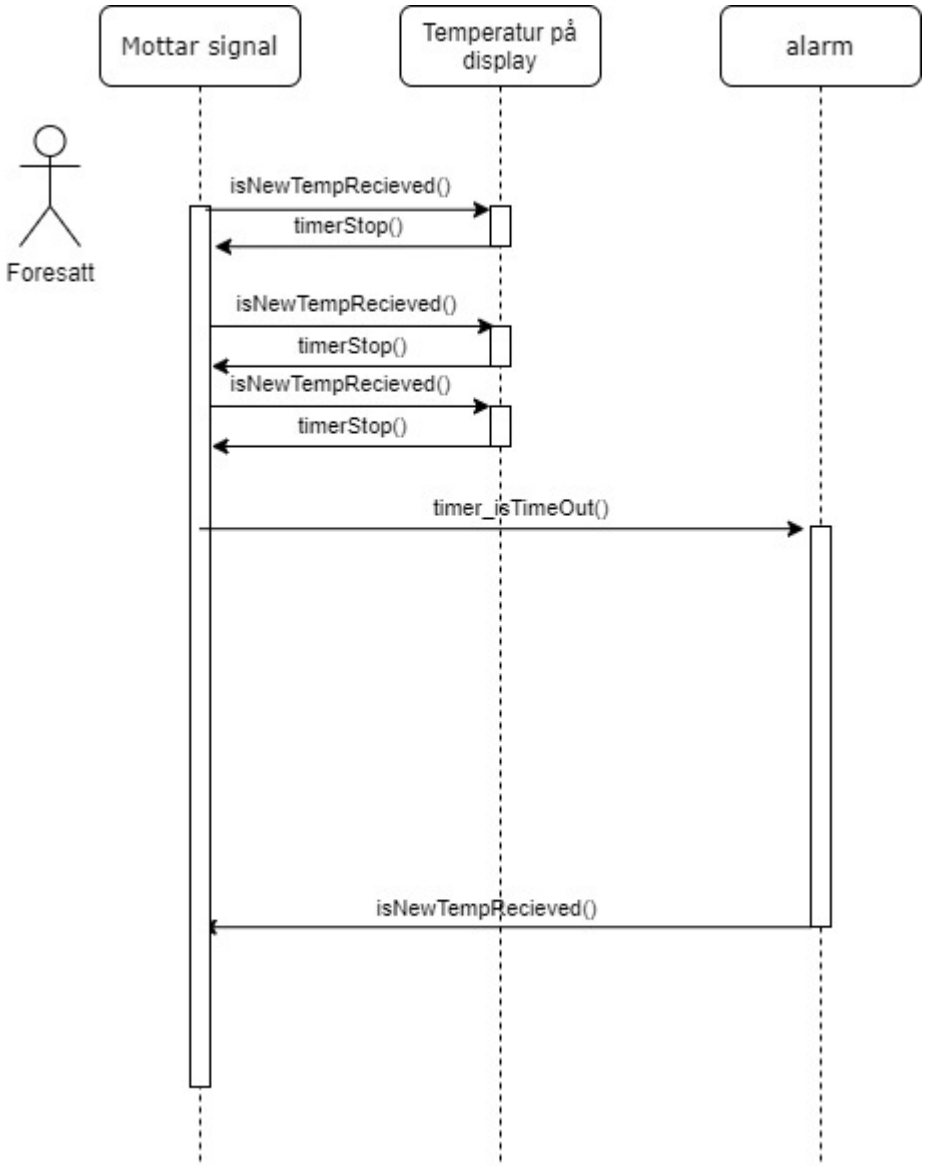
Setter den så høy som mulig så lenge det ikke fører til andre problemer. Men alt over 45kb/s (360 000 bits/s) fungerer. Det vil si vi kan ta den tregeste hastigheten også

# Oppgave 4

## a) b)


av
Oppdatering temperatur på display
- / timerStop()
- / timerStart()
isNewTempResieved() / alarmOff()
isNewTempRecieved() / timerStop()
Mottak av data
timer_isTimeOut()/ alarmOn()
Alarm ikke kvittert
isRecievingSound() / alarmOff()
aknoledgeAlarmPressed() / alarmOff()
isRecievingSound() / timerStop()
Alarm kvittert
- / timerStop()
Avspilling av lyd fra buffer

c)


Mottar signal
Temperatur på display
alarm
Foresatt
isNewTempRecieved()
timerStop()
isNewTempRecieved()
timerStop()
isNewTempRecieved()
timerStop()
timer_isTimeOut()
isNewTempRecieved()