

**NTNU**

Institutt for elektroniske systemer

---

TTT4270 - ELEKTRONISK SYSTEMDESIGN, PROSJEKT

---

## Floating and Identifying Signal Capsule (FISC)

---

Elinda F. Engvik, Kim Hamberg, Tuva Ludvigsen, Nhan-Daniel V. Nguyen,  
Kritagya Panthi



Gruppe 10

4. mai 2025

# Sammendrag

Havbruksnæringen medfører betydelig risiko, og fall over bord kan få fatale konsekvenser [1]. Denne rapporten presenterer design, implementasjon og testing av prototypen FISC (Floating and Identifying Signal Capsule); et sikkerhetssystem utviklet for å redusere responstiden ved slike hendelser. Målet er å varsle mannskap om bord og nødetater raskt, samt gi periodevis posisjonsoppdatering av personen i vannet.

Designet av FISC består av en senderenhett, basert på Nordic Semiconductor nRF-teknologi med mobilnettbasert kommunikasjon, plassert i en vanntett, flytende kapsel festet til brukerens redningsvest. FISC er også designet for å utløses automatisk ved kontakt med vann eller manuelt av brukeren. Ved utløsning aktiveres en lokal alarm, et varsel sendes til et alarmsystem på fartøyet, og posisjonsdata publiseres periodevis til en webapplikasjon visualisert på et kart.

En prototype ble implementert med nRF9160 DK-kort for både sender og mottaker. Testing verifiserte sentral funksjonalitet: Senderen utløste alarm på mottakeren innenfor kravet på 5 sekunder, og posisjonsdata ble korrekt vist og oppdatert på webkartet kun etter aktivering.

Prosjektet demonstrerte at FISC-konseptet er teknisk gjennomførbart og møter sentrale funksjonelle krav og gir et solid grunnlag for videreutvikling til et robust sikkerhetsprodukt for maritime nærlinger.

# **Anerkjennelser**

Takk til Carl Richard Steen Fosse for god veiledning gjennom hele prosjektet og utdeling av nRF9160 DK og Nordic Thingy:91.

Takk til Trine Thorvaldsen og Siri Mariane Holen fra SINTEF Ocean for god innsikt i HMS og sikkerhet ved havbruksnæringen.

Takk til Sverre Flatebø fra Sjøfartsdirektoratet.

Takk til Simen Dymbe og Sima Moradi for hjelpen med bestilling av redningsvest.

Takk til Redningsselskapet for redningsvest.

Takk til Omega Verksted for hjelp med 3D-printing.

# Innhold

<b>1 Innledning</b>	<b>1</b>
1.1 Bakgrunn . . . . .	1
1.2 Motivasjon . . . . .	1
1.3 Problemstilling . . . . .	2
1.4 Idé . . . . .	2
1.5 Brukerkrav . . . . .	2
<b>2 Konsept</b>	<b>4</b>
2.1 Beskrivelse . . . . .	4
2.2 Andre bruksområder og egenskaper . . . . .	5
2.3 Systemkrav . . . . .	5
<b>3 Design</b>	<b>9</b>
3.1 Kapsel . . . . .	11
3.2 Utløsermekanisme . . . . .	13
3.3 Kommunikasjon: LTE-M og MQTT . . . . .	14
3.4 Sender . . . . .	16
3.5 Alarmsystem . . . . .	17
3.6 Webapplikasjon . . . . .	17
<b>4 Implementering</b>	<b>19</b>
4.1 Fysisk utforming . . . . .	19
4.2 Kapsel . . . . .	20
4.3 Utløsermekanisme . . . . .	24
4.4 Sender . . . . .	25
4.5 Alarmsystem . . . . .	29
4.6 Webapplikasjon . . . . .	32
<b>5 Validering, verifikasjon og test</b>	<b>35</b>
5.1 Verifikasjonsplan . . . . .	35
5.2 Testing av verifikasjonsplan . . . . .	37
5.2.1 Systemkrav 1 . . . . .	37
5.2.2 Systemkrav 2 . . . . .	37
5.2.3 Systemkrav 3 . . . . .	38
5.2.4 Systemkrav 4 . . . . .	39
5.2.5 Systemkrav 5 . . . . .	40
5.2.6 Systemkrav 6 . . . . .	40

5.2.7	Systemkrav 7 . . . . .	41
5.2.8	Systemkrav 8 . . . . .	41
5.2.9	Systemkrav 9 og 10 . . . . .	41
5.2.10	Systemkrav 11 . . . . .	42
5.2.11	Systemkrav 12 . . . . .	43
5.2.12	Systemkrav 13 og 14 . . . . .	43
5.2.13	Systemkrav 15 . . . . .	44
5.3	Testresultater . . . . .	44
5.4	Validering . . . . .	46
<b>6</b>	<b>Forbedringer</b>	<b>47</b>
6.1	Fysisk utforming . . . . .	47
6.2	Brukervennlighet . . . . .	47
6.3	Pålitelighet . . . . .	47
6.4	Programvare og tilgjengelighet . . . . .	48
<b>7</b>	<b>Konklusjon</b>	<b>49</b>
<b>Bibliografi</b>		<b>50</b>
<b>A GitHub for fullstendig prototype</b>		<b>53</b>
<b>B Kode for nettside</b>		<b>54</b>
B.1	server.py . . . . .	54
B.2	index.html . . . . .	56
<b>C Kode for alarm</b>		<b>61</b>
C.1	main.c . . . . .	61
C.2	prj.conf . . . . .	70
<b>D Kode for sender</b>		<b>72</b>
D.1	main.c . . . . .	72
D.2	prj.conf . . . . .	80
<b>E KI-deklarasjon</b>		<b>82</b>

# Forkortelser

**FISC** Floating and Identifying Signal Capsule. i, 1, 2, 5, 10, 19, 20, 47, 49

**GNSS** Global Navigation Satellite System. 14, 15, 20, 26

**GPIO** General-purpose input/output. 20, 26, 29

**IoT** Internet of Things. 14

**JSON** JavaScript Object Notation. 32

**LTE-M** Long-Term Evolution Machine Type Communication. 14–16

**MCU** Microcontroller unit. 16

**MQTT** Message Queue Telemetry Transport. 15–18, 26, 32

**NB-IoT** Narrowband Internet of Things. 14

**SINTEF** Stiftelsen for industriell og teknisk forskning ved Norges tekniske høgskole. ii, 1, 2

**SNR** Signal-to-noise ratio. 41

# 1 Innledning

## 1.1 Bakgrunn

Havbruksnæringen er en av de største eksportindustriene i Norge [2], og er predikert til å øke enda mer i fremtiden [3]. Verdensbefolkningen øker [4], og som følge av dette må matproduksjonen følge med. Havbruksnæringen blir sett på som en god løsning av flere forskere, og investeringen og utviklingen av norsk havbruk er derfor stor. Det er derfor et betydelig behov for teknologisk innovasjon for å sikre effektiv, lønnsom, miljøvennlig og trygg produksjon i disse kystnære operasjonsmiljøene [5].

## 1.2 Motivasjon

Havbruksnæringen er også en farlig industri vi har i Norge. Tungt arbeid, dårlig vær og store maskiner leder til slitasjeskader, arbeidsulykker og i verste fall dødsfall. Med en voksende industri øker også antallet ansatte, hvilket utsetter flere mennesker for skadelige situasjoner [1, s. 4, 12].

Mange arbeidere jobber i tillegg alene eller langt ute på havet, for eksempel ved fiskeoppdrettsanlegg som ofte er plassert i værharde kystområder et stykke fra land [6]. Dersom en arbeider skulle falle over bord under slike forhold, er det derfor ikke alltid enkelt å få tak i hjelp, og enklere blir det ikke på høsten og vinteren [1, s. 26]. I 2023 etablerte regjeringen en nullvisjon for omkomne og hardt skadde på sjøen [7].

Sjøfartsdirektoratet, Kystverket, Fiskeridirektoratet, politiet og Direktoratet for samfunnssikkerhet og beredskap (DSB) har også vedtatt en handlingsplan fra starten av 2025 for å jobbe mot nullvisjonen [8]. Mowi ASA, Salmar ASA og Lerøy Seafood Group er blant de store aktørene i havbruk som har delt sine erfaringer og gitt innspill til utviklingen av handlingsplanen [8, s. 80]. Utvikling av ny teknologi innenfor HMS og sikkerhet er derfor svært relevant og nyttig for industriens framtid.

Ifølge en rapport skrevet av SINTEF fra 2021 utgjør ”fall til sjøs” drøyt 4 % ( $n = 273$ ) av alle ulykkeshendelsene [1, s. 11] i årene 2012-2022. Likevel er det denne typen ulykke som er av de mest kritiske [1, s. 24], og som kan få fatale konsekvenser dersom det ikke handles raskt nok [9, s. 84].

## 1.3 Problemstilling

Basert på næringens betydning, de iboende farene, og den nasjonale satsingen på sikkerhet, defineres prosjektets problemstilling som følger: *Hvordan redusere omfanget og forhindre dødsfall ved fall over bord?*

## 1.4 Idé

For å løse problemstillingen om fall over bord introduseres konseptet FISC (Floating and Identifying Signal Capsule). Ideen er å utvikle et system som festes som et tillegg til arbeidernes redningsvester. Dette systemet skal automatisk eller manuelt kunne detektere en 'fall over bord'-situasjon, og deretter fungere som et varslings- og posisjoneringssystem for å tilkalle hjelpe raskt.

Selv om det finnes eksisterende teknologier som satellittbaserte nødpeilesendere (EPIRB [10]) og systemer for sporing om bord på fartøy [11], adresserer FISC-konseptet behovet for en personlig, raskt aktiverende enhet som er uavhengig av kontinuerlig kontakt med et moderfartøy for den initiale varslingen og lokaliseringen.

## 1.5 Brukerkrav

Gjennom dialog med vår veileder Carl Richard Steen Fosse, og dialog med Trine Thorvaldsen og Siri Mariane Holen fra SINTEF Ocean om realisering av idéen, utarbeidet vi brukerkravene presentert i Tabell 1.1.

Tabell 1.1: Brukerkrav

Nr.	Brukerkrav
1	Systemet skal varsle og sende posisjon ved fall over bord.
2	Systemet skal kunne utløses og sende signal selv om personen er i vann.
3	Systemet skal ikke sende signal før personen har falt over bord eller før manuell utløsning.
4	Systemet skal ikke komme i veien for arbeidernes daglige arbeidsoppgaver.
5	Systemet skal være enkelt å komme til for vedlikehold.
6	Systemet skal ikke dempe effekten av redningsvesten.

Hvert brukerkrav er begrunnet med et punkt i den nummererte listen:

1. Dette kravet er essensielt da det er hovedoppgaven til systemet.

2. Det er viktig at systemet er funksjonelt til tross for at det er i vann, da det skal brukes på sjøen.
3. For å unngå falske alarmer bør ikke systemet sende varsel før en faktisk hendelse har hendt.
4. Systemet burde ikke gjøre arbeidet til brukerne vanskeligere, da dette kan føre til at færre vil bruke produktet.
5. Dette kravet er satt med tanke om at det skal være lett tilgjengelig å reparere eventuelle feil eller mangler ved systemet.
6. Det er sentralt at redningsvesten gjør sin oppgave etter tilleggsystemet er implementert, da den spiller en avgjørende rolle for sikkerheten ved fall over bord.

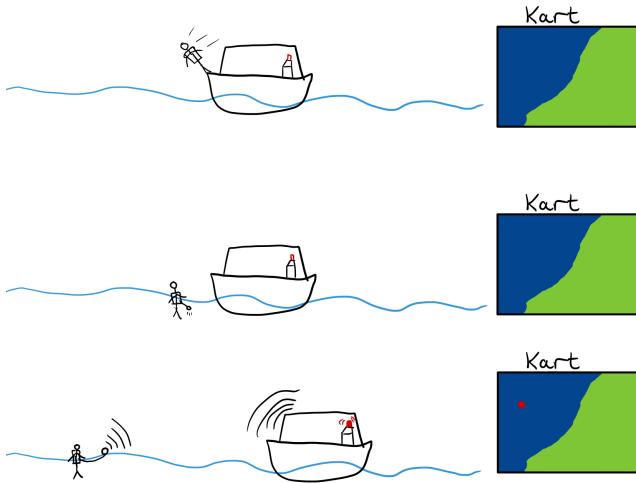
## 2 Konsept

### 2.1 Beskrivelse

Som nevnt i problembeskrivelsen i seksjon 1.3 er formålet å redusere omfanget og forhindre dødsfall ved fall over bord. Med bakgrunn i dette har det blitt utviklet et konsept som er et personlig varslingssystem som sporer posisjonen og videresender denne via mobilnett.

Systemet består av en kapsel med en sender i, et alarmsystem og en webapplikasjon for nødetatene. Når personen faller over bord, vil senderen i kapselen sende personens lokasjon til webapplikasjonen og meddele alarmsystemet. Et alarmsystem kan for eksempel være montert på et fartøy, slik at mannskapet til fartøyet vil høre en tydelig alarm fra alarmsystemet når personen faller over. Når webapplikasjonen har mottatt posisjonen vil kapselen begynne å lyse, slik at personen som har falt over bord vet at varslingen er sendt, samt at det kan være lettere å finne personen ved en eventuell redningsaksjon.

Systemet skal fungere som et tillegg til redningsvester som brukes ved arbeid på dekk, og vil utløses enten manuelt eller ved kontakt med vann. Når systemet utløses, vil kapselen løsne fra redningsvesten og flyte opp til vannoverflaten. Dette er for å sørge for god kommunikasjon mellom senderen og mottakeren, da vann vil absorbere mobilnettbaserte signaler [12]. For å sikre at kapselen ikke forsvinner fra personen, er den festet i redningsvesten med en snor, som er langt nok til at kapselen kan flyte uten risiko for å bli dratt under vann. En enkel skisse av systemets virkemåte er vist i Figur 2.1.



Figur 2.1: Illustrasjon av systemets virkemåte.

## 2.2 Andre bruksområder og egenskaper

Det oppstår en rekke ulike farlige situasjoner på både oppdrettsanlegg og fartøy, og det å kunne tilkalle hjelp er viktig [1, s. 10]. Med FISC vil varslingmekanismen være festet til personen og være lett tilgjengelig, da signalet også skal kunne sendes om redningsvesten utløses manuelt. Dette kan være nyttig i flere situasjoner, for eksempel dersom en arbeider er alene på en båt eller merd, og av forskjellige grunner har problemer med å komme seg til en annen alarm eller telefon. Dette kan for eksempel være en kranulykke der arbeideren er i klem, og ikke klarer å bevege seg [1, s. 11].

Det er også andre farlige yrker i Norge med lignende ulykker. Siden 2000 har 155 arbeidere i fiskerinæringen omkommet i arbeidsulykker, og mange av disse ved fall over bord [13]. FISC vil derfor kunne være med på å redde liv også i denne industrien. Andre bruksområder kan også være til privat bruk i fritidsbåter eller seilbåter, og i oljebransjen dersom mobildekningen blir utvidet.

## 2.3 Systemkrav

For å oppnå brukerkravene definert i Tabell 1.1 er det utarbeidet systemkrav. Disse er satt for å kunne implementere ønsket funksjonalitet i prototypen, og er presentert i Tabell 2.1.

Tabell 2.1: Systemkrav inndelt i delsystemer

Nr.	Systemkrav	Oppfyller: Brukerkrav
↓	<b>Kapsel</b>	
1	Senderen skal beskyttes av en kapsel.	2, 5
2	Kapselen som senderen beskyttes av burde ikke slippe inn vann.	2
3	Kapselen skal flyte stabilt, med senderen over vann.	2
4	Kapselen med senderen skal ha en massetethet mindre enn vann.	2, 6
5	Kapselen med senderen burde sitte komfortabelt på arbeidernes kropp, og ikke begrense daglig bevegelse.	4
6	Kapselen med senderen burde være monterbar og avtagbar innen en tidsramme på 5 minutter, utført med hendene.	5, 6
7	Etter utløsning skal kapselen være festet til redningsvesten via en snor på minst 2 meter	2
↓	<b>Sender</b>	
8	Senderens evne til å sende signal burde ikke svekkes i vannet sammenlignet med på land.	1, 2
9	Senderen skal ikke sende signal før det utløses manuelt.	3
10	Senderen skal ikke sende signal før det kommer i kontakt med vann.	3
11	Senderen skal ikke dele posisjon i kartet før systemet er blitt utløst.	1, 3
12	Senderen skal begynne å lyse når webapplikasjonen har mottatt posisjonen.	1
↓	<b>Alarmsystem</b>	
13	Alarmsystemet skal utløse en hørbar alarm innen 5 sekunder etter at senderen er blitt utløst i vann.	1, 2, 3
14	Alarmsystemet skal utløse en hørbar alarm innen 5 sekunder etter manuell utløsning.	1, 2, 3
↓	<b>Webapplikasjon</b>	
15	Posisjonen til senderen skal oppdateres regelmessig innen hvert 5. sekund.	1

Hvert systemkrav er begrunnet med et punkt i den nummererte listen:

- Motiveres av brukerkrav 2 og 5. Kravet er satt med bakgrunn i at senderen skal kunne beskyttes mot ytre påvirkninger som vær og fysisk arbeid, eller fall om bord,

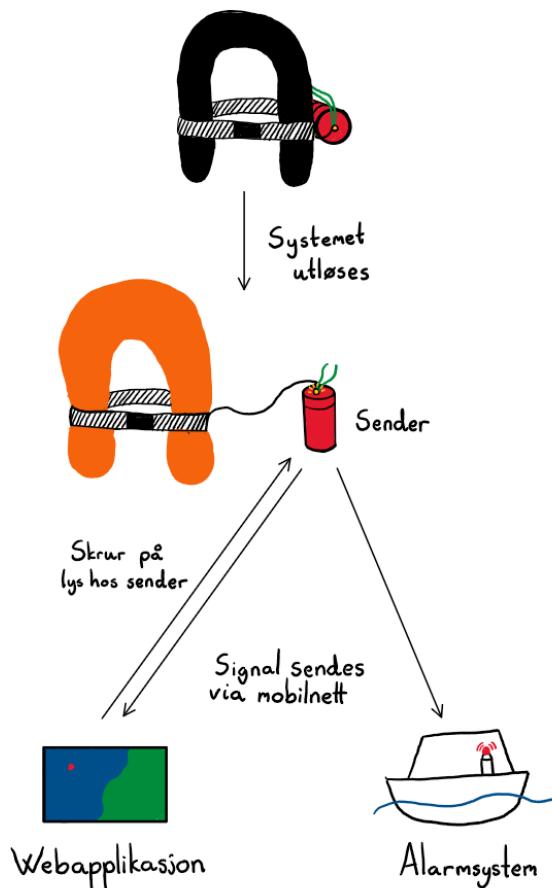
samt at det skal være mulig å åpne kapselen for vedlikehold.

2. Motiveres av brukerkarav 2. Det er sentralt at systemet skal kunne tåle vann, da omgivelsene det skal brukes i er sjøen. Likevel blir ikke dette sett på som nødvendig for prototypen.
3. Motiveres av brukerkarav 2. Dette kravet er satt med bakgrunn i at senderen gir best kommunikasjon når den ligger over vannoverflaten. Derfor skal kapselen flyte stabilt, med senderen i toppen.
4. Motiveres av brukerkarav 2 og 6. For å sørge for at kapselen med senderen flyter må den ha en massetethet mindre enn vann. Dersom kapselen med senderen ikke flyter vil dette videre dempe effekten av senderen.
5. Motiveres av brukerkarav 4. Dette kravet er satt med tanke på å sikre brukervennlighet og funksjonalitet. Systemet skal være behagelig for brukeren, og brukeren skal ikke påføre systemet skade ved daglige arbeidsoppgaver. Dette kravet blir ikke sett på som nødvendig for prototypen.
6. Motiveres av brukerkarav 5 og 6. For å sørge for enkelt vedlikehold av systemet for vedlikeholdsansvarlige burde kapselen være lett å ta av og på redningsvesten. Dette kravet blir ikke sett på som nødvendig for prototypen.
7. Motiveres av brukerkarav 2. For å sørge for at kapselen ikke blir dratt under vannoverflaten, for eksempel dersom personen skulle bli dratt under vann, er det hensiktsmessig at kapselen er festet til vesten med en snor. I tillegg vil posisjonen til senderen kunne følge bevegelsen til personen.
8. Motiveres av brukerkarav 1 og 2. Dette kravet er satt for å sørge for at webapplikasjonen får posisjonen til senderen og at alarmsystemet går av når personen faller i vannet.
9. Motiveres av brukerkarav 3. Dette kravet er satt med en tanke om å unngå falske alarmer, noe som ville gitt usikkerhet ved bruk av produktet.
10. Motiveres av brukerkarav 3. Her gjelder det samme som for systemkrav 9, men ved kontakt med vann.
11. Motiveres av brukerkarav 1 og 3. Dette kravet er satt for å forhindre unødvendige distraksjoner på kartet, slik at det kun er mennesker i nød som vises for nødetatene.
12. Motiveres av brukerkarav 1. Dette kravet er satt med tanke om å gi den rammende en følelse av trygghet, da et lys på senderen informerer at posisjonen er sendt til nødetatene. I tillegg kan lyset bidra til en lettere og mer effektiv leteaksjon.
13. Motiveres av brukerkarav 1, 2 og 3. Det er viktig for funksjonaliteten at det ikke er for stor forsinkelse av den hørbare alarmen om bord, da det er kritisk å handle raskt ved farlige situasjoner som fall over bord.

14. Motiveres av brukerkarav 1, 2 og 3. Her gjelder det samme som for systemkarav 13, men ved manuell utløsning istedenfor ved kontakt med vann.
15. Motiveres av brukerkarav 1. På grunn av bevegelse og strømmer i sjøen er det nødvendig å oppdatere posisjonen til senderen regelmessig, slik at systemet har tilstrekkelig nøyaktighet.

### 3 Design

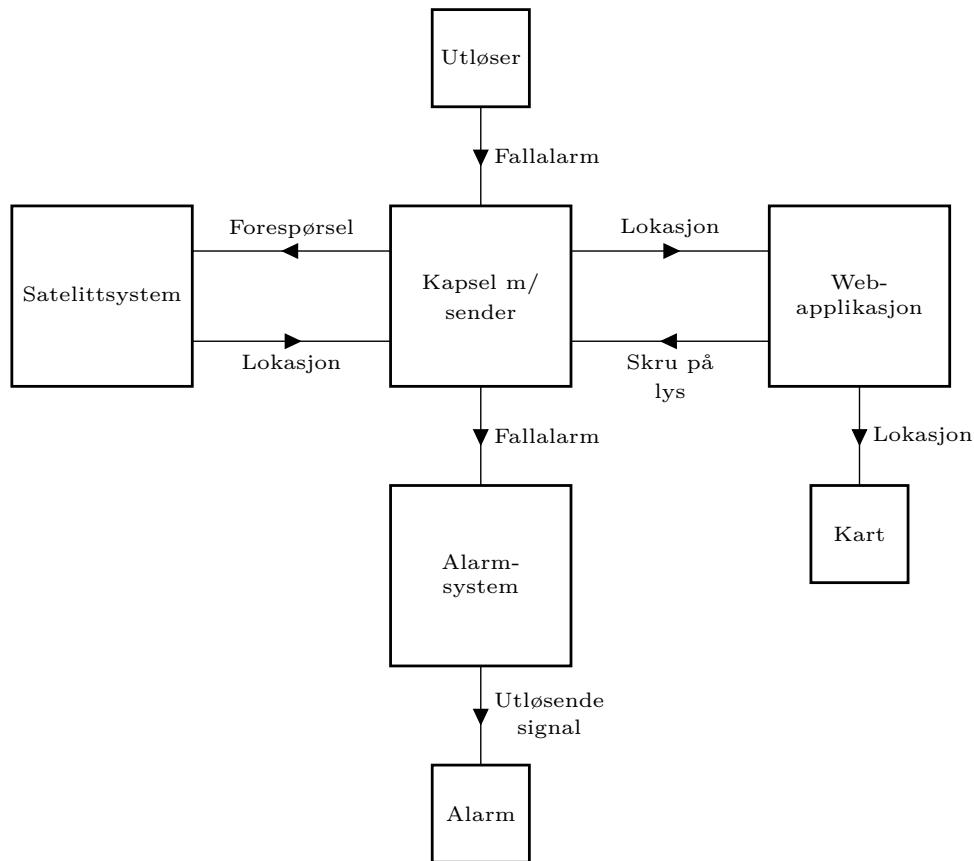
FISC-systemet består av en senderenhet plassert i en kapsel på brukerens redningsvest, en utløsermekanisme koblet til vesten, et alarmsystem, og en webapplikasjon for nødtilkall. Det totale systemet er vist i Figur 3.1. Kommunikasjonen mellom enhetene og posisjonsinnhenting baseres på mobilnettverksteknologi og lokasjonsteknologi.



Figur 3.1: Designet for den fysiske utformingen av systemet.

Systemflyten er illustrert i Figur 3.2. Systemet fungerer på følgende måte:

- **Utløser:** En hendelse (vannkontakt eller manuell aktivering via vesten) endrer tilstanden til utløseren.
- **Kapsel med sender:** Senderen detekterer tilstandsendringen fra utløseren og aktiveres. Den henter posisjonsdata via lokasjonsteknologi.
- **Kommunikasjon:** Senderen publiserer en fallalarm og lokasjonsdata via mobilnettverksteknologi.
- **Alarmsystem:** Alarmsystemet mottar fallalarmen, som utløser en lokal alarm ved lyd og lys.
- **Webapplikasjon:** Webapplikasjonen mottar lokasjonen, og viser det på et kart i sanntid for nødetatene. Etter å ha mottatt lokasjonen sender webapplikasjonen et ”skru på lys”-signal tilbake til senderen.



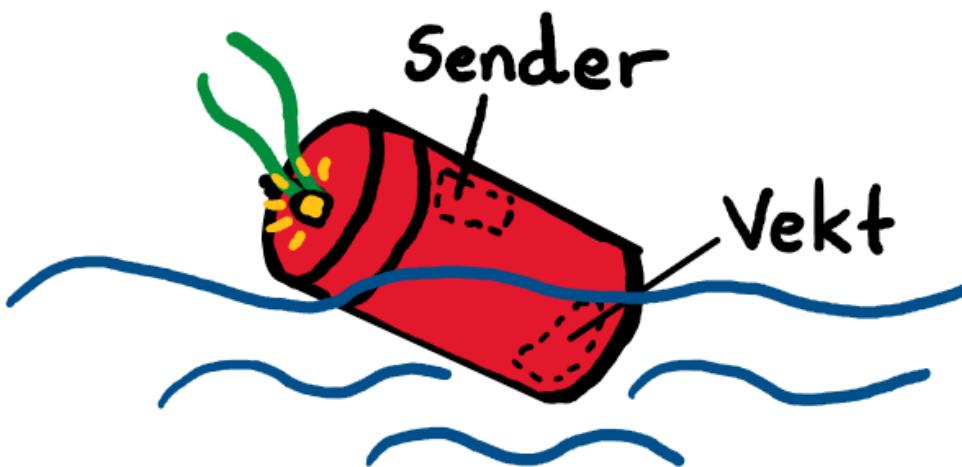
Figur 3.2: Overordnet systemflyt for FISC.

### 3.1 Kapsel

Senderen skal plasseres i en kapsel som skal være vanntett og robust, og på den måten oppfylle systemkrav 1 og 2. Kapselen må da være stor nok til å romme senderen, og være laget av et vanntett materiale som kan tåle slitasje.

For at signalet skal kunne gå langt må systemet ligge ved vannoverflaten slik at det ikke trenger å reise gjennom vann. Det er derfor viktig at systemet ligger i en kapsel som flyter, og oppfyller systemkrav 3 og 4. For at dette skal oppnås må Arkimedesloven brukes. Den sier at "En gjenstand som er helt eller delvis nedsenket i en væske eller gass, får en oppdrift lik tyngden av den væskemengden som gjenstanden fortrenger" [14]. Dette betyr at for at kapselen skal flyte, må den ha en gjennomsnittlig massetetthet mindre enn væsken den flyter i [14].

Vann har en massetetthet på  $999.87 \text{ kg/m}^3$  ved  $0^\circ\text{C}$  og en massetetthet på  $997.04 \text{ kg/m}^3$  ved  $25^\circ\text{C}$ . [15, s. 2].  $0^\circ\text{C}$  til  $25^\circ\text{C}$  er temperaturintervallet som vannet rundt merder ideelt holder, og dersom kapselens gjennomsnittlige massetetthet blir noe mindre enn vannets, vil kapselen flyte. For at signalet skal sendes er det viktig at selve senderen er over vann, og massetettheten må derfor være tilstrekkelig liten for at dette skal skje. Dette kan oppnås enklere ved at senderen er festet mot toppen av kapselen. I tillegg burde det være størst vekt nederst i kapselen slik at den holder seg balansert og ikke snur, for å unngå at senderen går under vann. Se Figur 3.3.

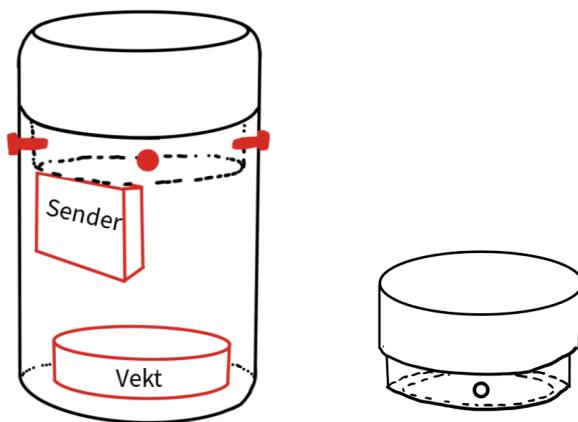


Figur 3.3: Design for hvordan en vekt skal holde senderen stabilt over vann.

Formen på kapselen vil derfor være en avlang sylinder, slik som vist i Figur 3.4 og Figur 3.3. Da vil vekten i bunn hjelpe kapselen med å holde seg stabil, men senderen vil

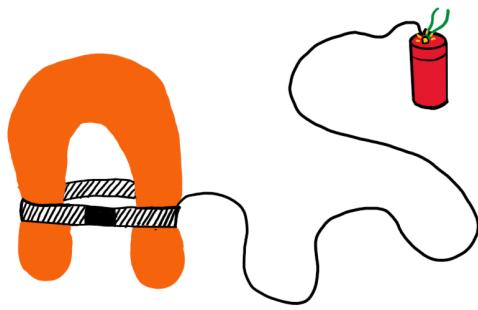
være festet nærmere toppen slik at den flyter over vann, og på denne måten hjelpe til med å oppnå systemkrav 3.

Som vist i Figur 3.5 skal kapselen ha et lokk slik at den skal være vanntett, men fortsatt være mulig å åpne opp for vedlikehold og dermed oppfylle systemkrav 6. Lokket vil derfor ha en forlengelse på innsiden av kapselen slik at det skal være mulig å feste. Dette vil da føre til at senderen må plasseres lenger ned i kapselen. Da må volumet til kapselen økes, slik at den totale massetettheten blir mindre. Dette er for å passe på at systemkrav 3 og 4 fortsatt blir oppfyllt. Da vil kapselen flyte lenger opp og senderen fortsatt holde seg over vannoverflaten. Likevel er det viktig at volumet ikke blir for stort slik at kapselen er forstyrrende for brukeren og ikke kan oppfylle systemkrav 5.



Figur 3.4: Enkelt design for kapselen.      Figur 3.5: Enkelt design av kun lokket.

Som nevnt i systemkrav 7 i Tabell 2.1 skal kapselen være festet til redningsvesten via en snor. Kapselen blir derfor designet til å ha en snor festet i lokket, mens den andre enden av snoren festes i selve redningsvesten, som vist i Figur 3.6. På denne måten vil ikke kapselen bli dratt under vann av personen, i tillegg til at posisjonen til senderen konstant vil følge bevegelsen til personen.

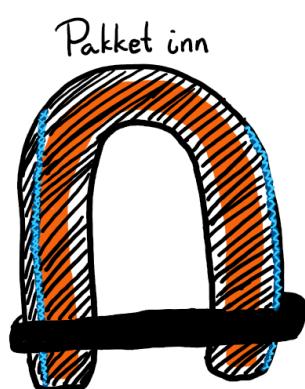


Figur 3.6: Kapselen er festet til redningsvesten med en snor etter at systemet er utløst.

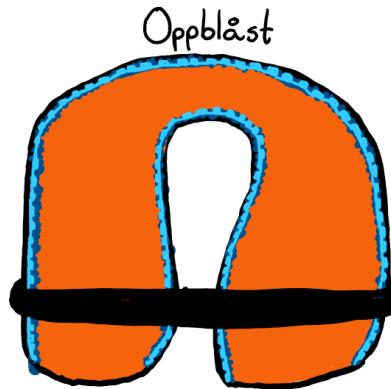
### 3.2 Utløsermekanisme

Senderen skal, som nevnt, sende et signal ved kontakt med vann og ved manuell utløsning. Dette oppnås ved å feste utløsermekanismen til redningsvesten, slik at signalet kun sendes når redningsvesten utløses manuelt eller brukeren har havnet i vannet, og ikke f.eks. når det regner. På denne måten kan systemkrav 9 og 10 oppfylles.

En utløsbart redningsvest er konstruert slik at en flytevest er pakket inn i en stofflomme, og lukket igjen med glidelås. Når redningsvesten utløses blåses flytevesten opp med en slik kraft at glidelåsen brytes opp og lommen åpner seg, som vist i Figur 3.7 og Figur 3.8.



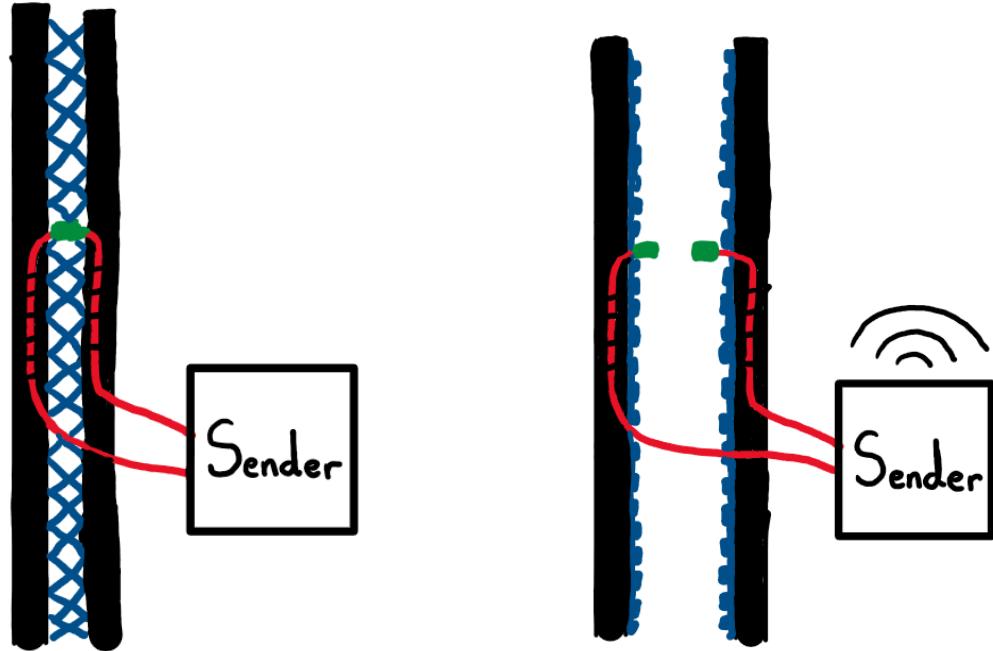
Figur 3.7: Flytevesten er pakket inn i en stofflomme og glidelåsen er lukket.



Figur 3.8: Flytevesten er blåst opp og har revet sidene av glidelåsen fra hverandre.

Ved å feste en ledning på hver side av glidelåsen, og deretter koble de sammen, vil ledningene kobles fra hverandre når glidelåsen brytes opp, som vist i Figur 3.9 og

Figur 3.10. På denne måten vil tilstanden til utløsermekanismen endres, og deretter gi beskjed til senderen.



Figur 3.9: Glidelåsen er lukket og ledningene koblet sammen, slik at det ikke sendes noe signal.

Figur 3.10: Glidelåsen er åpen og ledningene er koblet fra hverandre, slik at det sendes et signal.

### 3.3 Kommunikasjon: LTE-M og MQTT

Med tanke på design av senderen, alarmsystemet og webapplikasjonen må senderen ha evne til å trådløst kommunisere sin lokasjon til eventuelle mottakere og nødetatene. For å trådløst kommunisere, må det sendes via basestasjoner eller satellitter. Senderen, alarmsystemet og webapplikasjonen vil bruke basestasjoner for enklere implementering. Integrerte kretser som kommuniserer over nettet faller under kategorien Tingenes internett (IoT). IoT-enheter som kommuniserer med mobilnett, gjør det enten med *Long-Term Evolution Machine Type Communication* (LTE-M) eller *Narrowband Internet of things* (NB-IoT).

En kritisk faktor for FISC er at en person som har falt over bord vil bevege seg på grunn av strømning og bølger i vannet. Lokasjonen, som hentes via et globalt navigasjonssatellittsystem (GNSS) [16], må derfor kunne sendes fra en enhet i bevegelse. Mens NB-IoT

primært er designet for stasjonære enheter, er LTE-M utviklet for å håndtere mobilitet. Derfor falt valget på LTE-M som kommunikasjonsteknologi.

Siden FISC er tiltenkt å brukes langs kysten ut til fiskemerdene er det avgjørende å verifisere at den valgte teknologien, LTE-M, har tilstrekkelig dekning. Figur 3.12 viser mobildekningen i Norge, mens Figur 3.11 viser fiskemerdene i Norge. Som figurene indikerer er det en fullstendig overlapp mellom dekningen og fiskemerdene. Dette validerer bruken av LTE-M og oppfyller systemkrav 8.

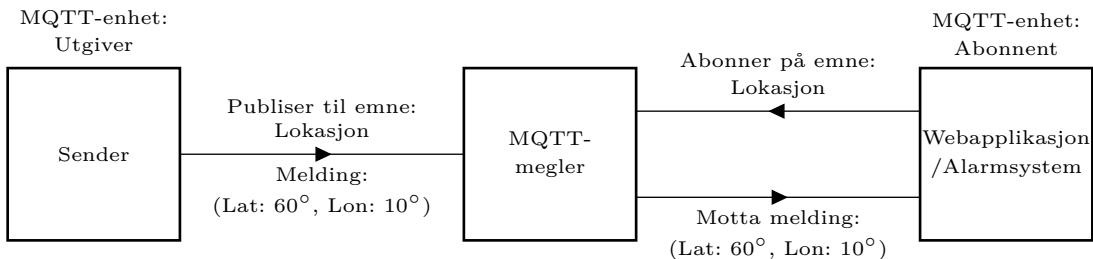


Figur 3.11: Fiskemerder i Norge [17].



Figur 3.12: Mobildekning i Norge [18].

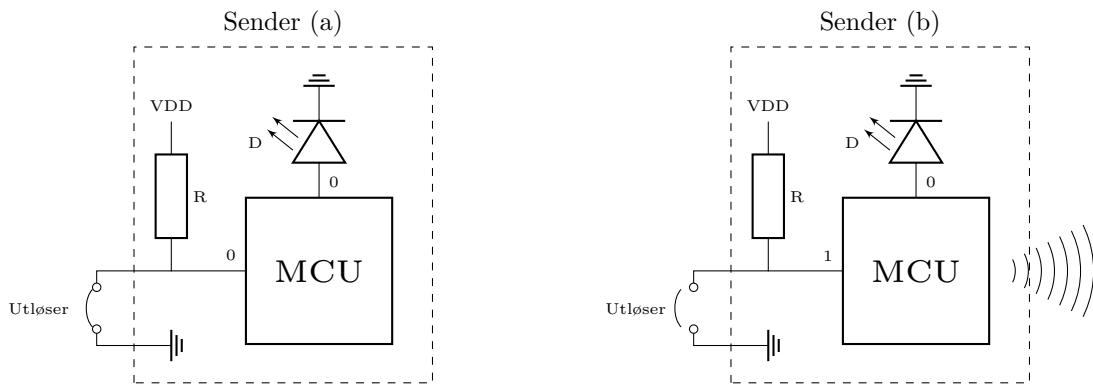
For å håndtere selve dataoverføringen av lokasjonsinformasjon (fra GNSS) over LTE-M-nettverket, benyttes *Message Queue Telemetry Transport* (MQTT) [19]. MQTT er en meldingsprotokoll som bruker publisere/abonnere-modell. Dette betyr at man kan ha flere enheter som publiserer meldinger på et emne, og andre enheter som abонnerer på det emnet, som mottar meldingene. For å kunne gjøre dette trengs det en MQTT-megler som mottar og distribuerer meldinger mellom utgiver og abonnent, som vist i Figur 3.13.



Figur 3.13: Prinsippskisse for MQTT publisera/bonusser-arkitektur.

### 3.4 Sender

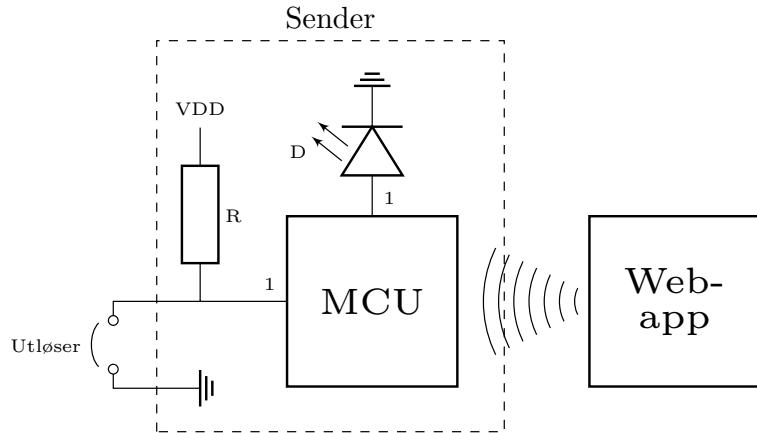
For å designe en sender som kan kommunisere gjennom LTE-M, trengs det en mikrokontroller (MCU) i senderen som kan kommunisere gjennom LTE-M. Denne MCU-en skal starte å publisere posisjoner til et MQTT-emne ved en åpen krets, men ved kortslutning av utløsermekanismen skal senderen ikke sende noe signal. Dette kan gjøres med en opptrekksmotstand som vist i Figur 3.14.



Figur 3.14: Opptrekksmotstand i senderdesign: Utløseren i Sender (a) er på, og MCU får dermed inn spenning lik jord (0). I Sender (b) er utløseren av, og VDD har ingen lett vei til jord; da vil spenningsfallet over R være tilnærmet null (gitt  $R_{inn, MCU} \gg R$ ) og spenningen inn i MCU-en være tilnærmet VDD (1).

Som illustrert i Figur 3.14 vil MCU ikke sende noe signal når utløseren er festet sammen. MCU-en vil starte å sende signal når ledningen koblet til enheten er åpen, slik som i Figur 3.10. Dermed blir signalet sendt når en person faller i sjøen og redningsvesten blåses opp, eller når noen utløser redningsvesten manuelt. Dette oppfyller systemkrav 9 og 10.

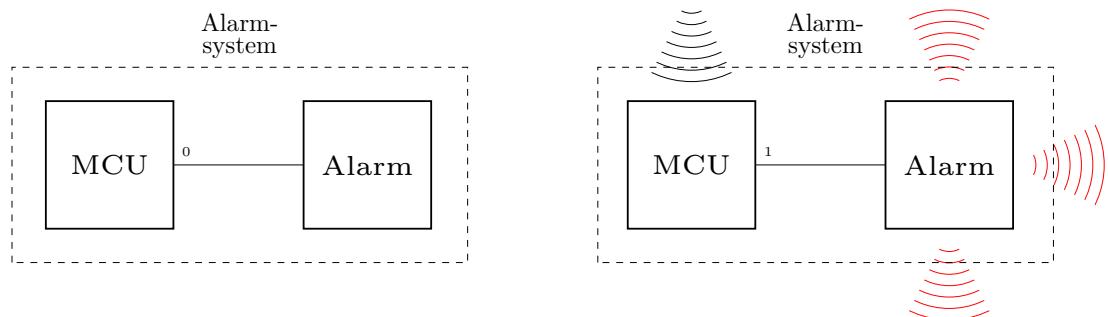
For å oppfylle systemkrav 12 må det designes et lys som lyser opp når webapplikasjonen mottar posisjonen. Senderen skal abonnere på et MQTT-emne som webapplikasjonen også publiserer på. I det webapplikasjonen mottar posisjoner vil den publisere på det emnet og senderen vil motta en melding og lyse opp. Dette er vist i Figur 3.15.



Figur 3.15: Sender skrur på lyset (D) når den motter signal fra webapplikasjonen.

### 3.5 Alarmsystem

Alarmsystemet står for varsling når noen faller over bord. Når alarmsystemet mottar et signal fra senderen, skal utgangen bli høy og skru på alarmsystemet. Dette er vist i Figur 3.16.



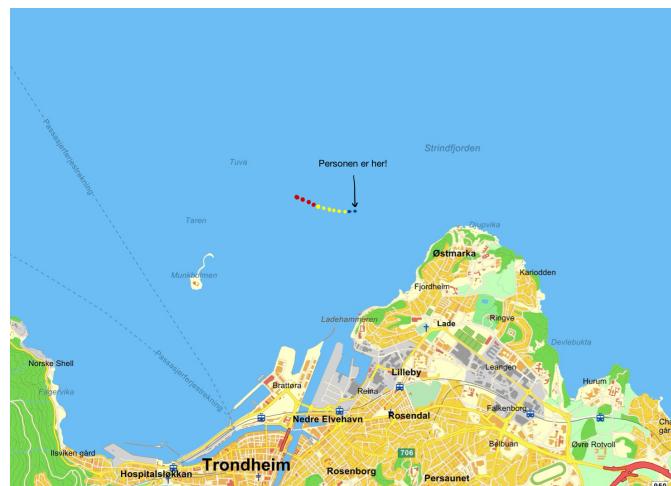
Figur 3.16: Prinsipp for mottaker som aktiverer lokalt alarmsystem ved mottak av MQTT-melding.

Med tanke på systemkrav 13 og 14 blir alarmsystemet designet med en hørbar alarm og et blinkende lys.

### 3.6 Webapplikasjon

Webapplikasjonen er grensesnittet for nødetatene, og skal vise posisjonen til senderen når utløst. Funksjonaliteten til webapplikasjonen kan forklares på følgende måte:

- Applikasjonen abоннерerer på et MQTT-emne for å motta posisjonsdata.
- I det webapplikasjonen mottar en gyldig posisjon vil den publisere et ”skru på lys”-melding på MQTT-emnet som senderen er abonnert på for å oppfylle systemkrav 12.
- Posisjonen(e) vises som markører på et interaktivt kart.
- Kartet oppdateres periodevis for å reflektere bevegelse som vil oppfylle systemkrav 15.
- For å gi et tidsbilde av hendelsen, kan markørene fargekodes basert på hvor gamle posisjonsdataene er, slik som skissert i Figur 3.17. Dette hjelper redningsmannskapet å se bevegelsesmønsteret.



Figur 3.17: Konseptskisse for webapplikasjon med kartvisning og fargekodede posisjonsmarkører [20]

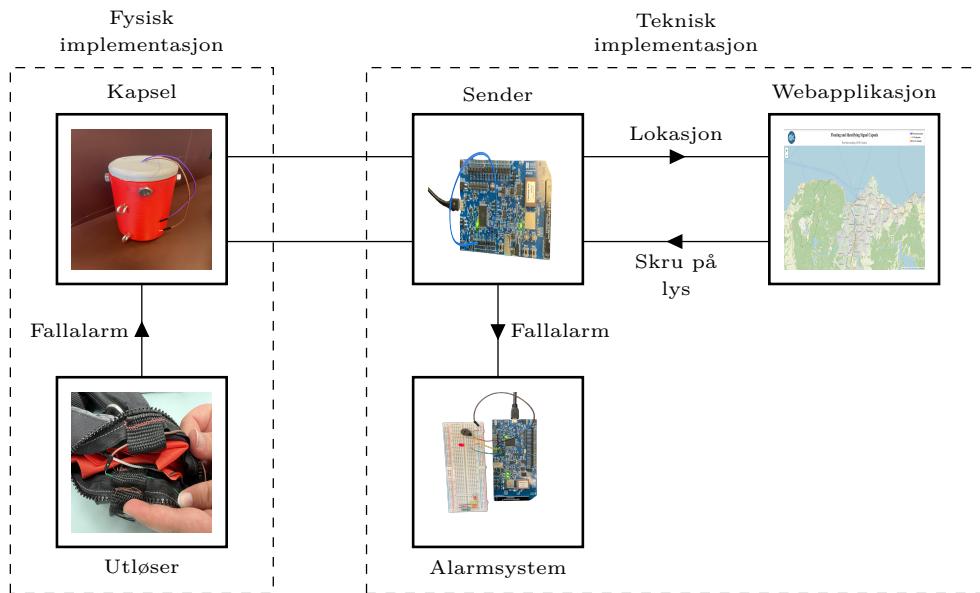
# 4 Implementering

## 4.1 Fysisk utforming

Prototypen ble satt sammen med følgende hovedkomponenter:

- **Kapsel:** 3D-printet PETG-materiale [21] og fiskesnøre.
- **Sender:** nRF9160 Development Kit (DK) [22].
- **Utløsermekanisme:** Sydd stoff-feste med ledninger på redningsvesten.
- **Alarmsystem:** nRF9160 Development Kit (DK) [22], OSOYOO-Buzzer [23], standard LED, BS170-transistor [24].
- **Webapplikasjon:** Flask [25] og Leaflet [26] kjørende på en bærbar datamaskin.

Figur 4.1 viser systemene brukt til å implementere FISC.



Figur 4.1: Systemflyten deles opp i fysisk og teknisk implementasjon. I videreutvikling ville begge implementasjonene integreres til ett produkt.

Figur 4.2 viser FISC montert på en bruker. Ledningene til utløsermekanismen er festet inni vesten, og kobles til kapselen. Kapselen beskytter senderen, som er plassert inni. Se Figur 3.4.



Figur 4.2: Bilde av hvordan FISC ser ut ved bruk.

Figur 4.1 viser at komponenten brukt for å realisere senderen var en nRF9160 DK [22]. Kapselen blir designet for å bruke en Nordic Thingy:91 [27] som senderen, da dette prototypebrettet er batteridrevet, kan dele lokasjon via GNSS, og er liten i størrelse. Likevel blir en nRF9160 DK brukt for implementasjonen av senderen i stedet. Dette ble bestemt fordi nRF9160 DK er enklere å utvikle. For eksempel, i forhold til bruk av *General-Purpose Input/Output Pins* (GPIO-pinnene), krevde implementeringen irreversible modifikasjoner av Thingy:91, som ikke var ønskelig å utføre ved utvikling av prototypen. nRF9160 DK kan fortsatt dele lokasjon via GNSS. I tillegg har begge brettene samme *system on a chip* (SOC)[28], som betyr overføring av kode mellom Thingy:91 og nRF9160 DK kun krever små modifikasjoner.

## 4.2 Kapsel

Implementering av kapsel er gjort med bakgrunn i systemkravene vist i ??.

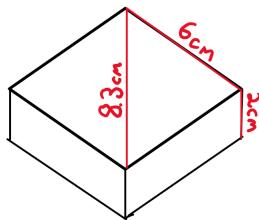
Nr. Systemkrav

Oppfyller: Brukerkrav

- 1 Senderen i systemet skal beskyttes av en kapsel. 2, 5
- 2 Kapselen som senderen beskyttes av burde ikke slippe inn vann. 2
- 3 Kapselen skal flyte stabilt, med senderen over vann. 2

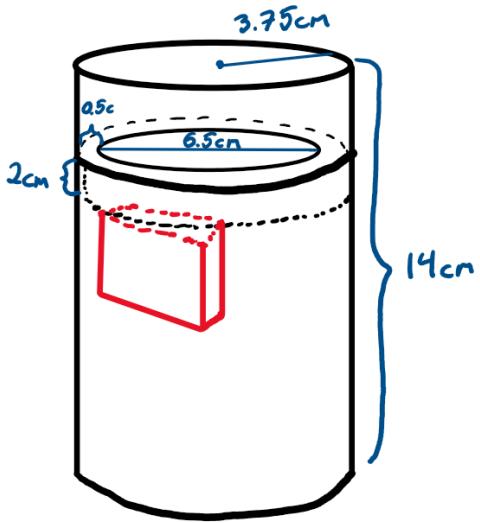
- 4 Kapselen med senderen skal ha en massetethet mindre enn vann. 2, 6
- 5 Kapselen med senderen burde sitte komfortabelt på arbeidernes kropp, og ikke begrense daglig bevegelse 4
- 6 Kapselen med senderen burde være monterbar og avtagbar innen en tidsramme på 5 minutter, utført med hendene 5, 6
- 7 Etter utløsning skal kapselen være festet til redningsvesten via en snor på minst 2 meter. 2

For å oppfylle systemkrav 1 må kapselen ha plass til Thingy:91. Som vist i Figur 4.3 har Thingy:91 sider på 6 cm, høyde på 2 cm og diagonaler på 8.3 cm.



Figur 4.3: Mål på Nordic Thingy:91.

For at systemkrav 5 skal opprettholdes er det viktig at kapselen ikke er for stor, slik at den forstyrrer brukeren i sitt arbeid. Senderen blir derfor plassert på siden inni kapselen. Da kan den innvendige diameteren være så liten som mulig, og settes til 6.5 cm. For å gjøre kapselen mer robust og vanntett blir veggene satt til å være 0.5 cm tykke. Dette vil være med på å oppfylle systemkrav 1 og 2. For at systemkrav 3 og 4 skal oppfylles når kapselen skal inneholde vekt, er det viktig at volumet til kapselen er tilstrekkelig stort. De ferdige målene på kapselen blir da, som vist i Figur 4.4, høyde = 14 cm og radius = 3.75 cm. Forlengelsen på lokket ble på 2 cm. Dette var for å oppfylle systemkrav 2 og 6. Lokket ble da festet med fire skruer og muttere, se Figur 4.5, slik at det ved nødvendighet skal være mulig å åpne kapselen, men ellers vil være festet godt.



Figur 4.4: Tegning av kapsel med ferdig mål. Figur 4.5: Bilde av den ferdige kapselen.

For å implementere kapselen brukes 3D-printing. På denne måten kan kapselen lages med ønsket form og mål. For mer beskyttelse ble det valgt et materiale av typen PETG, da dette skal være slitesterkt, vannresistent og tåle forskjellige temperaturer [21]. I tillegg er materialet relativt lett, med en massetetthet på  $1270 \text{ kg/m}^3$  [29], noe som vil gjøre det enklere for kapselen å flyte og dermed oppfylle systemkrav 3.

Kapselen er festet med et fiskesnøre, på 2.5 m, til lokket og til vesten. Dette er vist i Figur 4.6. Den midtre delen av snøret er surret rundt kapselen. For at den skal sitte godt og ikke forstyrre brukeren, som er viktig for systemkrav 5, tres fiskesnøret gjennom øyekrokene vist i Figur 4.5. Videre er snøret festet i glidelåsen. Dermed vil delen av tråden som holder kapselen inntil vesten løsne når systemet utløses slik at kapselen vil flyte fritt, med kun enden på tråden festet til vesten. Se Figur 3.6. På denne måten kan systemkrav 7 oppfylles.



Figur 4.6: Figuren viser hvordan fiskesnøret er brukt til å holde kapselen på plass på vesten.

Vekten i bunnen av kapselen må være større enn vekten til senderen, slik at tyngden havner i bunnen og systemkrav 3 kan oppfylles. Thingy:91 har, etter måling, en vekt på 61 g. For ekstra stabilitet, men tilstrekkelig lite vekt, brukes det derfor et lodd på 250 g. Dette er festet, som vist i Figur 4.7, på siden i bunnen av boksen slik at den skal flyte som i illustrasjonen i Figur 4.1.



Figur 4.7: Loddet festet med gaffateip inni kapselen.

Med målene vist i Figur 4.4 får kapselen et volum på  $618.5 \text{ cm}^3$ . For at Systemkrav 4 skal oppfylles, med dette volumet, må den totale vekten til kapselen være mindre enn  $618.5 \text{ g}$ . Det er fordi massetettheten til vann er ca.  $1000 \text{ kg/m}^3$  [15], og  $1000 \text{ kg/m}^3 = 1 \text{ g/cm}^3$ . Så massen må dermed være lik volumet.

### 4.3 Utløsermekanisme

Ved implementering av utløsermekanismen burde systemkravene i Tabell 4.1 oppfylles.

Tabell 4.1: Systemkravene til utløsermekanismen.

Nr.	Systemkrav	Oppfyller: Brukerkrav
7	Etter utløsning skal kapselen være festet til redningsvesten via en snor på minst 2 meter.	2
9	Senderen skal ikke sende signal før det utløses manuelt.	3
10	Senderen skal ikke sende signal før det kommer i kontakt med vann.	3

Figur 4.8 viser hvordan ledningene er festet til glidelåsen. En stoffbit er sydd fast på hver side av glidelåsen, hvor ledningene er tredd gjennom, med god nok plass slik at de lett sklir ut og ledningene ikke blir sittende fast i vesten når den utløses. Dette er viktig for at kapselen skal kunne slippes vekk fra brukeren slik at systemkrav 7 kan oppfylles.



Figur 4.8: Realisering av utløsermekanismen, viser hvordan ledningene er festet på innsiden av vesten.

Stoffet er av samme stoff som på remmen til redningsvesten, slik at festet er robust og ikke rives av ved utløsning. Dette er viktig for at utløsermekanismen skal fungere som designet, og dermed oppfylle systemkrav 9 og 10.

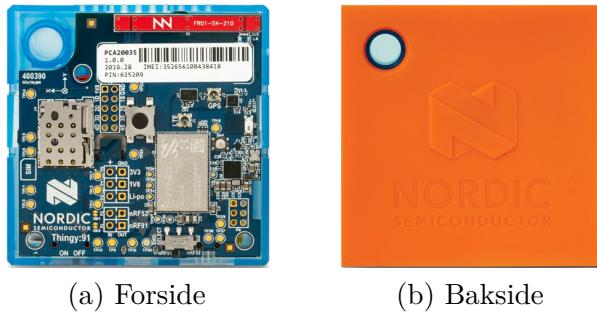
## 4.4 Sender

Når senderen realiseres, er det gjort med bakgrunn på systemkravene i Tabell 4.2.

Tabell 4.2: Systemkravene til senderen.

Nr.	Systemkrav	Oppfyller:	Brukerkrav
8	Senderens evne til å sende signal burde ikke svekkes i vannet sammenlignet med på land.	1, 2	
9	Senderen skal ikke sende signal før det utløses manuelt.	3	
10	Senderen skal ikke sende signal før det kommer i kontakt med vann.	3	
11	Senderen skal ikke dele posisjon i kartet før systemet er blitt utløst.	1, 3	
12	Senderen skal begynne å lyse når webapplikasjonen har mottatt posisjonen.	1	

Nordic Thingy:91[27] var prototypebrettet som skulle benyttes til å kommunisere med mottakeren. Figur 4.9 presenterer Nordic Thingy:91.



(a) Forside (b) Bakside

Figur 4.9: Nordic Thingy:91 [27].

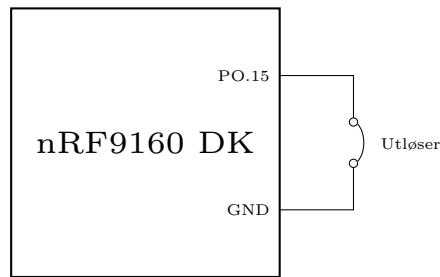
Som forklart og argumentert for i seksjon 4.1, ble det bestemt å bruke en nRF9160 DK [22] for implementeringen av senderen. Figur 4.10 presenterer utviklingsbrettet.



Figur 4.10: nRF9160 DK [22]

Som forklart, kan nRF9160 DK dele lokasjon via GNSS, som fører til at senderens evne til å sende signal ikke er avhengig av om kapselen er i vannet eller ikke. Det er kun avhengig av om senderen ligger innenfor mobildekningssområdet presentert i Figur 3.12, så lenge kapselen virker som den skal og sørger for at senderen ligger over vannet. Systemkrav 8 er dermed tatt hensyn til.

Figur 4.11 viser senderen sin konfigurasjon, hvor PO.15 er en av GPIO-pinnene til nRF9160 DK, og GND er den innebygde ground i brettet.



Figur 4.11: Krettskjema til sender, med en pull-up konfigurasjon kodet på PO.15 som vist i Figur 3.14.

Ledningen mellom PO.15 og GND representerer ledningen i utløsermekanisme i Figur 3.14. Når ledningene kobles fra hverandre, publiserer senderen et varsel til alarmsystemet og webapplikasjonen gjennom emnet `gruppe10/elsysprosjekt/hovedtopic` innenfor MQTT-verten `test.mosquitto.org` [30]. Alarmsystemet og webapplikasjonen er abonnerert på samme emne. På denne måten er systemkrav 9 og 10 oppfylt, da senderen kun sender signal når ledningen i utløsermekanismen er utløst.

Publiseringen gjøres i `main.c` til senderen, som presenteres i seksjon D.1. Først konfigureres det en opptrekksmotstand på PO.15 som forklart i Figur 3.14, slik at når PO.15 ikke er koblet til jord som i Figur 4.11, får PO.15 et høyt signal. Konfigurasjonen skjer i `int main(void)` funksjonen med kodelinjen vist i Listing 4.1.

Listing 4.1: Kode-snippet, fra senderen sin main.c

```
// Konfigurer \gls{GPIO} som utgang
gpio_pin_configure (gpio_dev , INPUT_PIN ,
GPIO_INPUT | GPIO_PULL_UP );
```

int main(void) kaller da `check_input()` funksjonen, som sjekker om PO.15 er et høyt signal. Hvis den er det, blir det sendt en kommando til emnet som varsler alarmsystemet om å skru på sitt innebygde LED-lys. Listing 4.2 presenterer denne logikken.

Listing 4.2: Kode-snippet, fra `check_input()` funksjonen

```
if (!led1_checked) {
    led1_checked = true;
    int err = data_publish( & client ,
        MQTT_QOS_1_AT_LEAST_ONCE ,
        CONFIG_BUTTON_EVENT_PUBLISH_MSG ,
        sizeof(CONFIG_BUTTON_EVENT_PUBLISH_MSG) - 1 );
    if (err) {
        LOG_INF("Failed to send message , %d", err);
        return;
    }
}
```

Listing 4.3 viser at videre blir det sendt en simulert posisjon i bevegelse til emnet. Webapplikasjonen henter og plotter dette kartet. På denne måten er systemkrav 11 oppfylt, da det kun genereres og publiseres posisjoner etter PO.15 er høyt ved at senderen er utløst.

Listing 4.3: Kode-snippet, fra `check_input()` funksjonen

```
generate_infinity_position (&lat , &lon);
...
...

//Lager en JSON-streng for posisjonen
char json_str[100];
int lat_i = (int)(lat * 1e6);
int lon_i = (int)(lon * 1e6);

snprintf(json_str, sizeof(json_str),
    "{\"lat\": %d.%06d, \"lon\": %d.%06d}",
```

```

        lat_i / 1000000, abs(lat_i % 1000000),
        lon_i / 1000000, abs(lon_i % 1000000));

    int err = data_publish(&client,
                           MQTT_QOS_1_AT_LEAST_ONCE,
                           json_str, strlen(json_str));

    if (err) {
        LOG_INF("Feil ved sending av posisjon: %d",
                err);
    } else {
        LOG_INF("Sendte ø-posisjon: %s", json_str);
    }
    k_sleep(K_MSEC(2000));

```

Senderens nRF9160 DK er også abonnert på gruppe10/elsysprosjekt/sendersub som webapplikasjonen publiserer til. Webapplikasjonen varsler til senderen om å skru på sitt innebygde LED-lys når den begynner å få posisjonsdata, slik presentert i Figur 3.15. Dette oppfyller systemkrav 12. Emnene senderen publiserer og abоннерер på er bestemt i prj.conf til senderen, som vist i Listing 4.4. Hele prj.conf er presentert i seksjon D.2.

Listing 4.4: Kode-snippet, fra check\_input() funksjonen

```

#Application
CONFIG_MQTT_PUB_TOPIC="gruppe10/elsysprosjekt/hovedtopic"
CONFIG_MQTT_SUB_TOPIC="gruppe10/elsysprosjekt/sendersub"

```

Figur 4.12 demonstrerer den realiserte Figur 4.11.



Figur 4.12: Sendersystemet realisert. En nRF9160 DK med en ledning mellom PO.15 og GND.

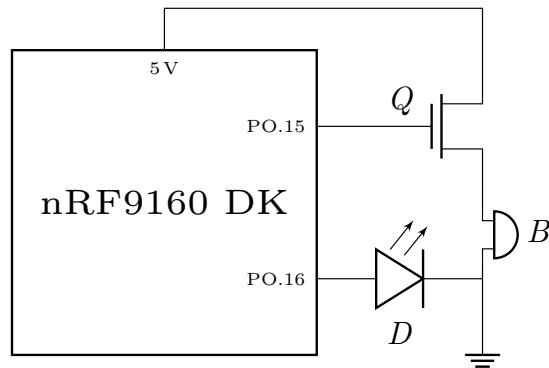
## 4.5 Alarmsystem

Implementering av alarmsystemet er gjort med bakgrunn i systemkravene i Tabell 4.3.

Tabell 4.3: Systemkravene til alarmsystemet.

Nr.	Systemkrav	Oppfyller:	Brukerkrav
13	Alarmsystemet skal utløse en hørbar alarm innen 5 sekunder etter at senderen er blitt utløst i vann.		1, 2, 3
14	Alarmsystemet skal utløse en hørbar alarm innen 5 sekunder etter manuell utløsning.		1, 2, 3

Figur 4.13 viser hvordan alarmsystemet er implementert



Figur 4.13: Kretsskjema til alarmsystem.

nRF9160 DK er valgt som enheten i alarmsystemet som kommuniserer med senderen sin nRF9160 DK, fordi kommunikasjon mellom disse er enkel å implementere [22].

Da man ønsker at alarmsystemet skal reagere med lyd og lys, har vi en lysdiode  $D$  og buzzer  $B$  som skal aktiveres når senderen varsler om utløsning, slik presentert i Figur 3.16. I tillegg er det en transistor  $Q$  som gir buzzeren 5 V når PO.15 gir et høyt signal. Dette er fordi buzzeren som brukes er en OSOYOO-buzzer, som krever 4-8V for å tolke et signal som høyt [23]. Spenningen GPIO-pinnene leverer er kun 3.3 V [22].



Figur 4.14: OSOYOO-buzzer.

Utgangspinnene PO.15 og PO.16 gir et varierende signal ved utløsning, fordi det er ønskelig at *D* skal være et blinkende lys, og *B* krever et varierende signal for å skape lyd [23]. *Q* er en BS170-transistor [24].

Logikken bak hvordan PO.15 og PO.16 sender spenning er funnet i `main.c` til alarmen, som vist i seksjon C.1. Funksjonen `check_alarm()` blir kalt i `int main(void)`, hvor den først tester om det innebygde LED-lyset er på. Hvis den er det, betyr det at senderen er utløst. Da leverer PO.15 og PO.16 et varierende signal, hvor funksjonen `k_timer_start()` sørger for at utgangspinnene bytter mellom å sende høyt og lavt signal. Listing 4.5 presenterer denne logikken. *B* bytter mellom høy og lav spenning hvert 500 µs, mens *D* bytter hvert 300 ms.

Listing 4.5: Kode-snippet, fra `check_alarm()` funksjonen

```

if (input_state == 1 && !blinking) {
    blinking = true;
    int ret = gpio_pin_set(gpio_dev, INPUT_PIN, 1);
    if (ret < 0) {
        printk("Failed to set pin %d high (ret = %d)\n",
               INPUT_PIN, ret);
    }

    int ret2 = gpio_pin_set(gpio_dev, INPUT_PIN2, 1);
    if (ret2 < 0) {
        printk("Failed to set pin %d high (ret = %d)\n",
               INPUT_PIN2, ret2);
    }

    //Starter alarm, lyd og lys
    k_timer_start( & blink_timer_alarm, K_USEC(500),
                   K_USEC(500));
    k_timer_start( & blink_timer_lyd, K_MSEC(300),
                   K_MSEC(300));
}

```

```

    k_timer_start( & blink_timer_diode , K_MSEC(300) ,
    K_MSEC(300));

    printk("Input is HIGH: starting LED blink\n");
}

```

Systemkrav 13 og 14 er da tatt hensyn til både i `main.c` til senderen og `main.c` til alarmen. Ved at `check_input()` funksjonen kalles kontinuerlig i senderens `int main(void)`, sørges det for at senderen varsler til mottakeren ved utløsning så raskt som tillatt. Samtidig kalles `check_amarm()` funksjonen kontinuerlig i mottakerens `int main(void)`, slik at mottakeren reagerer på varslingen så raskt som tillatt.

Emnet alarmsystemet abonnerer til er bestemt i `prj.conf` til alarmsystemet, som vist i Listing 4.6. Hele `prj.conf` er presentert i seksjon C.2.

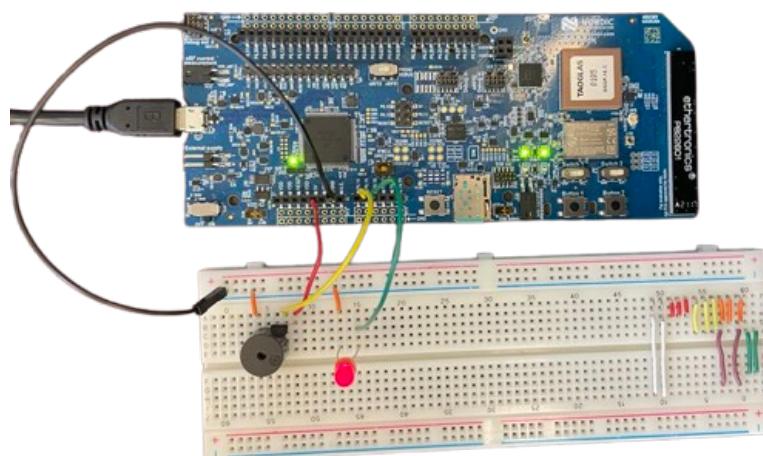
Listing 4.6: Kode-snippet, fra `check_input()` funksjonen

```

CONFIG_MQTT_SUB_TOPIC =
"gruppe10/elsysprosjekt/hovedtopic"

```

Figur 4.15 demonstrerer den realiserte Figur 4.13.



Figur 4.15: Det realiserte alarmsystemet med en svart buzzer og en rød lysdiode.

## 4.6 Webapplikasjon

Realisering av webapplikasjonen er gjort med bakgrunn i systemkravet i Tabell 4.4.

Tabell 4.4: Systemkravene til Webapplikasjonen.

Nr.	Systemkrav	Oppfyller:	Brukerkrav
15	Posisjonen til senderen skal oppdateres regelmessig innen hvert 5. sekund.		1

For å vise posisjonen til kapselen på en webapplikasjon bruker `server.py` et Python-basert web-rammeverk, Flask [25], for å kontinuerlig motta posisjonsdata fra MQTT-emnet `gruppe10/elsysprosjekt/hovedtopic`. `server.py` er presentert i seksjon B.1.

Videre er posisjonsdataene gjort tilgjengelig til `index.html`, som lager brukergrensesnittet til applikasjonen. Det blir gjort tilgjengelig via en Flask-rute i `server.py`, som utløser funksjonen `get_location()`, som videre returnerer den nyeste posisjonsdataen hentet fra emnet i *JavaScript Object Notation* (JSON)-format. Listing 4.7 viser dette.

Listing 4.7: Kode-snippet, fra server.py

```
@app.route("/location", methods=["GET"])
def get_location():
    return jsonify(latest_location)
```

Seksjonen `<head>` i `index.html` inneholder metainformasjon om webapplikasjonen og hjelper til å konfigurere hvordan webapplikasjonen skal oppføre seg. Det er her JavaScript-biblioteket Leaflet [26] blir inkludert. `<body>` har informasjon om utseendet og oppførselen til webapplikasjonen, hvor Leaflet brukes til å konstruere og vise det interaktive kartet. Hele `index.html` kan finnes i seksjon B.2.

I seksjonen `<body>` hentes posisjonsdataene fra `server.py` gjennom Flask-ruten i Listing 4.7. Listing 4.8 viser dette hvor, dersom senderen er utløst, den nyeste posisjonsdataen senderen publiserte hentes hvert 2. sekund. Da oppdaterer kartet seg slik at brukeren kan se den posisjonen, i tillegg til de tidlige posisjonene som har vært.

Listing 4.8: Kode-snippet, fra index.html

```
// Hent posisjon fra serveren hvert 2. sekund
setInterval(function() {
    fetch('/ location ')
        .then(response => response.json())
        .then(data => {
            updateLocation(data.lat, data.lon);
        })
        .catch(error => console.error('Error
fetching
location: ', error));
}, 2000);
// Oppdater farger på eldre markører hvert
// 2. sekund
setInterval(updateMarkerColors, 2000);
```

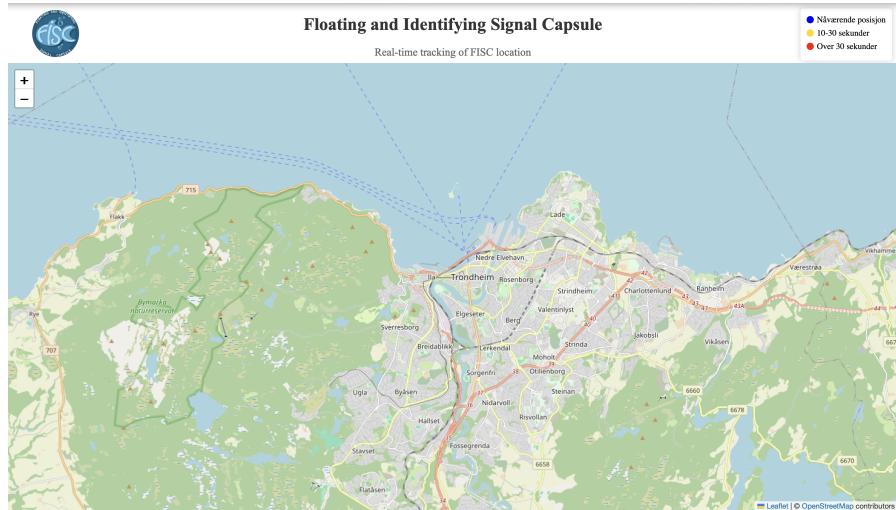
Systemkrav 15 er dermed oppfylt. Som presentert i Listing 4.3, sørger funksjonen `k_sleep(2000)` for at senderen publiserer ny posisjonsdata hvert 2. sekund til emnet `gruppe10/elsysprosjekt/hovedtopic`. I tillegg var det nettopp forklart at den nyeste posisjonsdataen publisert i emnet blir hentet hvert 2. sekund, som vist i Listing 4.8. Da sørges det for at posisjonen oppdateres innen hvert 5. sekund.

Når senderen er utløst, varsler også `server.py` til senderen å skru på sitt innebygde LED-lys, for å oppfylle senderens systemkrav 12. Dette skjer i `on_message()` funksjonen, under if-koden presentert i Listing 4.9.

Listing 4.9: Kode-snippet, fra server.py

```
if not has_sent:
    #Hvis flagget has_sent er False, sender vi en
    #melding til gruppe10/elsysprosjekt/sendersub
    print("[DEBUG] Sender 'LED1ON' til
          gruppe10/elsysprosjekt/sendersub")
    client.publish("gruppe10/elsysprosjekt/sendersub",
                  payload="LED1ON", qos=1, retain=False)
    has_sent = True
else:
    print("[DEBUG] JSON er ikke en gyldig posisjon.")
```

Et bilde av webapplikasjonen er vist i Figur 4.16. Som vist øverst til høyre på nettsiden i Figur 4.16 er posisjonen fargekodet i forhold til hvor lenge siden personen var i den sendte posisjonen.



Figur 4.16: Bilde av webapplikasjonen før sendt posisjon.

Figur 4.17 demonstrerer hvordan webapplikasjonen endrer seg da den får posisjonskoordinater.



Figur 4.17: Bilde av webapplikasjonen etter sendt posisjon.

De blå lokasjonspinnene er den nåværende posisjonen, de gule lokasjonspinnene er koordinater som ble plassert på kartet 10–30 sekunder siden, og de røde lokasjonspinnene er koordinater som ble plassert for over 30 sekunder siden.

# 5 Validering, verifikasjon og test

## 5.1 Verifikasjonsplan

Systemet må testes for å verifisere at det oppfyller systemkravene definert i seksjon 2.3. For å verifisere kravene er det laget en plan, vist i Tabell 5.1.

Tabell 5.1: Verifikasjonsplan inndelt i delsystemer.

Systemkrav ↓	Hva skal testes	Hvordan skal det testes
	<b>Kapsel</b>	
1	Det testes at kapselen tåler fysisk motstand.	Dette gjøres ved å slippe kapselen fra 0.5 m, 1 m og 2 m.
2	Det testes at kapselen tåler å flyte i vann uten at vann trenger gjennom.	Dette gjøres ved å legge papir i kapselen og la den flyte i vannet i 10 min. Det analyseres da om papiret har blitt vått.
3	Det testes at kapselen flyter stabilt, med senderen over vann.	Dette gjøres ved å la kapselen flyte i vann i 10 min, og analysere at den er stabil hele den tiden.
4	Det testes at kapselen med senderen har en massettethet mindre enn vann.	Dette gjøres ved å ta vekten på kapselen med senderen, og dele det på volumet av kapselen.
5	Det skal testes at produktet sitter komfortabelt på kroppen, og at det ikke begrenser bevegelse.	Dette gjøres ved å få fem utvalgte personer til å prøve redningsvesten og utføre diverse bevegelser, og få tilbakemelding på hvordan de opplevde det.
6	Det skal testes om det er enkelt nok å montere og ta av kapselen fra redningsvesten slik at det er innen tidsrammen.	Dette gjøres ved å måle tiden det tar å montere og ta av kapselen fra redningsvesten.

Fortsettelse på neste side

Tabell 5.1: Verifikasjonsplan inndelt i delsystemer. (Fortsettelse)

<b>Systemkrav</b>	<b>Hva skal testes</b>	<b>Hvordan skal det testes</b>
7	Det skal testes at kapselen er festet til redningsvesten etter utløsing, med en snorlengde på minst 2 meter.	Dette gjøres ved å måle lengden på snoren, og observere at den sitter fast i vesten.
↓	<b>Sender</b>	
8	Det skal testes og sammenlignes hvordan det utløste signalet blir mottatt når kapselen med senderen er i luft og vann.	Dette gjøres ved å skrive kode som måler Signal-to-Noise Ratio (SNR) ved å sammenligne det sendte og mottatte signalet. Vi forventer at verdien skal ligge mellom 35 dB til 50 dB i luft, og dempes med maks 5 % etter at kapselen med senderen har vært i vannet i 10 min.
9, 10	Det testes at alarmsystemet kun gir advarsel etter senderen er utløst og ikke før.	Dette gjøres ved å koble ledningene til senderen fra hverandre, og analysere at alarmsystemet kun reagerer etter.
11	Det skal testes at kartet ikke viser posisjonen til senderen før den har blitt utløst.	Dette gjøres ved å observere at kartet kun viser og oppdaterer posisjonen til senderen når den er utløst.
12	Det skal testes at senderen begynner å lyse etter webapplikasjonen har mottatt posisjon.	Dette gjøres ved å observere om lyset på senderen lyser så snart posisjonen er tilgjengelig i kartet i webapplikasjonen.
↓	<b>Alarmsystem</b>	
13, 14	Det skal testes forsinkelsen fra når senderen er blitt utløst både i vann og manuelt, til alarmsystemet gir lyd.	Dette gjøres ved å ta tiden fra ledningene til senderen kobles fra hverandre, til alarmsystemet kommer med lyd.
↓	<b>Webapplikasjon</b>	

Fortsettelse på neste side

Tabell 5.1: Verifikasjonsplan inndelt i delsystemer. (Fortsettelse)

<b>Systemkrav</b>	<b>Hva skal testes</b>	<b>Hvordan skal det testes</b>
15	Det skal testes at kartet oppdaterer posisjonen til senderen innen hvert 5 s på webapplikasjonen.	Dette gjøres ved å ta tiden fra da posisjonen sist ble oppdatert til posisjonen oppdateres igjen, så lenge systemet er utløst. Dette gjøres 12 ganger etter hverandre.

## 5.2 Testing av verifikasjonsplan

### 5.2.1 Systemkrav 1

For å verifisere kapselens robusthet ble den sluppet fra høyder på 0.5, 1 og 2 meter. Disse høydene ble valgt med tanke på mulige tilfeller der kapselen kan bli skadet, for eksempel at en person faller på bakken.

Kapselen tålte alle høydene, men fikk noen lettere skader på utsiden av lokket etter målingen på 2 meter. Til tross for dette anses testen som bestått da skaden var minimal, som vist i Figur 5.1. Systemkrav 1 er dermed verifisert.



Figur 5.1: Overfladisk skade på lokk etter verifikasiing av systemkrav 1.

### 5.2.2 Systemkrav 2

For å verifisere at kapselen kan flyte i vann uten at vann trenger gjennom ble det lagt tørt papir på innsiden av kapselen, for så å la den flyte i vann i 10 minutter, som vist i Figur 5.2 .



Figur 5.2: Kapselen som flyter i vann.

Etter 10 minutter ble kapselen åpnet og papiret ble analysert. Det hadde trengt inn vann, og papiret var vått. Dermed er ikke systemkrav 2 verifisert, noe som aksepteres da det ikke var et nødvendig krav for prototypen.

### 5.2.3 Systemkrav 3

For å verifisere at kapselen flyter stabilt i vann ble kapselen lagt i vann i en periode på 10 minutter, med analyseering av bevegelse underveis. Det er markert et lite merke for å markere hva som må være over vann. Kapselen fløyt som vist i Figur 5.3 .



Figur 5.3: Kapselen som flyter i vann, med markert merke for å vise hvor mye av kapselen som må være over vann.

Testen viser at kapselen flyter tilstrekkelig stabilt til at senderen befinner seg over vannoverflaten. Systemkrav 3 er derfor verifisert.

#### 5.2.4 Systemkrav 4

For å verifisere at kapselen med senderen har en massetetthet mindre enn vann ble vekten på kapselen med senderen målt, som vist i Figur 5.4, og dette ble videre delt på volumet av kapselen for å beregne massetettheten.



Figur 5.4: Kapselens totale vekt blir målt.

Resultatet ble, som nevnt i seksjon 4.2,  $769.6 \text{ kg/m}^3$ , som er mindre enn vannets massetethet på ca.  $998 \text{ kg/m}$ . Systemkrav 4 er dermed verifisert.

### 5.2.5 Systemkrav 5

For å verifisere produktets komfort ble det valgt ut fem tilfeldige personer til å prøve redningsvesten med kapselen. Disse skulle deretter utføre diverse bevegelser for å avgjøre om produktet kom i veien eller var ukomfortabelt.

På en skala fra 1 til 10, der høyere tall tilsvarer bedre komfort, ble produktet vurdert til verdiene 8, 8, 8, 9 og 10. Altså var produktet meget komfortabelt, og systemkrav 5 er dermed verifisert.

### 5.2.6 Systemkrav 6

For å verifisere om montering og demontering av kapselen fra redningsvesten er innenfor tidsrammen på 5 minutter ble tiden tatt tre ganger for både montering og demontering. Resultatene ble som vist i Tabell 5.2.

Tabell 5.2: Tid for montering og demontering av kapselen.

Testnr.	Montering [m:ss]	Demontering [m:ss]
1	3:08	0:57
2	2:56	1:00
3	3:01	0:52

Som resultatene i Tabell 5.2 over viser er tidsmålingene godt innenfor kravet på 5 minutter, og systemkrav 6 er verifisert.

### 5.2.7 Systemkrav 7

For å verifisere at kapselen er festet til redningsvesten etter utløsing via en snor på minst 2 meter ble først snoren målt, etterfulgt av observering av at enden av snoren sitter fast i vesten.

Snoren ble målt til 2.5 meter, og testen gir at enden av snoren sitter fast i stoffet på vesten. Dermed er systemkrav 7 verifisert.

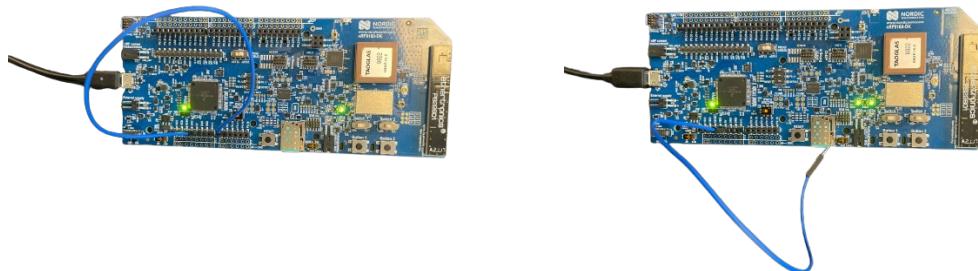
### 5.2.8 Systemkrav 8

For å verifisere at signalet er tilnærmet likt i luft som i vann skal Signal-to-Noise Ratio (SNR) for det sendte og det mottatte signalet måles, og deretter sammenlignes.

Denne testen ble aldri gjennomført grunnet mangel på tid, og fordi den ble prioritert under de andre testene.

### 5.2.9 Systemkrav 9 og 10

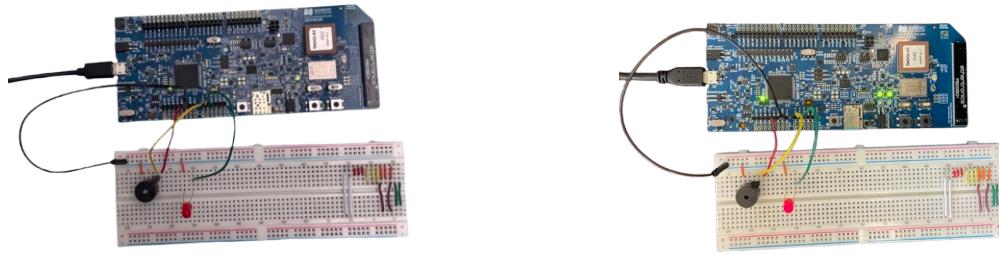
For å verifisere at alarmsystemet kun gir advarsel når senderen er utløst, utløses senderen ved å koble ledningene til senderen fra hverandre, som vist i Figur 5.5.



(a) ledningen koblet mellom PO.15 og GND. (b) ledningen koblet av mellom PO.15 og GND.

Figur 5.5: Systemets sender.

Alarmsystemet reagerte ved å ha et blinkende LED-lys og en høyfrekvent lyd fra buzzeren, kun etter utløsningen. Dette vises i Figur 5.6.



(a) før senderen er utløst, hvor LED-lyset og buzzeren er av.

(b) etter senderen er utløst, hvor LED-lyset blinker og buzzeren lager lyd.

Figur 5.6: Systemets alarm.

Systemkrav 9 og 10 er da verifisert.

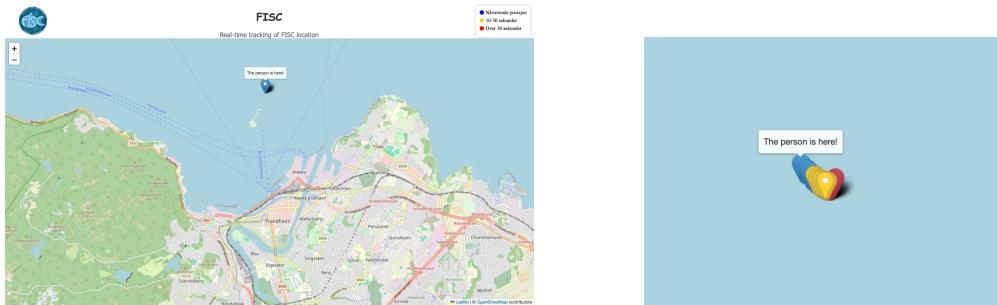
### 5.2.10 Systemkrav 11

For å verifisere at kartet kun viser senderens posisjon etter den har blitt utløst, ble det analysert hvordan kartet ser ut før og etter den blir utløst. Den blir utløst som vist i Figur 5.5. Figur 5.7 viser hvordan webapplikasjonen ser ut før utlösning.



Figur 5.7: Webapplikasjon uten posisjon.

Figur 5.8 viser hvordan webapplikasjonen ser ut etter utlösning, hvor senderen publiserer simulerte posisjonsdata i bevegelse på havet.



(a) Webapplikasjon med posisjon, zoomet ut.  
(b) Webapplikasjon med posisjon, zoomet inn på posisjonen.

Figur 5.8: Webapplikasjon med posisjon.

Figurene illustrerer tydelig at senderen ikke kommuniserer med webapplikasjonen før systemet er utløst. Systemkrav 11 er verifisert.

### 5.2.11 Systemkrav 12

For å verifisere at senderen begynner å lyse etter at webapplikasjonen har mottatt posisjonen, ble systemet utløst som vist i Figur 5.5. Videre ble det observert at det innebygde LED-lyset på senderen ikke begynte å lyse før etter posisjonen ble vist i kartet.

Figur 5.5 demonstrerer dette, hvor (a) viser at kun ett grønt lys på høyre siden av brettet er på, mens (b) viser at to grønne lys er på etter kartet plotter posisjonen. Dermed er systemkrav 12 verifisert.

### 5.2.12 Systemkrav 13 og 14

For å verifisere forsinkelsen fra når senderen er blitt utløst til mottakeren gir lyd, ble det startet en tidsaker samtidig som ledningene til senderen ble koblet fra hverandre, som vist i Figur 5.5. Dette ble gjort fem ganger, og tiden ble stoppet når alarmsystemet reagerte med lyd fra buzzeren.

Tabell 5.3 viser reaksjonstiden til alarmsystemet. Reaksjonstiden har en middelverdi på 3.64 s og standardavvik på 0.66 s. Dette er innenfor kravene til systemkrav 13 og 14, og de er dermed verifisert.

Tabell 5.3: Tiden det tok før mottakeren reagerte

Testnr.	Tid [s]
1	2.64

Fortsettelse på neste side

Tabell 5.3: Tiden det tok før mottakeren reagerte (Fortsettelse)

Testnr.	Tid [s]
2	3.29
3	3.64
4	4.14
5	4.54

### 5.2.13 Systemkrav 15

For å verifisere at kartet oppdaterer posisjonen innen hvert femte sekund ble tiden tatt mellom hver gang posisjonen oppdaterte seg når senderen var utløst og publiserte simulerte posisjonsdata. Tiden ble tatt 12 ganger etter hverandre. Siden webapplikasjonen har en lokal vert, mottar terminalen som kjører backend-koden [31] oppdateringer når webapplikasjonen henter nye posisjonsdata. Figur 5.9 viser at tiden når posisjonen har blitt oppdatert også er presentert.

```
10.22.89.202 -- [24/Apr/2025 13:36:45] "GET /location HTTP/1.1" 200 -
10.22.89.202 -- [24/Apr/2025 13:36:47] "GET /location HTTP/1.1" 200 -
10.22.89.202 -- [24/Apr/2025 13:36:49] "GET /location HTTP/1.1" 200 -
10.22.89.202 -- [24/Apr/2025 13:36:51] "GET /location HTTP/1.1" 200 -
10.22.89.202 -- [24/Apr/2025 13:36:53] "GET /location HTTP/1.1" 200 -
10.22.89.202 -- [24/Apr/2025 13:36:55] "GET /location HTTP/1.1" 200 -
10.22.89.202 -- [24/Apr/2025 13:36:57] "GET /location HTTP/1.1" 200 -
10.22.89.202 -- [24/Apr/2025 13:36:59] "GET /location HTTP/1.1" 200 -
10.22.89.202 -- [24/Apr/2025 13:37:01] "GET /location HTTP/1.1" 200 -
10.22.89.202 -- [24/Apr/2025 13:37:03] "GET /location HTTP/1.1" 200 -
10.22.89.202 -- [24/Apr/2025 13:37:06] "GET /location HTTP/1.1" 200 -
10.22.89.202 -- [24/Apr/2025 13:37:08] "GET /location HTTP/1.1" 200 -
10.22.89.202 -- [24/Apr/2025 13:37:11] "GET /location HTTP/1.1" 200 -
```

Figur 5.9: Terminalen som kjører server.py på en bærbar datamaskin.

Man ser på Figur 5.9 at nettsiden henter ny posisjonsdata fra senderen hvert andre sekund, slik at systemkrav 15 er verifisert.

## 5.3 Testresultater

Tabell 5.4 viser en oppsummering av testresultatene.

Tabell 5.4: Oppsummering av resultatene fra verifikasjonstestene.

Systemkrav	Beskrivelse	Resultat
1	Testen ble gjennomført uten komplikasjoner. Lokket på kapselen fikk en lettere skade etter slipp fra høyden 2 meter.	Bestått
2	Testen ble gjennomført uten komplikasjoner. Kapselen hindret ikke vann fra å trenge gjennom, og papiret ble vått.	Ikke bestått
3	Testen ble gjennomført uten komplikasjoner. Kapselen flyter stabilt, og senderen holder seg over vannoverflaten.	Bestått
4	Testen ble gjennomført uten komplikasjoner. Massetetheten til kapselen med senderen ble $769.6 \text{ kg/m}^3$ , altså mindre enn vannets massetethet.	Bestått
5	Testen ble gjennomført uten komplikasjoner. Produktets komfort ble vurdert mellom 8 og 10 på en skala fra 1 til 10.	Bestått
6	Testen ble gjennomført uten komplikasjoner. Både montering og demontering av systemet på vesten tok under 5 minutter.	Bestått
7	Testen ble gjennomført uten komplikasjoner. Snoren ble målt til 2.5 meter, og satt fast i redningsvesten.	Bestått
8	Testen lot seg ikke gjennomføre på grunn av tidsbegrensning. Ideelt sett skulle signalets kvalitet blitt sammenlignet i luft og i vann.	Ikke bestått
9, 10	Testen ble gjennomført uten komplikasjoner. Alarmsystemet reagerte kun etter senderen hadde blitt utløst.	Bestått
11	Testen ble gjennomført uten komplikasjoner. Senderen delte ikke posisjon i kartet før den hadde blitt utløst.	Bestått
12	Testen ble gjennomført uten komplikasjoner. Lyset på senderen begynte ikke å lyse før etter posisjonen ble vist i kartet.	Bestått
13, 14	Testen ble gjennomført uten komplikasjoner. Alarmsystemet hadde en gjennomsnittlig reaksjonstid på 3.65 sekunder.	Bestått
15	Testen ble gjennomført uten komplikasjoner. Posisjonen ble oppdatert hvert annet sekund.	Bestått

Tabell 5.4 viser at systemet bestod 11 av 13 tester. Test av systemkrav 2 ble ikke bestått

da kapselen ikke var vanntett, og vann trengte gjennom. Test av systemkrav 8 ble som nevnt ikke gjennomført grunnet mangel på tid, og er derfor heller ikke bestått.

## 5.4 Validering

For å avgjøre i hvilken grad brukerkravene, definert i Tabell 1.1, er realisert, ble det utført valideringstesting. Ideelt sett skulle disse testene blitt utført i felt, ved å la den faktiske målgruppen (arbeiderne i havbruksindustrien) teste produktet og dets funksjonalitet i arbeidslivet. Grunnet tidsbegrensning og mangel på ressurser var det dessverre ingen mulighet til dette. Alternativt blir valideringen av brukerkravene vurdert basert på verifikasjonstesting i seksjon 5.2.

Resultatene fra verifikasjonstesting, oppsummert i seksjon 5.3, viser at systemet i stor grad oppfyller systemkravene. Da disse bygger på brukerkravene anses dermed også brukerkravene som oppfylt i stor grad.

# 6 Forbedringer

For å videreutvikle FISC-systemet og forbedre dets funksjonalitet, pålitelighet, brukervennlighet og sikkerhet, er det flere områder som kan forbedres.

## 6.1 Fysisk utforming

En viktig forbedring er å designe egne kretskort for senderen. Tilpasset kretskort vil eliminere unødvendig funksjonalitet, redusere strømforbruket og gjøre senderen mer kompakt. Som en direkte følge av et mer kompakt sender, kan også selve kapselen gjøres mindre og lettere. For å oppfylle systemkrav 2 om vanntetthet, må også kapselen være fullstendig vanntett. Bruk av robuste materialer som er motstandsdyktige mot saltvann og mekanisk stress vil i tillegg sikre holdbarhet i maritime miljøer. Alarmsystemet kan også få en egen robust beholder og et tilpasset kretsskort for mindre arealbruk.

## 6.2 Brukervennlighet

Festemekanismen kan forbedres ved å integrere en spole med uttrekkbar snor, lik en jojo, for å øke brukervennligheten og redusere risikoen for at snoren blir en hindring. Den reduserte størrelsen og vekten som følger av et mindre kapseldesign vil også direkte forbedre brukervennligheten gjennom økt bærekomfort.

## 6.3 Pålitelighet

For å sikre at FISC er pålitelig i en nødssituasjon, bør det gjennomføres en gründigere risikoanalyse for å identifisere og mitigere potensielle feilkilder. Utilstrekkelig LTE-M-dekning kan hindre dataoverføring, og løsninger som redundans via satellittkommunikasjon eller lokale nødsignaler bør vurderes. Påliteligheten til utløsermekanismen er kritisk; den kan potensielt svikte eller aktiveres utilsiktet. Redundante sensorer eller forbedret mekanisk design kan øke sikkerheten her. Batterifeil er en annen åpenbar risiko som kan deaktivere hele systemet. Implementering av bedre batteriovervåking og varsling ved lavt batterinivå kan redusere denne risikoen betraktelig.

## **6.4 Programvare og tilgjengelighet**

For å bedre tilgjengeligheten kan webapplikasjonen kjøres på en dedikert server for å gjøre den tilgjengelig for autoriserte brukere som nødetatene. En mobilapplikasjon med sanntidsvarsler og posisjonsvisning kan utvikles for raskere respons. Integrasjon med eksisterende sikkerhetssystemer på fartøy eller oppdrettsanlegg vil øke effektiviteten ved å koordinere varsler.

## 7 Konklusjon

Dette prosjektet har utviklet og testet prototypen FISC (Floating and Identifying Signal Capsule), et varslings- og posisjoneringssystem designet for å øke sikkerheten for personell i havbruksnæringen ved fall over bord. Målet var å lage et system som automatisk eller manuelt utløses, varsler om bord og sender posisjonsdata til en webapplikasjon for nødetater.

Prototypen, implementert med nRF9160 DK-kort for sender- og alarmfunksjonalitet samt en Flask-basert webapplikasjon. Verifikasjonstestingene viste at systemet bestod av 13 systemkrav, der ett av systemkravene mangler gjennomføring av test.

Designet omfatter en 3D-printet, flytende kapsel i PETG-materiale og en utløsermekanisme integrert med glidelåsen på en redningsvest. Implementeringen av senderen benyttet imidlertid et nRF9160 DK-kort i stedet for den tiltenkte Nordic Thingy:91, noe som medførte at senderenheten ikke passet i kapselen.

Veien videre bør fokusere på blant annet:

1. Gjennomføring av resterende test i verifikasjonsplanen
2. Integrering av senderenheten (enten ved å bruke Nordic Thingy:91 eller redesigne kapselen) for å få en komplett prototype.
3. Forbedring av festemekanismen til redningsvesten.

Prosjektet har lagt et godt grunnlag og demonstrert et fungerende konsept for FISC. Videre arbeid er nødvendig for å fullføre testingen og den fysiske integrasjonen før systemet kan anses som en fullverdig prototype klar for videreutvikling til et kommersielt produkt.

# Bibliografi

- [1] I. M. Holmen og S. M. Holen, «Arbeidsulykker i havbruk - Analyser av registrerte personulykker på havbruksanlegg og -fartøy,» SINTEF Ocean / Fiskeri og ny biomarin industri og Norges teknisk-naturvitenskapelige universitet, 2023 (se s. i, 1, 5).
- [2] C. Knudsen og E. T. Berg. «Regjeringen foreslår nye regler for lakseoppdrett.» (2025), adresse: <https://e24.no/hav-og-sjoemat/i/kwOK7X/regjeringen-foreslaar-nye-regler-for-lakseoppdrett> (se s. 1).
- [3] N. sjømatråd. «2024 ble tidenes beste år for norsk sjømateksport.» (2025), adresse: <https://www.seafood.no/aktuelt/nyheter/2024-ble-tidenes-bestе-ar-for-norsk-sjomateksport/> (se s. 1).
- [4] W. Bank. «Population, total.» (), adresse: <https://data.worldbank.org/indicator/SP.POP.TOTL> (sjekket 04.05.2025) (se s. 1).
- [5] Ø. H. og Sven Martin Jørgensen. «Havbruk – løsningen for framtidens matsystem.» (2025), adresse: <https://www.nordnorskdebatt.no/havbruk-losningen-for-framtidens-matsystem/o/5-124-328964> (se s. 1).
- [6] SINTEF. «156 fiskere har dødd på jobb siden 2000.» (2025), adresse: <https://www.sintef.no/siste-nytt/2024/155-fiskere-har-dodd-pa-jobb-siden-2000/> (se s. 1).
- [7] regjeringen.no. «Nullvisjon til havs: - Alle skal komme hjem fra jobb på havet.» (2023), adresse: <https://www.regjeringen.no/no/aktuelt/nullvisjon-til-havs-alle-skal-komme-hjem-fra-jobb-pa-havet/id2959674/> (se s. 1).
- [8] Sjøfartsdirektoratet. «Nasjonal handlingsplan for sjøsikkerhet.» (2025), adresse: <https://www.sdir.no/contentassets/d5e5695353fc43acaa91b6b7878a930d/nasjonal-handlingsplan-for-sjosikkerhet-bruk.pdf> (se s. 1).
- [9] Fellesforbundet og S. Norge. «Arbeidsmiljø og sikkerhet i havbruk.» (2024), adresse: <https://sjomatnorge.no/wp-content/uploads/2024/10/Veileder-HMS-mars2024-A4-LR.pdf> (se s. 1).
- [10] *Jotron Tron 60GPS EPIRB nødpeilesender med GPS, manuell brakett, seatronic.* adresse: <https://seatronic.no/jotron-tron-60gps-nodpeilesender-med-gps> (sjekket 02.05.2025) (se s. 2).
- [11] Dimeq. «Innovative Maritime Safety Solutions | Dimeq.» (), adresse: <https://www.dimeq.no/> (se s. 2).

- [12] Våt drøm om smart, trådløst nettverk under vann blir virkelighet, Norceresearch, 2022. adresse: <https://www.norceresearch.no/aktuelt/smart-ocean-tradlost-nettverk-under-vann-blir-virkelighet> (sjekket 03.05.2025) (se s. 4).
- [13] R. N. og Torhild M. Martinussen. «155 fiskere har omkommet på jobb siden år 2000: – Tallene er tragisk høye.» (2024), adresse: <https://www.fiskeribladet.no/samfunn/155-fiskere-har-omkommet-pa-jobb-siden-ar-2000-tallene-er-tragisk-hoye/2-1-1593891> (se s. 5).
- [14] L. E. Helseth. «Arkimedesloven.» (2025), adresse: <https://snl.no/arkimedesloven> (se s. 11).
- [15] R. Skatvedt. «Densitet målinger.» (), adresse: <https://nfogm.no/wp-content/uploads/2014/02/RS-Densitet-artikkelen-NFOGM.pdf> (sjekket 03.05.2025) (se s. 11, 23).
- [16] Forssell, Børje (NTNU), Kjerstad, Norvald (NTNU) og Mæhlum, Lars. «GNSS.» (2024), adresse: <https://snl.no/GNSS> (se s. 14).
- [17] Fiskeridirektoratet. «Akvakultur.» (2025), adresse: <https://portal.fiskeridir.no/portal/apps/webappviewer/index.html?id=87d862c458774397a8466b148e3dd147> (se s. 15).
- [18] Telia. «Dekningskart.» (2025), adresse: <https://www.telia.no/nett/dekning/> (se s. 15).
- [19] MQTT.org. «MQTT: The Standard for IoT Messaging.» (2024), adresse: <https://mqtt.org> (se s. 15).
- [20] Gule Sider. «Kart Gule Sider.» (), adresse: <https://kart.gulesider.no/?c=63.460904,10.428085&z=13> (se s. 18).
- [21] 3dp.no. «PETG Filament (Polyethylene terephthalate glycol).» (), adresse: [https://www.3dp.no/blogs/filament-kunnskap-for-fdm-3d-printing/petg-filament-polyethylene-terephthalate-glycol-for-3d-printing?srsltid=AfmB0oqsG-PN2nUp0kGR9GA\\_0\\_19KMJHRyx5YgSJRB1vca7uuf4MGHYt](https://www.3dp.no/blogs/filament-kunnskap-for-fdm-3d-printing/petg-filament-polyethylene-terephthalate-glycol-for-3d-printing?srsltid=AfmB0oqsG-PN2nUp0kGR9GA_0_19KMJHRyx5YgSJRB1vca7uuf4MGHYt) (sjekket 02.05.2025) (se s. 19, 22).
- [22] Nordic Semiconductor. «nRF9160 DK.» (2025), adresse: <https://www.nordicsemi.com/Products/Development-hardware/nRF9160-DK> (se s. 19, 20, 25, 26, 29).
- [23] Pro-Signal. «Buzzer.» (2016), adresse: <https://www.farnell.com/datasheets/2171929.pdf> (se s. 19, 29, 30).
- [24] Onsemi. «Field Effect Transistor - N-Channel, Enhancement Mode.» (2024), adresse: <https://www.farnell.com/datasheets/2171929.pdf> (se s. 19, 30).
- [25] FLASK. «FLASK documentation.» (2010), adresse: <https://flask.palletsprojects.com/en/stable/> (se s. 19, 32).
- [26] Leaflet. «Leaflet documentation.» (), adresse: <https://leafletjs.com/reference.html> (se s. 19, 32).

- [27] Nordic Semiconductor. «Nordic Thingy:91.» (2025), adresse: <https://www.nordicsemi.com/Products/Development-hardware/Nordic-Thingy-91> (se s. 20, 25).
- [28] Ansys. «System on a Chip: How Smaller, Faster Devices are Made.» (2023), adresse: <https://www.ansys.com/blog/what-is-system-on-a-chip> (se s. 20).
- [29] 3DNet. «3dnet PETG 1.75.» (), adresse: <https://3dnet.no/products/petg-1-75-1-0-kg?srsltid=AfmB0oo0sLVPK2ymQrWKpg7jIcs4lJIzCjnpK7V1B9WxiHkDrkBCb5jz> (sjekket 02.05.2025) (se s. 22).
- [30] Mosquitto. «Mosquitto.» (), adresse: <https://test.mosquitto.org/> (se s. 26).
- [31] Merethe Granevang. «Backend.» (2020), adresse: <https://sn1.no/backend> (se s. 44).

## A GitHub for fullstendig prototype

<https://github.com/nhan-daniel/FISC>

## B Kode for nettside

### B.1 server.py

```
from flask import Flask, jsonify, render_template
from flask_cors import CORS
import paho.mqtt.client as mqtt
import json

# Initialiserer Flask-applikasjonen
app = Flask(__name__)

# Aktiverer CORS for å tillate forespørsel fra andre
# domener
CORS(app)

# Initialiserer en global variabel for å lagre den
# siste posisjonen
latest_location = {"lat": 0, "lon": 0}
# Flag for å kun sende én gang
has_sent = False

def on_message(client, userdata, msg):
    global latest_location, has_sent
    try:
        # Dekoder meldingen som en UTF-8 string
        payload_str = msg.payload.decode("utf-8")
        print(f"[DEBUG] Mottatt melding fra topic
'{msg.topic}': {payload_str}")

        # Prøver å laste dataene som JSON
        data = json.loads(payload_str)

        if isinstance(data, dict) and "lat" in data and
           "lon" in data:
```

```

        # Oppdaterer vi posisjonen hvis dataene er
        # et JSON-objekt med lat og lon
        latest_location = data
        print(f"[DEBUG] Oppdatert posisjon:
              {latest_location}")

        if not has_sent:
            # Hvis flagget has_sent er False,
            # sender vi en melding til
            # gruppe10/elsysprosjekt/sendersub
            print("[DEBUG] Sender 'LED1ON' til
                  gruppe10/elsysprosjekt/sendersub")
            client.publish(
                "gruppe10/elsysprosjekt/sendersub",
                payload="LED1ON",
                qos=1,
                retain=False,
            )
            has_sent = True
        else:
            print("[DEBUG] JSON er ikke en gyldig
                  posisjon.")
    except json.JSONDecodeError:
        # Hvis JSON-dekodingen feiler, skriver vi en
        # feilmelding og sender "LED1OFF"
        print(f"[DEBUG] Ikke gyldig JSON:
              {payload_str}")
        print("[DEBUG] Sender 'LED1OFF' til
              gruppe10/elsysprosjekt/sendersub")
        client.publish(
            "gruppe10/elsysprosjekt/sendersub",
            payload="LED1OFF", qos=1, retain=False
        )
        has_sent = False
    except Exception as e:
        # Håndterer generelle unntak og logger feil
        print(f"[ERROR] Feil i on_message: {e}")

# Initialiserer en MQTT-klient og kaller på on_message
# for når meldinger mottas
client = mqtt.Client()
client.on_message = on_message

```

```

# Forsøker å koble til MQTT-brokeren test.mosquitto.org
# på port 1883
client.connect("test.mosquitto.org", 1883, 60)

try:
    # Abонnerer på topicen
    # gruppe10/elsysprosjekt/hovedtopic for å motta
    # meldinger
    client.subscribe("gruppe10/elsysprosjekt/hovedtopic")
    # Starter MQTT-klientens hovedloop for å motta
    # meldinger
    client.loop_start()
except Exception as e:
    # Logger eventuelle feil ved tilkobling til MQTT
    print(f"Feil ved tilkobling til MQTT: {e}")

# Flask-rute som returnerer den siste mottatte
# posisjonen som JSON
@app.route("/location", methods=["GET"])
def get_location():
    return jsonify(latest_location)

# Flask-rute som returnerer en HTML-side når brukeren
# besøker URL-en
@app.route("/")
def home():
    return render_template("index.html")

# Slår på automatisk omlasting av maler under utvikling
app.config["TEMPLATES_AUTO_RELOAD"] = True

if __name__ == "__main__":
    # Starter Flask-applikasjonen og gjør den
    # tilgjengelig på alle IP-adresser på port 5000
    app.run(host="0.0.0.0", port=5000, debug=True)

```

## B.2 index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
      initial-scale=1.0">
    <title>Persons Location</title>
    <link rel="stylesheet"
      href="https://unpkg.com/leaflet/dist/leaflet.css"/>
    <link href="https://fonts.googleapis.com/css2?
family=Roboto:wght@400;700&display=swap"
      rel="stylesheet">
  <style>
    #map { height: calc(100vh - 100px); width: 100%;
      position: absolute;
      top: 100px;
      left: 0;}
    .legend {
      position: absolute;
      top: 10px;
      right: 10px;
      background: white;
      padding: 10px;
      border-radius: 5px;
      font-size: 14px;
      box-shadow: 0 0 10px rgba(0,0,0,0.2);
    }
    .legend div {
      display: flex;
      align-items: center;
      margin-bottom: 5px;
    }
    .legend span {
      width: 12px;
      height: 12px;
      display: inline-block;
      margin-right: 5px;
      border-radius: 50%;
    }
    .logo {
      position: absolute;
      top: 10px;
```

```

        left: 10px;
        height: 90px;
        z-index: 1000;
    }
}

</style>
</head>
<body>
    
    <h1 style="font-family: 'Times New Roman', serif;
        font-size: 29px; font-weight: bold; color: #333;
        text-align: center;">
        Floating and Identifying Signal Capsule
    </h1>
    <h2 style="font-family: 'Times New Roman', serif;
        font-size: 18px; font-weight: normal; color:
        #666; text-align: center;">
        Real-time tracking of FISC location
    </h2>
    <div id="map"></div>
    <div class="legend">
        <div><span style="background: blue;"></span>
            Nåværende posisjon</div>
        <div><span style="background: gold;"></span>
            10-30 sekunder</div>
        <div><span style="background: red;"></span> Over
            30 sekunder</div>
    </div>
    <script
        src="https://unpkg.com/leaflet/dist/leaflet.js">
    </script>
    <script>
        var map = L.map('map').setView([63.4182254,
            10.4020677], 13); // Standardposisjon

        L.tileLayer('https://s.tile.openstreetmap.org
        /{z}/{x}/{y}.png', {
            attribution: '&copy; <a
                href="https://www.openstreetmap.org/copyright">
                OpenStreetMap</a> contributors'
        }).addTo(map);

        var markers = [];
        var latestMarker = null;
    
```

```

function getMarkerIcon(age) {
    if (age < 10) {
        return new L.Icon({ iconUrl:
            'https://raw.githubusercontent.com/pointhi/leaflet-color-
            shadowUrl:
            'https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.7.1/im-
            iconSize: [25, 41], iconAnchor: [12,
            41] });
    } else if (age < 30) {
        return new L.Icon({ iconUrl:
            'https://raw.githubusercontent.com/pointhi/leaflet-color-
            shadowUrl:
            'https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.7.1/im-
            iconSize: [25, 41], iconAnchor: [12,
            41] });
    } else {
        return new L.Icon({ iconUrl:
            'https://raw.githubusercontent.com/pointhi/leaflet-color-
            shadowUrl:
            'https://cdnjs.cloudflare.com/ajax/libs/leaflet/1.7.1/im-
            iconSize: [25, 41], iconAnchor: [12,
            41] });
    }
}

function updateLocation(lat, lon) {
    var now = Date.now();

    // Hvis koordinatene er gyldige (ikke 0,0),
    // opprett en markør
    if (lat !== 0 && lon !== 0) {
        // Flytt den gamle nyeste markøren til
        // markers-listen (uten tekst)
        if (latestMarker) {
            latestMarker.unbindTooltip(); //
            Fjerner teksten på gamle markører
            markers.push({ marker: latestMarker,
                timestamp: now });
        }

        // Opprett ny markør med tekst (svevende
        // over pinnen)
        latestMarker = L.marker([lat, lon], {

```

```

        icon: getMarkerIcon(0) // Blå ikon
    }).addTo(map).bindTooltip("The person is
        here!", {
        permanent: true,
        direction: "top",
        offset: [0, -35] // Flytter teksten
            opp
    });

    // Oppdater kartvisningen
    // map.setView([lat, lon], 13);
}
}

// Oppdater fargene på eldre markører
function updateMarkerColors() {
    var now = Date.now();
    markers.forEach((m) => {
        var age = (now - m.timestamp) / 1000; //
            Alder i sekunder
        m.marker.setIcon(getMarkerIcon(age));
    });
}

// Hent posisjon fra serveren hvert 2. sekund
setInterval(function() {
    fetch('/location')
        .then(response => response.json())
        .then(data => {
            updateLocation(data.lat, data.lon);
        })
        .catch(error => console.error('Error
            fetching location:', error));
}, 2000);

// Oppdater farger på eldre markører hvert 2.
    sekund
setInterval(updateMarkerColors, 2000);
</script>
</body>
</html>

```

# C Kode for alarm

## C.1 main.c

```
/*
 * Copyright (c) 2018 Nordic Semiconductor ASA
 *
 * SPDX-License-Identifier: LicenseRef-Nordic-5-Clause
 */

/*Koden har tatt utgangspunkt fra Lesson 4, oppgave 1 i
 Cellular IoT Fundamentals i DevAcademy*/

#include <stdio.h>
#include <ncs_version.h>
#include <zephyr/kernel.h>
#include <zephyr/net/socket.h>
#include <zephyr/logging/log.h>
#include <dk_buttons_and_leds.h>
#include <modem/nrf_modem_lib.h>
#include <modem/lte_lc.h>
#include <zephyr/drivers/gpio.h>
#include <zephyr/sys/printk.h>
#include <zephyr/devicetree.h>
#include <math.h>
#include <zephyr/net/mqtt.h>
#include "mqtt_connection.h"

//GPIO-nummer for utgangene som skal brukes
#define INPUT_PIN 15
#define INPUT_PIN2 16

//Henter GPIO-enheten (gpio0)
#define INPUT_NODE DT_NODELABEL(gpio0)

//Henter LED-definisjoner fra device tree
```

```

#define LED0_NODE DT_ALIAS(led0)
#define LED2_NODE DT_ALIAS(led2)

//Referanser til GPIO og LEDs
const struct device *gpio_dev =
    DEVICE_DT_GET(INPUT_NODE);
static const struct gpio_dt_spec led0 =
    GPIO_DT_SPEC_GET(LED0_NODE, gpios);
static const struct gpio_dt_spec led2 =
    GPIO_DT_SPEC_GET(LED2_NODE, gpios);

//Tre forskjellige timere for å blinking og lyd
static struct k_timer blink_timer_lys;
static struct k_timer blink_timer_alarm;
static struct k_timer blink_timer_diode;

//Timer-handler for LED3
static void blink_timer_handler_lys(struct k_timer
    *timer)
{
    gpio_pin_toggle_dt(&led2);
}
//Timer-handler for GPIO-pin 15
static void blink_timer_handler_alarm(struct k_timer
    *timer)
{
    gpio_pin_toggle(gpio_dev, INPUT_PIN);
}

//Timer-handler for GPIO-pin 16
static void blink_timer_handler_diode(struct k_timer
    *timer)
{
    gpio_pin_toggle(gpio_dev, INPUT_PIN2);
}

//Funksjon som sjekker om alarm er aktivert
static bool blinking = false;
static void check_alarm(){

    int input_state = gpio_pin_get_dt(&led0);

    printk("Input state: %d\n", input_state);
}

```

```

if (input_state == 1 && !blinking) {
    blinking = true;

        //Starter GPIO-pin 15 og 16 høye
    int ret = gpio_pin_set(gpio_dev,
        INPUT_PIN, 1);
    if (ret < 0) {
        printk("Failed to set pin %d
            high (ret = %d)\n",
            INPUT_PIN, ret);
    }
    int ret2 = gpio_pin_set(gpio_dev,
        INPUT_PIN2, 1);
    if (ret2 < 0) {
        printk("Failed to set pin %d
            high (ret = %d)\n",
            INPUT_PIN2, ret2);
    }

        //Starter alarm, lyd og lys
    k_timer_start(&blink_timer_alarm, K_USEC(500),
        K_USEC(500));
        k_timer_start(&blink_timer_lyd,
            K_MSEC(300), K_MSEC(300));
    k_timer_start(&blink_timer_diode,
        K_MSEC(300), K_MSEC(300));

    printk("Input is HIGH: starting LED blink\n");
} else if (input_state == 0 && blinking) {
    blinking = false;

        //Setter GPIO-pin 15, 16 og LED3 lave
    gpio_pin_set(gpio_dev, INPUT_PIN, 0);
    gpio_pin_set(gpio_dev, INPUT_PIN2, 0);
    gpio_pin_set_dt(&led2, 0);

        //Sjekker for feil
    int ret = gpio_pin_set(gpio_dev,
        INPUT_PIN, 0);
    if (ret > 0) {
        printk("Failed to set pin %d
            low (ret = %d)\n",
            INPUT_PIN, ret);
}

```

```

        }

        int ret2 = gpio_pin_set(gpio_dev,
                               INPUT_PIN2, 0);
        if (ret2 > 0) {
            printk("Failed to set pin %d
                   low (ret = %d)\n",
                   INPUT_PIN2, ret2);
        }

        //Stopper timere
        k_timer_stop(&blink_timer_ly);
        k_timer_stop(&blink_timer_alarm);
        k_timer_stop(&blink_timer_diode);
        printk("Input is LOW: stopping LED blink\n");
    }
}

//MQTT-klient og poll-struktur
static struct mqtt_client client;
static struct pollfd fds;

//Semaphore som venter på at LTE er tilkoblet
static K_SEM_DEFINE(lte_connected, 0, 1);

//Aktiverer logging
LOG_MODULE_REGISTER(Lesson4_Exercise1, LOG_LEVEL_INF);

//Håndterer LTE-hendelser (f.eks. tilkobling)
static void lte_handler(const struct lte_lc_evt *const evt)
{
    switch (evt->type) {
    case LTE_LC_EVT_NW_REG_STATUS:
        if ((evt->nw_reg_status !=
             LTE_LC_NW_REG_REGISTERED_HOME) &&
            (evt->nw_reg_status !=
             LTE_LC_NW_REG_REGISTERED_ROAMING)) {
            break;
        }
        LOG_INF("Network registration status:
                 %s",

```

```

                evt->nw_reg_status ==
                LTE_LC_NW_REG_REGISTERED_HOME
                ?
                "Connected - home
                network" :
                "Connected -
                roaming");
        k_sem_give(&lte_connected);
    break;
case LTE_LC_EVT_RRC_UPDATE:
    LOG_INF("RRC mode: %s", evt->rrc_mode
    == LTE_LC_RRC_MODE_CONNECTED ?
            "Connected" : "Idle");
    break;
default:
    break;
}
}

//Initierer modem og LTE-tilkobling
static int modem_configure(void)
{
    int err;

    LOG_INF("Initializing modem library");
    err = nrf_modem_lib_init();
    if (err) {
        LOG_ERR("Failed to initialize the modem
                library, error: %d", err);
        return err;
    }

    LOG_INF("Connecting to LTE network");
    err = lte_lc_connect_async(lte_handler);
    if (err) {
        LOG_ERR("Error in lte_lc_connect_async,
                error: %d", err);
        return err;
    }

    //Venter på tilkobling
    k_sem_take(&lte_connected, K_FOREVER);
    LOG_INF("Connected to LTE network");
    //Slår på LED2 som indikerer tilkobling
}

```

```

        dk_set_led_on(DK_LED2);

        return 0;
    }

//Knappetrykk: for å testing
static void button_handler(uint32_t button_state,
                           uint32_t has_changed)
{
    //Knapp som starter alarmen
    if (has_changed & DK_BTN1_MSK) {
        if (button_state & DK_BTN1_MSK) {
            int err = data_publish(&client,
                                  MQTT_QOS_1_AT_LEAST_ONCE,
                                  "LED1ON",
                                  sizeof("LED1ON")
                                  - 1);
            if (err) {
                LOG_INF("Failed to send message, %d",
                        err);
                return;
            }
        }
    }

    //Knapp som stopper og restarter alarmen
    if (has_changed & DK_BTN2_MSK) {
        if (button_state & DK_BTN2_MSK) {
            int err = data_publish(&client,
                                  MQTT_QOS_1_AT_LEAST_ONCE,
                                  "LED1OFF",
                                  sizeof("LED1OFF")
                                  - 1);
            if (err) {
                LOG_INF("Failed to send message, %d",
                        err);
                return;
            }
        }
    }
}

```

```

int main(void)
{
    int err;
    uint32_t connect_attempt = 0;

    // Initialiserer LED, modem, knapper
    if (dk_leds_init() != 0) {
        LOG_ERR("Failed to initialize the LED
                 library");
    }
    err = modem_configure();
    if (err) {
        LOG_ERR("Failed to configure the
                 modem");
        return 0;
    }
    if (dk_buttons_init(button_handler) != 0) {
        LOG_ERR("Failed to initialize the
                 buttons library");
    }
    if (!device_is_ready(gpio_dev)) {
        printk("gpio0 device not ready\n");
        return 0;
    }

    //Initialiserer MQTT
    err = client_init(&client);
    if (err) {
        LOG_ERR("Failed to initialize MQTT
                 client: %d", err);
        return 0;
    }

    //Konfigurer GPIO som utganger
    gpio_pin_configure(gpio_dev, INPUT_PIN,
                      GPIO_OUTPUT_INACTIVE);
    gpio_pin_configure(gpio_dev, INPUT_PIN2,
                      GPIO_OUTPUT_INACTIVE);

    //Forsøker å koble til MQTT-server
    do_connect:

```

```

//Forsinkelse ved reconnecting
if (connect_attempt++ > 0) {
    LOG_INF("Reconnecting in %d seconds...", CONFIG_MQTT_RECONNECT_DELAY_S);
    k_sleep(K_SECONDS(CONFIG_MQTT_RECONNECT_DELAY_S));
}

err = mqtt_connect(&client);
if (err) {
    LOG_ERR("Error in mqtt_connect: %d", err);
    goto do_connect;
}

err = fds_init(&client, &fds);
if (err) {
    LOG_ERR("Error in fds_init: %d", err);
    return 0;
}

//Konfigurer LED1
gpio_pin_configure_dt(&led0, GPIO_OUTPUT | GPIO_INPUT);

//Initialiserer timere
k_timer_init(&blink_timer_alarm,
             blink_timer_handler_alarm, NULL);
k_timer_init(&blink_timer_lyst,
             blink_timer_handler_lyst, NULL);
k_timer_init(&blink_timer_diode,
             blink_timer_handler_diode, NULL);

//Hovedløkke
while (1) {
    err = poll(&fds, 1, 5000);
    if (err < 0) {
        LOG_ERR("Error in poll(): %d", errno);
        break;
    }

    //Oppdater MQTT-klient
    err = mqtt_live(&client);
    if ((err != 0) && (err != -EAGAIN)) {

```

```

        LOG_ERR("Error in mqtt_live:
                  %d", err);
        break;
    }

    if ((fds.revents & POLLIN) == POLLIN) {
        err = mqtt_input(&client);
        if (err != 0) {
            LOG_ERR("Error in
                    mqtt_input: %d",
                    err);
            break;
        }
    }

    if ((fds.revents & POLLERR) == POLLERR)
    {
        LOG_ERR("POLLERR");
        break;
    }

    if ((fds.revents & POLLNVAL) ==
        POLLNVAL) {
        LOG_ERR("POLLNVAL");
        break;
    }
    //Sjekk om alarmen skal utløses
    check_alarm();
}

//Koble fra MQTT hvis løkka avsluttes
LOG_INF("Disconnecting MQTT client");
err = mqtt_disconnect(&client);
if (err) {
    LOG_ERR("Could not disconnect MQTT
            client: %d", err);
}

//Prøver å koble til på nytt
goto do_connect;

return 0;
}

```

## C.2 prj.conf

```
#  
# Copyright (c) 2020 Nordic Semiconductor ASA  
#  
# SPDX-License-Identifier: LicenseRef-Nordic-5-Clause  
#  
  
# Logging  
CONFIG_LOG=y  
  
# Button and LED support  
CONFIG_DK_LIBRARY=y  
  
# Newlib  
CONFIG_NEWLIB_LIBC=y  
  
# Networking  
CONFIG_NETWORKING=y  
CONFIG_NET_NATIVE=n  
CONFIG_NET_SOCKETS_OFFLOAD=y  
CONFIG_NET_SOCKETS=y  
CONFIG_POSIX_API=y  
  
# Memory  
CONFIG_MAIN_STACK_SIZE=4096  
CONFIG_HEAP_MEM_POOL_SIZE=2048  
  
# Modem library  
CONFIG_NRF_MODEM_LIB=y  
  
# LTE link control  
CONFIG_LTE_LINK_CONTROL=y  
CONFIG_LTE_NETWORK_MODE_LTE_M_NBIOT=y  
  
# MQTT  
# STEP 2.1 - Enable and configure the MQTT library  
CONFIG_MQTT_LIB=y  
CONFIG_MQTT_CLEAN_SESSION=y
```

```
# Application
# STEP 2.2 - Configure the broker name, TCP port, topic
names, and message
CONFIG_MQTT_PUB_TOPIC="gruppe10/elsysprosjekt/hovedtopic"
CONFIG_MQTT_SUB_TOPIC="gruppe10/elsysprosjekt/hovedtopic"
CONFIG_BUTTON_EVENT_PUBLISH_MSG="LED1ON"
CONFIG_MQTT_BROKER_HOSTNAME="test.mosquitto.org"
CONFIG_MQTT_BROKER_PORT=1883

CONFIG_NEWLIB_LIBC_FLOAT_PRINTF=y
CONFIG_NEWLIB_LIBC_FLOAT_SCANF=y
```

## D Kode for sender

### D.1 main.c

```
/*
 * Copyright (c) 2018 Nordic Semiconductor ASA
 *
 * SPDX-License-Identifier: LicenseRef-Nordic-5-Clause
 */

/*Koden har tatt utgangspunkt fra Lesson 4, oppgave 1 i
 Cellular IoT Fundamentals i DevAcademy*/

#include <stdio.h>
#include <ncs_version.h>
#include <zephyr/kernel.h>
#include <zephyr/net/socket.h>
#include <zephyr/logging/log.h>
#include <dk_buttons_and_leds.h>
#include <modem/nrf_modem_lib.h>
#include <modem/lte_lc.h>
#include <zephyr/drivers/gpio.h>
#include <zephyr/sys/printk.h>
#include <zephyr/devicetree.h>
#include <math.h>
#include <zephyr/net/mqtt.h>
#include "mqtt_connection.h"

//Det Elinda ikke forstår: MQTT (og klient), poll
//struktur, Semaphore, LTE. Føler det kommer til å gi
//mening når jeg vet hva de ordene betyr

//GPIO-nummer for utgangen som skal brukes
#define INPUT_PIN 15

//Henter GPIO-enheten (gpio0)
```

```

#define INPUT_NODE DT_NODELABEL(gpio0)

#define _USE_MATH_DEFINES

//Referanser til GPIO
const struct device *gpio_dev =
DEVICE_DT_GET(DT_NODELABEL(gpio0));

//MQTT-klient og poll-struktur
static struct mqtt_client client;
static struct pollfd fds;

//Semaphore som venter på at LTE er tilkoblet
static K_SEM_DEFINE(lte_connected, 0, 1);

//Aktiverer logging
LOG_MODULE_REGISTER(Lesson4_Exercise1, LOG_LEVEL_INF);

//Håndterer LTE-hendelser (f.eks. tilkobling)
static void lte_handler(const struct lte_lc_evt *const evt)
{
    switch (evt->type) {
    case LTE_LC_EVT_NW_REG_STATUS:
        if ((evt->nw_reg_status !=
            LTE_LC_NW_REG_REGISTERED_HOME) &&
            (evt->nw_reg_status !=
            LTE_LC_NW_REG_REGISTERED_ROAMING)) {
            break;
        }
        LOG_INF("Network registration status:
                 %s",
                 evt->nw_reg_status ==
                     LTE_LC_NW_REG_REGISTERED_HOME
                     ?
                     "Connected - home
                     network" :
                     "Connected -
                     roaming");
        k_sem_give(&lte_connected);
        break;
    case LTE_LC_EVT_RRC_UPDATE:

```

```

        LOG_INF("RRC mode: %s", evt->rrc_mode
                == LTE_LC_RRC_MODE_CONNECTED ?
                        "Connected" : "Idle");
                break;
        default:
                break;
}
}

//Initierer modem og LTE-tilkobling
static int modem_configure(void)
{
    int err;

    LOG_INF("Initializing modem library");
    err = nrf_modem_lib_init();
    if (err) {
        LOG_ERR("Failed to initialize the modem
                library, error: %d", err);
        return err;
    }

    LOG_INF("Connecting to LTE network");
    err = lte_lc_connect_async(lte_handler);
    if (err) {
        LOG_ERR("Error in lte_lc_connect_async,
                error: %d", err);
        return err;
    }

    //Venter på tilkobling
    k_sem_take(&lte_connected, K_FOREVER);
    LOG_INF("Connected to LTE network");
    //Slår på LED2 som indikerer tilkobling
    dk_set_led_on(DK_LED2);

    return 0;
}

//Knappetrykk: for å testing
static void button_handler(uint32_t button_state,
    uint32_t has_changed)
{
    //Knapp som starter alarmen
}

```

```

        switch (has_changed) {
            case DK_BTN1_MSK:
                if (button_state & DK_BTN1_MSK){
                    int err = data_publish(&client,
                        MQTT_QOS_1_AT_LEAST_ONCE,
                        CONFIG_BUTTON_EVENT_PUBLISH_MSG,
                        sizeof(CONFIG_BUTTON_EVENT_PUBLISH_MSG)-1);
                    if (err) {
                        LOG_INF("Failed to send
                            message, %d", err);
                        return;
                    }
                }
                break;
        }

    //Definerer variabler til posisjons simulering
    bool if_checked = false;
    bool led1_checked = false;

#define LOOP_RADIUS 30.0
#define CENTER_LAT 63.4575
#define CENTER_LON 10.3890

    static double t = 0.0;

    // Konverterer meter til grader (latitude og longitude)
    double meters_to_degrees_lat(double meters) {
        return meters / 111320.0;
    }

    double meters_to_degrees_lon(double meters, double lat)
    {
        return meters / (40075000.0 * cos(lat * 3.14 /
        180.0) / 360.0);
    }

    //Simulerer posisjon i en uendelig sløyfe
    void generate_infinity_position(double *lat_out, double
        *lon_out) {
        t += 0.05;

```

```

    double x = sin(t);
    double y = sin(t) * cos(t);

    double dx = x * LOOP_RADIUS;
    double dy = y * LOOP_RADIUS;

    double delta_lat = meters_to_degrees_lat(dy);
    double delta_lon = meters_to_degrees_lon(dx,
        CENTER_LAT);

    *lat_out = CENTER_LAT + delta_lat;
    *lon_out = CENTER_LON + delta_lon;
}

//Sjekker inngangsstatus og publiserer posisjon
static void check_input(void) {
    int input_state = gpio_pin_get(gpio_dev, INPUT_PIN);

    printk("Pin P0.15 state: %d\n", input_state);

    if (input_state == 0 && led1_checked) {
        led1_checked = false;
    }

    // Hvis input er 1 - send posisjoner kontinuerlig
    if (input_state == 1) {
        double lat, lon;
        generate_infinity_position(&lat, &lon);

        if (!led1_checked){
            led1_checked = true;
            int err = data_publish(&client,
                MQTT_QOS_1_AT_LEAST_ONCE,

                CONFIG_BUTTON_EVENT_PUBLISH_MSG,
                sizeof(CONFIG_BUTTON_EVENT_PUBLISH_MSG)-1);
            if (err) {
                LOG_INF("Failed to send
                    message, %d", err);
                return;
            }
        }
    }
}

//Lager en JSON-streng for posisjonen

```

```

    char json_str[100];
    int lat_i = (int)(lat * 1e6);
        int lon_i = (int)(lon * 1e6);

        snprintf(json_str, sizeof(json_str),
                  "{\"lat\": %d.%06d, \"lon\": %d.%06d}",
                  lat_i / 1000000, abs(lat_i % 1000000),
                  lon_i / 1000000, abs(lon_i % 1000000));

    int err = data_publish(&client,
                           MQTT_QOS_1_AT_LEAST_ONCE,
                           json_str,
                           strlen(json_str));

    if (err) {
        LOG_INF("Feil ved sending av posisjon: %d",
                err);
    } else {
        LOG_INF("Sendte ø-posisjon: %s", json_str);
    }
    k_sleep(K_MSEC(2000));
}

int main(void)
{
    int err;
    uint32_t connect_attempt = 0;

    //Initialiserer LED, modem, knapper
    if (dk_leds_init() != 0) {
        LOG_ERR("Failed to initialize the LED
                library");
    }

    err = modem_configure();
    if (err) {
        LOG_ERR("Failed to configure the
                modem");
        return 0;
    }

    if (dk_buttons_init(button_handler) != 0) {

```

```

        LOG_ERR("Failed to initialize the
                buttons library");
    }

    if (!device_is_ready(gpio_dev)) {
        printk("GPIO device not ready\n");
        return 0;
    }

    //Initialiserer MQTT
    err = client_init(&client);
    if (err) {
        LOG_ERR("Failed to initialize MQTT
                client: %d", err);
        return 0;
    }
    //Konfigurer GPIO som utganger
    gpio_pin_configure(gpio_dev, INPUT_PIN,
                       GPIO_INPUT | GPIO_PULL_UP);

    //Forsøker å koble til MQTT-server
    do_connect:
        //Forsinkelser ved reconnecting
        if (connect_attempt++ > 0) {
            LOG_INF("Reconnecting in %d seconds...", CONFIG_MQTT_RECONNECT_DELAY_S);
            k_sleep(K_SECONDS(CONFIG_MQTT_RECONNECT_DELAY_S));
        }
        err = mqtt_connect(&client);
        if (err) {
            LOG_ERR("Error in mqtt_connect: %d",
                   err);
            goto do_connect;
        }

        err = fds_init(&client, &fds);
        if (err) {
            LOG_ERR("Error in fds_init: %d", err);
            return 0;
        }

        //Hovedløkke
        while (1) {
            err = poll(&fds, 1, 5000);

```

```

        if (err < 0) {
            LOG_ERR("Error in poll(): %d",
                    errno);
            break;
    }

    //Oppdater MQTT-klient
    err = mqtt_live(&client);
    if ((err != 0) && (err != -EAGAIN)) {
        LOG_ERR("Error in mqtt_live:
                 %d", err);
        break;
    }

    if ((fds.revents & POLLIN) == POLLIN) {
        err = mqtt_input(&client);
        if (err != 0)
            LOG_ERR("Error in
                     mqtt_input: %d",
                     err);
        break;
    }
}

if ((fds.revents & POLLERR) == POLLERR)
{
    LOG_ERR("POLLERR");
    break;
}

if ((fds.revents & POLLNVAL) ==
POLLNVAL) {
    LOG_ERR("POLLNVAL");
    break;
}

//Sjekker om posisjoner skal sendes
check_input();
}

//Koble fra MQTT hvis løkka avsluttes
LOG_INF("Disconnecting MQTT client");
err = mqtt_disconnect(&client);
if (err) {

```

```

        LOG_ERR("Could not disconnect MQTT
                 client: %d", err);
    }

    //Prøver å koble til på nytt
    goto do_connect;

    return 0;
}

```

## D.2 prj.conf

```

#
# Copyright (c) 2020 Nordic Semiconductor ASA
#
# SPDX-License-Identifier: LicenseRef-Nordic-5-Clause
#

# Logging
CONFIG_LOG=y

# Button and LED support
CONFIG_DK_LIBRARY=y

# Newlib
CONFIG_NEWLIB_LIBC=y

# Networking
CONFIG_NETWORKING=y
CONFIG_NET_NATIVE=n
CONFIG_NET_SOCKETS_OFFLOAD=y
CONFIG_NET_SOCKETS=y
CONFIG_POSIX_API=y

# Memory
CONFIG_MAIN_STACK_SIZE=4096
CONFIG_HEAP_MEM_POOL_SIZE=2048

# Modem library
CONFIG_NRF_MODEM_LIB=y

```

```
# LTE link control
CONFIG_LTE_LINK_CONTROL=y
CONFIG_LTE_NETWORK_MODE_LTE_M_NBIOT=y

# MQTT
CONFIG_MQTT_LIB=y
CONFIG_MQTT_CLEAN_SESSION=y

# Application
CONFIG_MQTT_PUB_TOPIC="gruppe10/elsysprosjekt/hovedtopic"
CONFIG_MQTT_SUB_TOPIC="gruppe10/elsysprosjekt/sendersub"
CONFIG_BUTTON_EVENT_PUBLISH_MSG="LED1ON"
CONFIG_MQTT_BROKER_HOSTNAME="test.mosquitto.org"
CONFIG_MQTT_BROKER_PORT=1883

CONFIG_NEWLIB_LIBC_FLOAT_PRINTF=y
CONFIG_NEWLIB_LIBC_FLOAT_SCANF=y

CONFIG_MQTT_CLEAN_SESSION=y
```

## **E KI-deklarasjon**

## Deklarasjon

**Har det i utarbeidingen/utformingen/tilvirkingen av denne besvarelsen/oppgaven blitt brukt KI-baserte hjelpe midler?**

Nei  Ja

Hvis nei, signer:

---

**Hvis ja, fyll ut resten av deklarasjonsskjemaet.**

Jeg tar fullt ansvar for teksten som denne deklarasjonen gjelder for. Jeg har kryssjekket kildene og dataene som kildene er basert på, og jeg forstår argumentene som presenteres. Jeg har ikke brukt kilder som jeg ikke anerkjenner som troverdig.

I arbeidet har jeg tatt i bruk følgende KI-baserte hjelpe midler:

Open AI sin Chat GPT og Nordic Semiconductor sin AI.

---

**Jeg har brukt KI-baserte hjelpe midler på følgende måte (inntil 200 ord):**

Jeg har brukt KI-baserte hjelpe midler til utvikling av koder og hjelp med LaTeX.

---

**Eksempler:**

«Basert på en første idéutvikling brukte jeg Copilot til å skissere ulike mulige svar. Deretter brukte jeg Google Scholar for å finne kilder som kunne støtte eller avkrefte disse svarene»

«Jeg brukte ChatGPT for å oppsummere og forklare avsnitt. / Etter dette oppsummerte jeg resultatene og formulerte tekstavsnitt før jeg spurte ChatGPT om å forbedre tekstflyten»

---

Jeg er kjent med NTNUs regelverk for bruk av kunstig intelligens. Jeg har redegjort for all bruk av kunstig intelligens enten i) direkte i rapporten eller ii) i dette skjemaet.

**Underskrift/Dato/Sted**

*Uhan Daniel V. Nguyen*  
04.05.25  
Trondheim