Teknisk rapport gruppe 1

Tittel: Overvåkning av fiskebestand og miljødata i sund

Forfattere: Andreas Heyerdahl, Andreas Lindeman, Athena McQueen, Christin Bertelsen og Martin Steinsvoll Rikardsen

Versjon: 1	Dato: 07.05.26

Innhold

Akronymliste

MVP - Minimum Viable Product

TDS - Total Dissolved Solids

I2C - Inter-Integrated Circuit

MPU - MicroProcessor Unit

SQl - Structured Query Language

UI - User Interface

JSON - JavaScript Object Notation

RPI - Raspberry Pi

URL - Uniform Resource Locator

HTTP - Hypertext Transfer Protocol

1 Bakgrunn og problemstilling

1.1 Bakgrunn

Villaksen i Norge er i dag klassifisert som nær truet på den norske rødlista, og en av de raskest voksende truslene mot bestanden er den invaderende pukkellaksen [1]. Gjennom samtaler med forsker Jan Davidsen ved Institutt for naturhistorie (NTNU Vitenskapsmuseet) og professor Egil Eide ved Institutt for elektroniske systemer (NTNU), har vi fått innsikt i de kritiske utfordringene forvaltningen står overfor, og hvordan ny teknologi kan bidra til å bevare de naturlige laksebestandene.

Pukkellaksens negative påvirkning skyldes primært dens aggressive adferd og tidligere gytetidspunkt sammenlignet med villaksen. Ved å okkupere de beste gyteplassene tidlig i sesongen, fortrenger den villaksen fra sine naturlige gyteelver. Denne konkurransen rammer også ørreten, selv om denne arten foreløpig ikke anses som like sårbar for utryddelse. Pukkellaksen opptrer ofte i så enorme mengder at den fysiske trengselen driver ørreten bort fra sine leveområder [2].

Et ytterligere kritisk problem er pukkellaksens faste toårige livssyklus [3]. Siden all pukkellaks dør etter gyting, kan store mengder råtnende fisk forurense vassdragene. Dette medfører risiko for oksygenmangel og spredning av sykdommer, noe som påfører både villaks, ørret og det øvrige økosystemet stor belastning. En visuell sammenligning av disse artene er presentert i Figur 1.

(a) Pukkellaks, med hann øverst og hunn nederst.

(b) Laks (øverst) og ørret (nederst).

Figur 1: Visuell sammenligning mellom laks, ørret og pukkellaks.

Omfanget av problemet ble tydelig i 2025, da det ble iverksatt tiltak for å fjerne pukkellaks i hele 63 norske elver [1]. Selv om det i dag eksisterer systemer for utsortering og overvåkning i elvene, er det en økende etterspørsel etter teknologi som kan innhente informasjon om innsig av fisk før de går opp i vassdragene. Ved å overvåke kyst- og fjordområder kan forvaltningen få tidligere varsling og bedre forutsetninger for å sette inn riktige tiltak.

Det finnes i dag flere teknologiske aktører som jobber med relevante løsninger for overvåkning under vann, både i elv og sjø. Selskaper som **Mohn Technology**, **Troll Systems** og **ScaleAQ** er eksempler på aktører som utvikler kamerasystemer og metoder for automatisk identifikasjon av fisk. Samtidig besitter forskningsmiljøer som **NORCE (LFI)** viktig kompetanse på hvordan slik teknologi kan implementeres for å overvåke fiskebestander i ulike vannmiljøer. Innsikten fra disse aktørene danner grunnlaget for det videre arbeidet med å finne løsninger som kan styrke både datakvalitet og forvaltning av de norske laksefiskene. [4] [5] [6]

1.2 Problemstilling

De eksisterende teknologiene som overvåkningssystemer for laksefisk og sorteringssystemer for å fjerne pukkellaks er imidlertid enten er utviklet for bruk i elver, eller merdene brukt i oppdrettsnæringen. Et område det er mangel på tilgjengelig teknologi er innsiget inn i fjordene før fisken beveger seg opp i elvene. Problemstilling vi står ovenfor er derfor:

Hvordan kan vi utvikle et system for innhenting av informasjon til forskning på, og forvaltning av laksebestanden i sund?

1.3 Brukerkrav

I samtaler med Egil Eide har han lagt frem ønsker til funksjoner et system som svarer til problemstillingen bør ha. Etter disse ønskene er det utarbeidet et sett med brukerkrav, disse er samlet i tabellen under (Tabell 1). Kravene er delt inn i funksjon, grensesnitt, utforming, og fiskevelferd.

Tabell 1: Brukerkrav

Kravtype	Brukerkrav	Nr.
1 Funksjon	Systemet skal automatisk identifisere og telle laks i sanntid.	1.1
	Systemet skal automatisk identifisere og telle pukkellaks i sanntid.	1.2
	Systemet skal automatisk identifisere og telle ørret i sanntid.	1.3
	Systemet skal automatisk identifisere og telle sjørøye i sanntid.	1.4
	Systemet skal innhente temperaturdata for vann ved forskjellige dybder.	1.5
	Systemet skal innhente salinitetsdata for vann ved forskjellige dybder.	1.6
	Systemet skal innhente data om havstrøm ved forskjellige dybder.	1.7
	Systemet skal lagre innhentet data.	1.8
	Historisk data skal være tilgjengelig for brukeren.	1.9
2 Grensesnitt	Innhentet data skal presenteres på en måte som er forenlig med eksisterende arbeidsflyt.	2.1
	Grensesnittet skal oppdateres i sanntid.	2.2
3 Utforming	Systemet skal kreve lite vedlikehold.	3.1
	Kostnaden for implementering skal være lav.	3.2
	Systemets skal være simpelt å implementere.	3.3
4 Fiskevelferd	Løsningen skal ikke negativt påvirke den norske laksefisken.	4.1

I tillegg til disse brukerkravene er det også et overordnet fokus på bærekraft.

2 Konsept

I idemyldringen rundt konsepter samarbeidet vi med erfarne fagpersoner fra lignende prosjekter og studerte eksisterende løsninger, både for å hente inspirasjon fra vellykkede tilnærminger og for å avdekke forbedringspotensial

2.1 Eksisterende løsninger

Det finnes i dag flere etablerte systemer for overvåkning av fisk, men disse er i hovedsak utviklet for bruk i lukkede merder i oppdrettsnæringen eller i spesialbygde feller i elveløp. Ved å se på teknologien til ledende aktører kan vi trekke lærdom om hvilke metoder som er overførbare til overvåkning i åpne sund.

2.1.1 Mohn Technology

Mohn Technology utvikler kamerasystemer som benytter maskinsyn for automatisk telling og analyse av fiskens helsetilstand, blant annet deteksjon av lakselus. Teknologien er basert på avansert bildeanalyse, og viser at det er mulig å oppnå høy nøyaktighet i oppgaver som identifikasjon og klassifisering av fisk under vann. Deres løsninger er ofte rettet mot sanntidsovervåkning i oppdrettsmerder, hvor et relativt kontrollert miljø gir gode forutsetninger for at systemene skal fungere optimalt [4].

2.1.2 Troll Systems og ScaleAQ

Selskaper som Troll Systems og ScaleAQ leverer avanserte undervannskameraer og sensorplattformer som tåler tøffe marine forhold. Disse systemene brukes i dag primært for å overvåke fôring og fiskeadferd. Mens maskinvaren er robust og godt egnet for saltvannseksponering over tid, mangler disse løsningene ofte en integrert kobling mellom høyoppløselige miljødata (som salinitetsprofiler) og den visuelle observasjonen av fisken [7] [5].

2.1.3 Huawei

Huawei Norge har i samarbeid med bedrifter som Troll Systems og Simula Consulting utviklet et system for automatisk sortering av pukkellaks. Systemet er installert i Storelva i Berlevåg. Løsningen bruker kamerabasert bildeanalyse og kunstig intelligens til å gjenkjenne ulike fiskearter. Basert på klassifiseringen styres en autonom port som leder pukkellaksen inn i en egen tank, mens annen fisk slippes videre opp i elven. Det er rapportert høy identifikasjonsnøyaktighet, og systemet har bidratt til å fjerne et stort antall pukkellaks fra elven [8].

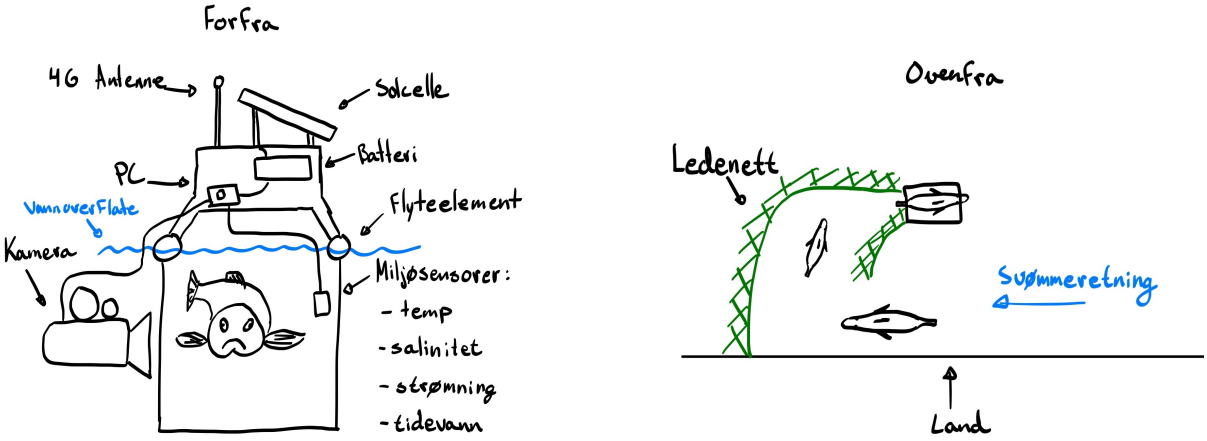
2.1.4 Overføringsverdi til prosjektet

Løsningene fra disse aktørene bekrefter at bildebehandling og kamerasensorikk er moden nok for bruk i felt. Spesielt Huawei-prosjektet viser at et avgrenset måleområde (bredden av en elv), hvor fisken passerer gjennom et definert punkt, muliggjør høy presisjon i artsgjenkjenning. Samtidig ser vi at det eksisterer et teknologisk gap når det gjelder mobile systemer som kan operere selvstendig i sund og fjorder uten fast infrastruktur. Dette danner grunnlaget for konseptet

foreslått av Egil Eide, hvor målet er å kombinere denne typen velprøvd kamerateknologi med detaljerte miljømålinger (f.eks. salinitet, temperatur og dybde) i en flytende enhet.

2.2 Konsepter

For å identifisere en optimal teknisk løsning på problemstillingen, er det utarbeidet tre ulike konseptuelle tilnærminger. Konseptene tar utgangspunkt i et grunndesign foreslått av idéopphaver Egil Eide, men skiller seg fra hverandre gjennom prioritering av brukerkrav for å balansere ressursbruk og funksjonalitet. En visuell fremstilling av den grunnleggende systemoppbygningen er presentert i Figur 2.



(a) Konseptillustrasjon sett forfra.

(b) Konseptillustrasjon sett ovenfra.

Figur 2: Konseptuell skisse av observasjonssystemet, basert på design av Egil Eide.

Kjernen i konseptene er et flytende observasjonssystem som benytter undervannskamera og kunstig intelligens for å identifisere og telle laksefisk (brukerkrav 1.1-1.4). Systemet er tenkt utstyrt med en sensorpakke for måling av miljøparametere som temperatur, salinitet, havstrøm og tidevann. Ved å strukturere disse dataene i vertikale dybdeprofiler og korrelere dem med observert fiskeadferd, kan man kartlegge langsiktige trender i vannmiljøet (brukerkrav 1.5-1.7).

Siden laksefisk ofte søker mot brakkvannsområder, vil detaljerte dybdeprofiler for salinitet og temperatur gi verdifull innsikt i migrasjonsmønstre. Dette gjør det mulig å undersøke om forflytning av fisk påvirkes av faktorer som ferskvannsavrenning fra elver eller endringer i vanntemperatur. Dataene lagres og tilgjengeliggjøres via et grafisk grensesnitt (brukerkrav 1.8–2.2), noe som muliggjør effektiv overvåkning for forskere og forvaltning.

2.2.1 Konsept 1: Fokus på miljødata og dybdeprofiler

Dette konseptet prioriterer en omfattende datainnsamling av det marine miljøet, med særlig vekt på innhenting av vannmiljø data ved forskjellige dybder (brukerkrav 1.5–1.7). Her utstyres observasjonssystemet med en høy tetthet av sensorer plassert i ulike dyp gjennom hele vannsøylen (fra overflaten og nedover). Dette muliggjør generering av dybdeprofiler med høy oppløsning, som gir nøyaktige målinger av hvordan temperatur, salinitet og strømningsforhold endrer seg med dybden.

For å kunne realisere den omfattende miljøovervåkningen innenfor prosjektets rammer, blir kompleksiteten i fiskeidentifiseringen nedprioritert. I dette konseptet utfører KI-modellen kun en

generell telling av fisk, uten å differensiere mellom ulike arter. Brukerkravene om spesifikk artsgjenkjenning (brukerkrav 1.1–1.4) blir dermed kun delvis tilfredsstilt. En sentral funksjon i dette konseptet er imidlertid evnen til å bestemme fiskens nøyaktige svømmedybde ved hjelp av bildebehandling mot en referanselengde. Dette gjør det mulig å plotte fiskens vertikale posisjon direkte inn i de høyoppløselige dybdeprofilene for å se nøyaktig hvilke miljøforhold fisken foretrekker.

2.2.2 Konsept 2: Fokus på artsgjenkjenning

Dette konseptet prioriterer utvikling og optimalisering av KI-modeller for å maksimere nøyaktigheten i identifiseringen av ulike fiskearter. Hovedfokuset ligger på å oppfylle brukerkrav 1.1–1.4, som omhandler sikker identifikasjon av laks, pukkellaks, ørret og sjørøye. Ved å trene algoritmen på spesifikke visuelle kjennetegn, sikrer man høy datakvalitet for overvåkning av artsmangfoldet.

For å kunne realisere en så spesialisert bildeanalyse innenfor systemets rammer, blir innsamlingen av miljødata (brukerkrav 1.5–1.7) nedprioritert. Likevel inkluderes beregning av fiskens svømmedybde via bildeanalyse også i dette konseptet. Selv uten omfattende dybdeprofiler for temperatur og salinitet, er informasjon om hvilke dyp de ulike artene foretrekker i seg selv verdifull for å forstå deres vertikale fordeling i vannsøylen.

2.2.3 Konsept 3: Balansert systemløsning

Det tredje konseptet benytter en helhetlig tilnærming der målet er å skape et samspill mellom fiskens adferd og miljøet. Ved å kombinere moderat artsgjenkjenning med representative dybdeprofiler, kan man se fiskens bevegelser i direkte sammenheng med de faktiske miljøforholdene. Konseptet fokuserer på de viktigste artene (laks, ørret og pukkellaks) for å sikre tilstrekkelig identifikasjonssikkerhet (brukerkrav 1.1–1.3). Oppløsningen på sensordata i vannsøylen vil være noe lavere enn i det spesialiserte konseptet, men fortsatt gi nyttig innsikt. Her er det valgt å prioritere de mest kritiske parameterne (temperatur og salinitet) og velge bort måling av havstrøm (brukerkrav 1.7).

Kjernen i dette konseptet er integrasjonen mellom KI-generert dybdedata fra fisken og sensordata fra vannsøylen (brukerkrav 1.5 og 1.6). Ved å korrelere fiskens posisjon med samtidige målinger av temperatur og salinitet på nøyaktig samme dyp, får man en unik forståelse for hvilke miljøtriggere som påvirker fiskevandringen.

2.3 Valg av konsept

For å finne den beste løsningen på problemstillingen har vi vurdert de tre konseptene opp mot hverandre i en konseptmatrise (Tabell 2). Hvert kriterium er gitt en poengsum fra 1 til 5, der 5 representerer en optimal oppfyllelse av behovet, mens 1 indikerer mangelfull oppfyllelse.

Tabell 2: Konseptmatrise for evaluering av løsningsforslag.

Evalueringskriterier	Konsept 1	Konsept 2	Konsept 3
Fiskidentifisering (Krav 1.1-1.4)	2	5	4
Miljødata/Dybdeprofiler (Krav 1.5-1.7)	5	2	4
Relasjon mellom fisk og dyp	4	3	5
Teknisk gjennomførbarhet	4	4	3
Kostnadseffektivitet	3	3	4
Energiforbruk	2	4	2
Totalsum	**20**	**21**	**22**

Poengfordelingen i matrisen er basert på følgende vurderinger:

- **Fiskeidentifisering:** Her scorer konsept 2 høyest grunnet sitt spesialiserte fokus på artsgjenkjenning. Konsept 3 får en sterk 4-er da det dekker de viktigste artene, mens konsept 1 nedprioriterer dette til fordel for generell telling.
- **Miljødata/dybdeprofiler:** Her scorer konsept 1 høyest da det prioriterer maksimal sensortetthet i vannsøylen. Konsept 3 får en sterk 4-er da det dekker de mest kritiske parameterne, mens konsept 2 nedprioriterer dette til fordel for bildeanalyse.
- **Relasjon mellom fisk og dyp:** Dette er selve kjerneelementet i prosjektet. Konsept 3 får full pott her, da det er designet spesifikt for å koble fiskens posisjon direkte mot de lokale miljømålingene.
- **Teknisk gjennomførbarhet:** De spesialiserte løsningene (Konsept 1 og 2) vurderes som de mest gjennomførbare, da de tillater et isolert teknisk fokus på enten miljødata eller artsgjenkjenning uten å måtte balansere motstridende systemressurser. Konsept 3 scorer noe lavere her, da det er mer omfattende; integrasjonen av både avansert bildebehandling og sanntidsprofilering av vannsøylen øker systemets kompleksitet og stiller strengere krav til synkronisering og databehandling.
- **Kostnadseffektivitet:** Konsept 3 gir mest verdi per investerte krone. Ved å kombinere miljøsensorer med bildebehandling får man ut langt mer informasjon om fiskens adferd enn ved de andre konseptene, samtidig som man unngår de høye kostnadene knyttet til ekstrem datainnsamling eller overdreven sensortetthet.
- **Energiforbruk:** Konsept 1 trekkes for et høyt antall sensorer som krever mye strøm. Konsept 2 er mest energieffektivt med færre komponenter, mens Konsept 3 lander på en gylden middelvei.

Begrunnelse for valg
Basert på totalvurderingen i matrisen er **Konsept 3: Balansert systemløsning** valgt som det beste utgangspunktet for prosjektet. Den største fordelen med dette valget er evnen til å se sammenhenger i vannmiljøet. Ved å kombinere artsgjenkjenning med representative miljødata, kan vi dokumentere nøyaktig hvilke forhold (som temperatur og saltinnhold) som trigger vandringen til for eksempel pukkellaks.

Selv om de andre konseptene scorer høyere på sine respektive spesialområder, gir den balanserte løsningen den mest verdifulle innsikten for både forskning og forvaltning. Konseptet vurderes som noe mindre teknisk gjennomførbart enn de spesialiserte alternativene grunnet systemets økte kompleksitet, men den modulære oppbyggingen gir mulighet til å nedskalere enkelte deler (fiskeidentifisering eller miljødata) dersom det skulle bli nødvendig uten at kjerneideen går tapt. En ytterligere fordel med dette konseptets omfang er at det legger til rette for en tydeligere teknisk arbeidsfordeling internt i gruppen. Dette konseptet legges derfor til grunn for den videre tekniske realiseringen.

2.4 Systemkrav

For at systemet skal tilfredsstille brukerkravene definert i Tabell 1, er det utarbeidet konkrete og målbare systemkrav. Disse kravene beskriver minimumsprestasjonen systemet må oppnå for at identifikasjon, kobling til miljødata og datalagring skal fungere etter hensikten.

Systemkravene i Tabell 3 er spesifisert med hensyn på utvikling av en fungerende prototype innenfor prosjektets tilgjengelige rammer og ressurser. Kravene er delt inn i skal- og bør-krav:

- **Skal-krav:** Definerer kjernefunksjonaliteten i et Minimum Viable Product (MVP). Dette er absolutte krav som må være oppfylt og verifisert for at konseptet skal anses som demonstrert.
- **Bør-krav:** Representerer tilleggsfunksjonalitet eller fysiske optimeringer som er viktige for et ferdig produkt, men som ikke er strengt nødvendige for en vellykket konseptdemonstrasjon.

For å sikre en entydig tolkning av de fysiske målene i tabellen, er systemets dimensjoner definert ut fra fiskens bane gjennom kammeret. Lengden følger fiskens svømmeretning, bredden angir åpningens horisontale utstrekning, og høyden refererer til den vertikale utstrekningen i vannsøylen.

Disse kravene danner grunnlaget for prosjektets videre design og testplan. I prosjektets sluttfase vil hvert krav evalueres for å dokumentere i hvilken grad prototypen har oppfylt de tekniske målene.

Tabell 3: Systemkrav for observasjonssystemet

Kravtype	Nr.	Systemkrav	Brukerkrav
Energihåndtering	1.1	Systemet **bør** kunne operere i minimum 72 timer på én batterilading.	3.1
	1.2	Systemet **bør** kunne lades via solcellepaneler.	3.1, 3.3
Utforming	2.1	Systemet **bør** ha en bredde på 100 ± 20 cm.	3.2, 3.3
	2.2	Systemet **bør** ha en lengde og høyde innenfor intervallet 200 – 400 cm.	1.1–1.6, 3.2, 3.3, 4.1
	2.3	Systemet **bør** ha positiv oppdrift og flyte stabilt i vannoverflaten.	3.3
	2.4	Systemet **bør** lede fisk inn i observasjonskammeret ved bruk av ledenett.	4.1, 1.1
	2.5	All elektronikk **bør** være plassert i en kapsling med tetthetsgrad tilsvarende IP67.	3.1, 4.1
	2.6	Systemet **skal** være modulært	3.2, 3.3
Databehandling	3.1	Systemet **skal** måle temperatur og salinitet med en nøyaktighet på ± 15%.	1.5, 1.6
	3.2	Systemet **skal** ha en oppløsning på minst tre målepunkter i vannsøylen.	1.5, 1.6
	3.3	Systemet **skal** benytte mobildata (4G) for opplasting av data til skyen.	1.8, 2.2
	3.4	Systemet **skal** kunne håndtere intern datakommunikasjon mellom sensorer og sentralenhet.	1.5, 1.6, 1.8
	3.5	Systemet **skal** håndtere frakobling fra 4G-nett.	1.8
Deteksjon	4.1	Systemet **skal** utføre deteksjon av fisk med $< 1\%$ falske negativer.	1.1 – 1.4
	4.2	Systemet **skal** utføre automatisk artsgjenkjenning av laks, ørret og pukkellaks med 90% treffsikkerhet.	1.1 – 1.3
	4.3	Systemet **skal** kunne estimere lengde på fisk med en nøyaktighet på ± 20%.	1.1
Grensesnitt	5.1	Grensesnittet **skal** visualisere antall detekterte fisk av hver art siste time i sanntid.	2.1, 2.2
	5.2	Systemet **skal** visualisere dybdeprofiler for temperatur og salinitet grafisk.	1.5, 1.6, 2.1
	5.3	Brukeren **skal** ha mulighet til å laste ned historiske datasett.	1.9

2.4.1 Utdypning av systemkrav

Tabell 3 konkretiserer hvordan de overordnede brukerkravene skal oppfylles teknisk. For å sikre at systemet er praktisk gjennomførbart som en prototype, er det gjort bevisste valg å bare fokusere på **skal-krav** videre i design og implementering. Begrunnelser for kravstillingen til de ulike kategoriene er videre gitt:

Energihåndtering (Krav 1.1–1.2)

Disse er definert som bør-krav for å gi fleksibilitet i utviklingsfasen. For en prototype er det viktigere å demonstrere kjernefunksjonalitet over en kortere periode enn å ferdigstille et system for fullverdig autonom drift, som ideelt sett krever 72 timer operasjonstid uten ekstern ladning. Solcellelading er inkludert som et bør-krav for å peke mot det ferdige produktets bærekraft og uavhengighet (brukerkrav 3.1).

Utforming (Krav 2.1–2.6)

Kravene til de fysiske målene er satt for å optimalisere forholdene i observasjonskammeret. Høyden er valgt for å kunne fange opp lagdelingen mellom brakkvann, saltvann og temperaturforskjeller (brukerkrav 1.5–1.6), mens bredden gir fisken tilstrekkelig plass til naturlig passasje. Lengden på kammeret er dimensjonert for å gi kamerasystemet tilstrekkelig tid til å analysere og identifisere fisken (brukerkrav 1.1–1.4). Det er stilt krav til IP-grad og oppdrift for å beskytte elektronikken i et marint miljø (brukerkrav 3.1). Bruk av ledenett (systemkrav 2.4) er inkludert for å ivareta fiskevelferd (brukerkrav 4.1) og sikre at fisken ledes gjennom observasjonssystemet. Ved å kreve modulerbarhet (systemkrav 2.6), vil implementeringskostnaden være lav (brukerkrav 3.2), og systemet vil være lett å implementere (brukerkrav 3.3).

Databehandling (Krav 3.1–3.4)

Denne kategorien består av skal-krav da den utgjør kjernen i systemets nytteverdi. Nøyaktighet i måling av temperatur og salinitet, kombinert med en vertikal oppløsning på minst tre målepunkter, er nødvendig for å generere troverdige dybdeprofiler (brukerkrav 1.5 og 1.6). Mobildata (4G) er valgt som kommunikasjonsløsning for å muliggjøre datalagring i skyen og sanntidsoppdatering på minst 1 gang i minuttet (brukerkrav 1.8 og 2.2).

Deteksjon (Krav 4.1–4.3)

For å bevise konseptet er det kritisk at systemet detekterer fisk med svært lav feilmargin, spesielt med tanke på å unngå falske negativer. Kravet om maksimalt 1% falske negativer sikrer at tilnærmet all fisk som passerer gjennom kammeret blir registrert, noe som er avgjørende for pålitelig statistikk. Den automatiske artsgjenkjenningen med 90% treffsikkerhet gjør systemet i stand til å differensiere mellom laks, ørret og pukkellaks, slik at man nøyaktig kan kartlegge artsmangfoldet og identifisere invasive arter (brukerkrav 1.1–1.3). Videre vil estimering av fiskens lengde gi verdifulle data om fiskens størrelsesfordeling og utviklingsstadium uten behov for fysisk håndtering.

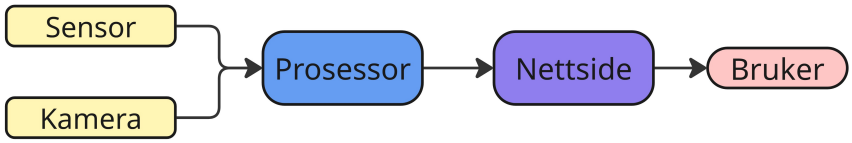
Grensesnitt (Krav 5.1–5.3)

Skal-kravene her sikrer at dataene transformeres fra råverdier til visualiseringer som er direkte sammenlignbare med eksisterende forskningsdata. Dette er avgjørende for at løsningen skal være forenlig med brukernes eksisterende arbeidsflyt og behov for oversikt (brukerkrav 2.1). Med «sanntid» menes her at grensesnittet oppdateres automatisk etter hvert som nye data blir tilgjengelige, uten at brukeren må laste inn siden på nytt.

3 Design

3.1 Systemarkitektur

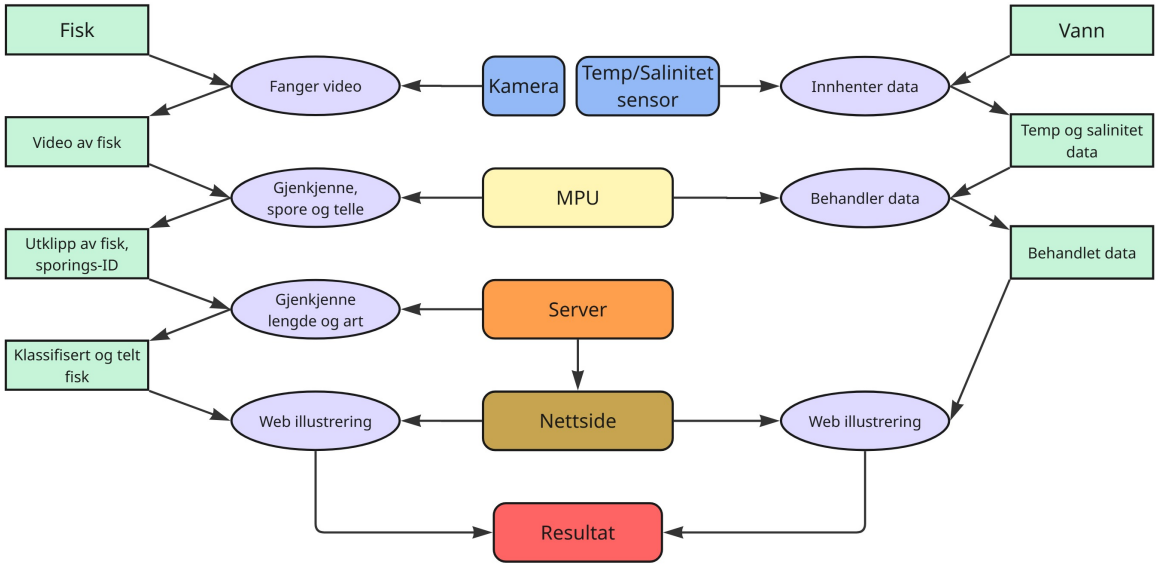
For å realisere systemet er det utviklet en modulær arkitektur som sikrer at kravene til både bildebehandling og miljøovervåkning ivaretas. En overordnet systemarkitektur er vist i Figur 3.



Figur 3: Systemarkitektur: Overordnet design av prototype

Denne inndelingen gjør det mulig å utvikle elektronikk, bildeanalyse og nettside uavhengig av hverandre. Den overordnede kontrollenheten i systemet omtales videre som en Microprosessor Unit (MPU), som i motsetning til en enklere mikrokontroller har regnekraft nok til å håndtere tunge oppgaver som videostrømming og initial bildebehandling. For de mest krevende maskinlæringsmodellene benyttes en ekstern server som fungerer som systemets serverenhet.

Sammenslått funksjonell systemarkitektur og produkttre er gitt i Figur 4.



Figur 4: Funksjonell systemarkitektur/produkttre

Figur 4 viser den logiske dataflyten gjennom systemet, fra rådata til ferdig presentert resultat. Arkitekturen er delt i to parallelle datastier:

- **Fiskegjenkjenning (venstre sti):** Kameraet fanger video som prosesseres av MPU-en for deteksjon, sporing og telling. Videre sendes data til en kraftigere server-PC for artsklassifisering og lengdeestimat før resultatet lastes opp til nettsiden.

- **Miljømålinger (høyre sti):** En sensorhub henter inn verdier for temperatur og salinitet fra vannsøylen. Dataene behandles av MPU-en for å generere dybdeprofiler som synkroniseres med fiskeobservasjonene.

De grønne boksene representerer operander (f.eks. råvideo eller ferdigbehandlet data), mens de lilla boksene beskriver selve prosessene som utføres. Instrumentene i midten (kamera, sensor, MPU og PC) utgjør de fysiske komponentene som muliggjør disse prosessene. Alt dette resulterer i ferdig behandlet informasjon som presenteres for brukeren gjennom et web-basert grensesnitt.

3.1.1 Designvalg og avgrensning

For å realisere det valgte konseptet til en fungerende prototype, er systemet dekomponert i fire tekniske hoveddeler. Denne oppdelingen sikrer at alle **skal-krav** adresseres gjennom designvalg for både maskin- og programvare. De følgende delkapitlene gir en dypere teknisk gjennomgang av:

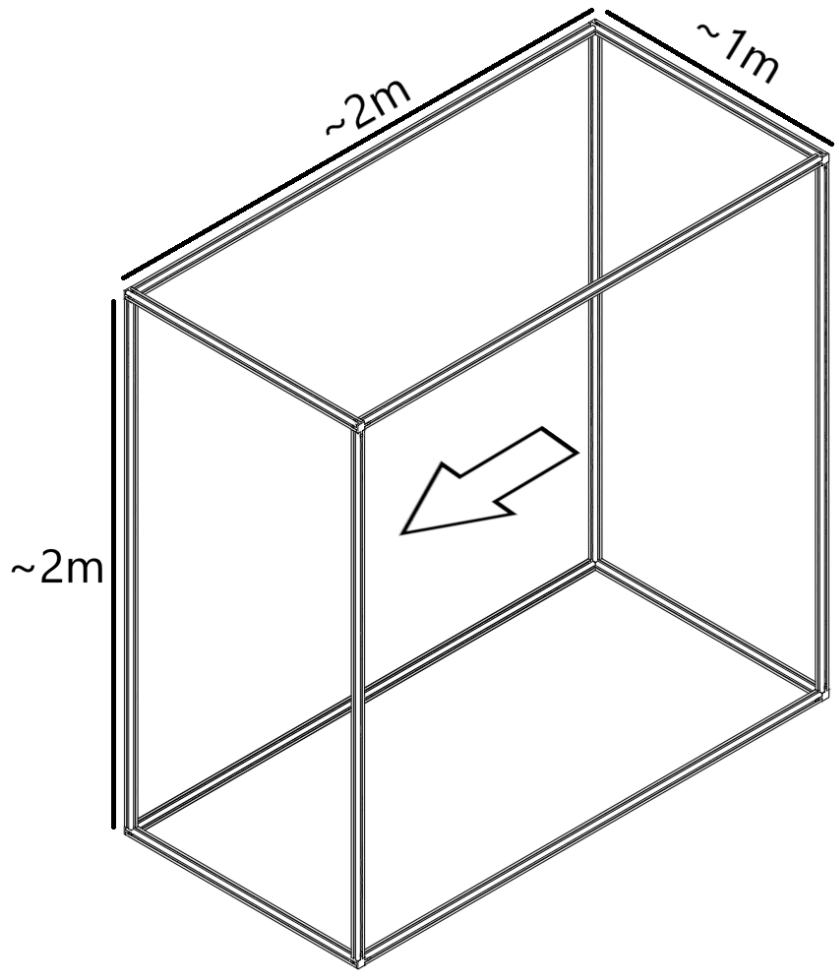
- **Mekanisk konstruksjon:** Den fysiske utformingen av den flytende plattformen, observasjonskammeret og kapslingen som beskytter elektronikken.
- **Datainnsamling:** Maskinvaren for miljøovervåkning og visuell observasjon. Dette omfatter kamerasystemet samt det distribuerte sensornettverket.
- **Databehandling:** Behandlingen av rådataene som samles inn. Dette dekker prosessering av videodata med maskinlæring for å detektere og klassifisere fisk, samt mikrokontrollernes strukturering og overføring av sensordata til server.
- **Brukergrensesnitt:** Utviklingen av den nettbaserte plattformen og databasen. Dette sikrer at forskere og forvaltning får visuell tilgang til sanntidsdata, historikk og dybdeprofiler på en intuitiv måte.

3.2 Mekanisk design

Under utformingen av den mekaniske prototypen har vi lagt til grunn flere sentrale prinsipper. Bærekraft utgjør en viktig kjerne i designet. Samtidig som løsningen skal være miljøvennlig, er det et overordnet mål å holde alle komponenter modulære. Dette sikrer at mindre skader kan utbedres, og at systemet kan oppgraderes uten omfattende endringer eller unødvendig materialsvinn.

3.2.1 Konstruksjon

Hovedmålet med den fysiske konstruksjonen er å lede fisken inn i et avgrenset område, slik at den effektivt kan analyseres av systemets sensorer. For å oppnå dette benyttes et ledernett som strekkes fra kystlinjen og inn mot åpningen av observasjonskammeret. Av hensyn til praktisk lagring og transport (systemkrav 2.6) er sideveggene på observasjonskammeret dimensjonert til rundt 2 meter (systemkrav 2.2). Bredden er satt til 1 meter (systemkrav 2.1) for å gi tilstrekkelig rom for at små stimer med fisk kan passere naturlig, samtidig som avstanden mellom fisken og kamerasystemet holdes optimal for bildeinnhenting. I Figur 5 vises de tenkte dimensjonene for observasjonskammeret.

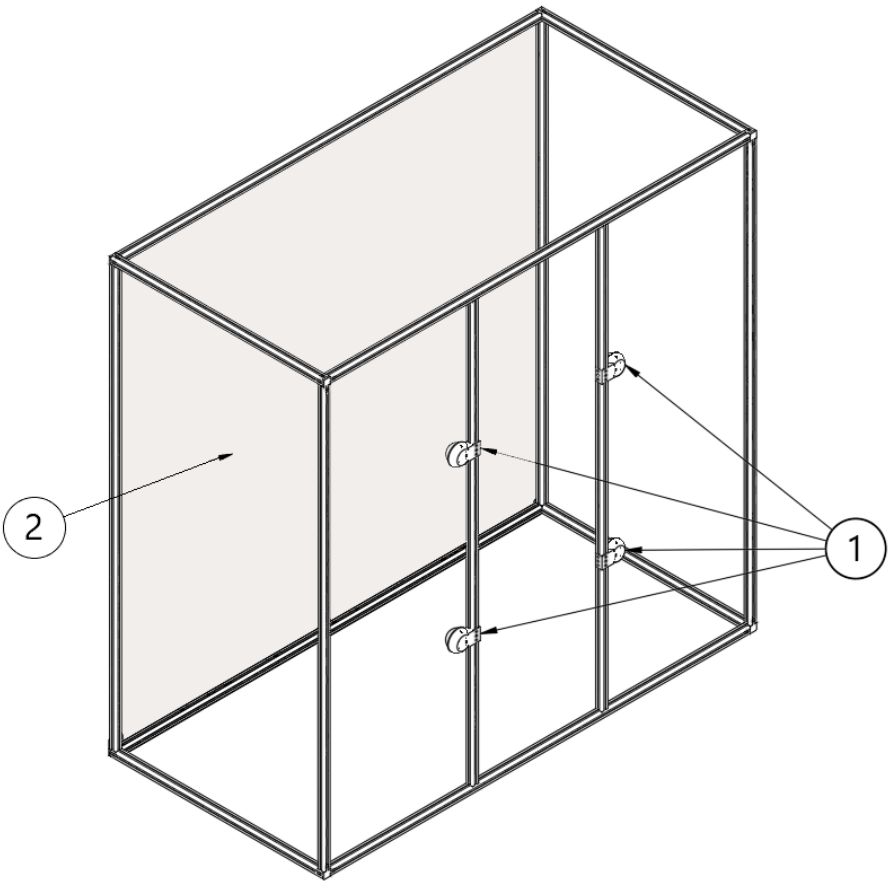


Figur 5: Tenkte dimensjoner for observasjonskammeret. Pilen viser fiskens tenkte svømmeretning.

3.2.2 Bildeinnsamling

For å sikre pålitelig visuell analyse av fisken, kreves det optimal plassering av kameraer og en egnet bakgrunn som gir klare og uforstyrrede bilder. Siden observasjonsområdet utgjør en flate på omtrent 2x2 meter, men kun er 1 meter i bredden, er det nødvendig med flere vidvinkelkameraer for å dekke hele passasjen. Det er derfor valgt et design med fire kameraer plassert i et rutemønster for å maksimere dekningsgraden, slik at systemkrav 4.1–4.3 blir ivaretatt.

På motsatt side av kameraene monteres en bakplate. Denne bør være hvit for å reflektere mest mulig lys og skape gode kontraster, noe som reduserer behovet for kunstig belysning. Overflaten må være glatt slik at marin begroing enkelt kan fjernes ved behov. Materialet må dessuten tåle krevende norske forhold og saltvann over tid, uten å avgi mikroplast eller påvirke miljøet negativt (brukerkrav 4.1). Kameraene vil monteres i vanntette kapslinger, og all dataoverføringskabling dimensjoneres for marint bruk (systemkrav 2.5). Et mulig design for kameramodulene og bakplaten er illustrert i Figur 6.

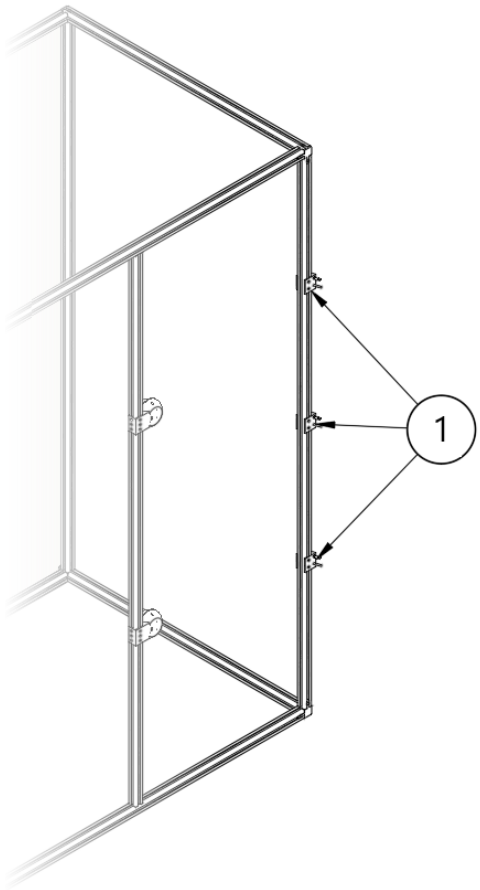


Figur 6: Tenkt design for klare bilder: (1) Kameramoduler, (2) Bakplate/Bakteppe.

3.2.3 Sensorer

Innhenting av miljødata krever en strategisk plassering av sensorsystemet. For å kunne samle data i en vertikal vannsøyle på ulike dybder, plasseres sensorene samlet i ett av observasjonskammerets hjørner (systemkrav 3.2). Sensormodulene monteres på en måte som gjør det enkelt å justere høyde og posisjon senere (systemkrav 2.6). I tråd med det overordnede designprinsippet er oppsettet modulært, noe som tillater rask utskiftning eller fremtidige oppgraderinger. Et

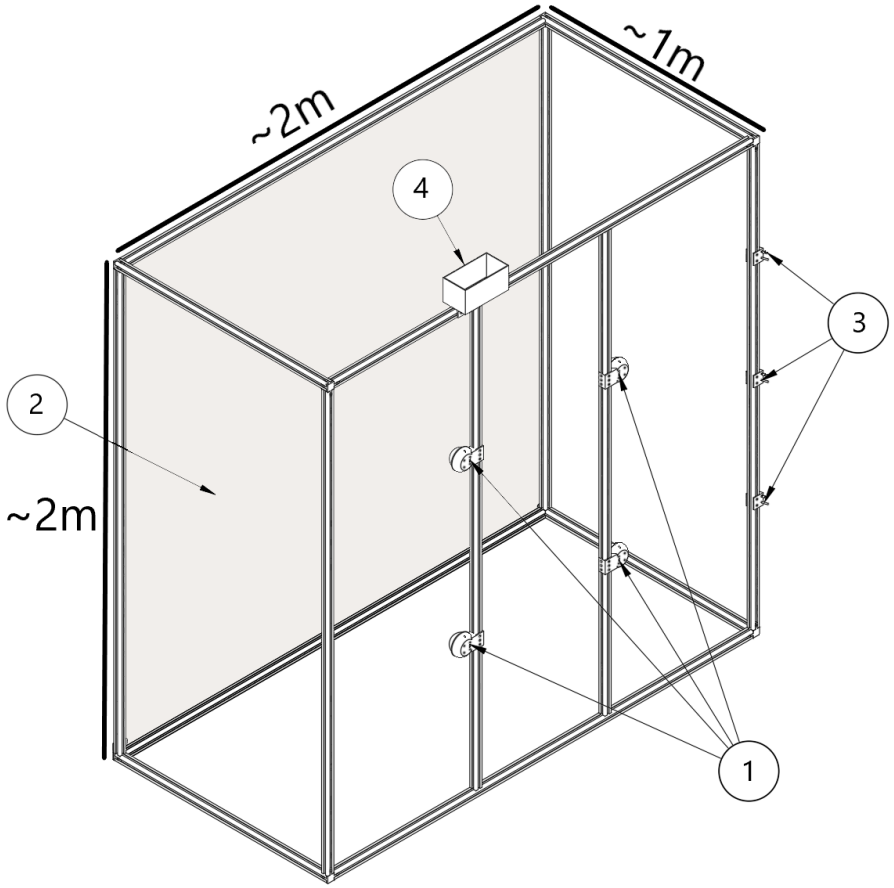
utkast til en stabel med jevnt fordelte sensormoduler er visualisert i Figur 7.



Figur 7: Sensorstabel visualisert: (1) Sensormodul.

3.2.4 Komplett fysisk design

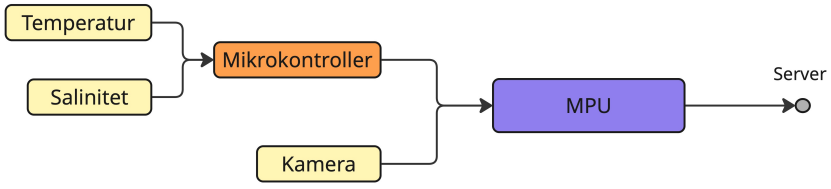
For å binde systemet sammen mekanisk og elektronisk, er det nødvendig med en sikker og tørr plassering av prosesseringsenhetene. Det er designet en vanntett boks som monteres på toppen av rammeverket til observasjonskammeret, hvor den mest kritiske elektronikken oppbevares tørt og trygt. Helheten i konseptet er illustrert i Figur 8.



Figur 8: Visualisert design av konseptet: (1) Kameraer, (2) Bakplate, (3) Sensormoduler, (4) Vanntett elektronikkboks.

3.3 Datainnsamling

Det elektroniske systemet er ansvarlig for informasjonsflyten fra det fysiske miljøet til den digitale databasen. Designet er delt inn i to funksjonelle hoveddeler: sensor-hub og bildebehandling. Den overordnede sammenhengen mellom sensorikk, prosessering og overføring er illustrert i Figur 9.



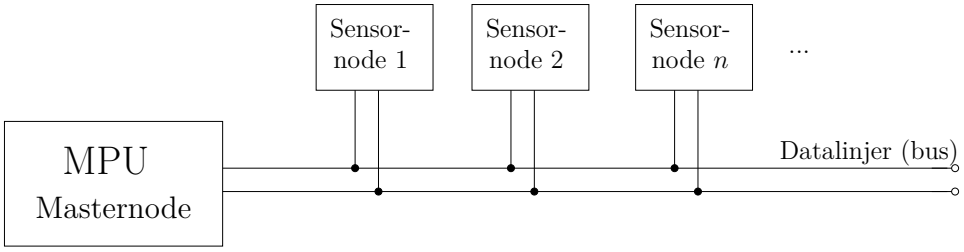
Figur 9: Overordnet flytdiagram for elektronisk systemdesign.

I Figur 9 utgjør sensorene og bildeinnhentingen systemets innganger. Miljødata hentes inn via en mikrokontroller og overføres til en MPU, samtidig som videodata fanges opp av et kamerasystem og sendes til den samme enheten. Ved å skille disse datastrømmene sikres det at systemet kan håndtere krevende bildebehandling og periodiske miljømålinger parallelt, før informasjonen lastes opp til backend trådløst.

3.3.1 Sensor-hub

For å kartlegge miljøforholdene i vannsøylen er systemet designet som en distribuert sensor-hub, der hvert dybdenivå fungerer som en uavhengig modul i et større nettverk. Det er tatt et bevisst designvalg om å benytte en egen prosesseringsenhet per modul. Denne distribuerte arkitekturen gir flere fordeler i tråd med systemets overordnede mål:

- **Modularitet og robusthet:** Ved å benytte en kommunikasjonsform der datalinjene er parallellkoblet, kan moduler enkelt kobles til eller fra uten at det krever fysiske endringer i resten av systemet. Denne topologien sikrer at dersom én modul slutter å fungere, vil det ha minimal påvirkning på de øvrige nodenes evne til å samle inn og sende data.
- **Signalintegritet:** Ved å digitalisere sensorsignalene lokalt ved kilden, reduseres sårbarheten for elektrisk støy over de 1,5 meterne med kabel. Dette sikrer at nøyaktighetsgraden i systemkrav 3.1 overholdes, uavhengig av avstanden til den sentrale prosesseringsenheten.
- **Effektiv ressursbruk:** Designet legger til rette for en buss-arkitektur som muliggjør kommunikasjon mellom mange noder over et minimalt antall ledere. Dette er en effektiv måte å realisere systemkrav 3.4 om strukturert intern datakommunikasjon.



Figur 10: Konseptuell oversikt over systemets buss-topologi.

Selv om en distribuert arkitektur med flere uavhengige enheter øker det totale strømforbruket sammenlignet med en sentralisert løsning, er dette vurdert som en akseptabel avveining for prototypen. Mengden ekstra strøm som kreves for lokalisert prosessering i sensornodene blir neglisjerbar ved optimalisert kode og modulariteten øker robusthet i stor grad.

Synkonisering og polling

For å sikre at dataene fra de ulike dypdenivåene kan sammenstilles til en nøyaktig profil av vannsøylen, benyttes en polling-strategi. Hovedenheten forespør data fra alle nodene i faste, regelmessige intervaller. Ved å kontrollere nøyaktig når data hentes inn, kan systemet generere klare, tidsstemplede øyeblikksbilder av miljøtilstanden. Dette forenkler den senere analysen av rådataene betraktelig, da man kan korrelere målinger fra ulike dyp og sensorer i nøyaktig samme tidsvindu. Hver sensor node legger også ved i sine datapakker høyden de er montert på, noe som spesifiseres under installasjon.

3.3.2 Kameraer

For å sikre pålitelig visuell overvåkning av laksefisken, er kamerasystemet designet for å dekke hele tverrsnittet av observasjonskammeret. Ettersom passasjen er en meter bred og to meter lang

og høy, benyttes et oppsett med flere vidvinkelkameraer plassert i et rutemønster. Dette matrisedesignet maksimerer dekningsgraden og minimerer blindsoner, slik at all fisk som passerer fanges opp av systemet. Kameraene er rettet mot observasjonskammerets hvite bakplate for å sikre optimale kontraster i bildeinnhentingen.

For å kunne håndtere de varierende optiske forholdene i marine miljøer, er kamerasystemet utformet med en kombinasjon av ulike sensoregenskaper. Designet baserer seg på en arbeidsfordeling der ett høyoppløselig kamera har hovedansvaret for å fange detaljerte visuelle markører, noe som kan være interessant for videre manuell analyse. Dette suppleres av flere spesialiserte lavlyskameraer som sikrer kontinuerlig og stabil bildeinnhenting selv under krevende lysforhold og i mørkere farvann.

Et kritisk designkrav for datainnsamlingen er effektiv overføring av den store mengden videodata. For å unngå overbelastning av systemets sentrale prosesseringsenhet (MPU), er det et krav om at kameraene støtter innebygd maskinvarekoding for videokomprimering. Dette minimerer prosessorbelastningen og forhindrer flaskehalser på kommunikasjonsbussen når flere parallelle videostrømmer skal prosesseres.

3.4 Databehandling

En viktig betraktning for databehandling i systemet er robusthet. Systemkrav 3.5 tilsier at innhentet data skal lagres lokalt. Dette vil muliggjøre at systemet kan drifte som normalt selv om det mistes tilkobling.

3.4.1 Sensor-databehandling

Behandlingen av sensordata er designet for å transformere råsignaler fra ulike fysiske sensorer til et enhetlig og pålitelig datasett. En betydelig del av denne prosesseringen foregår desentralisert på selve sensornodene før dataene aggregeres i hovedenheten. Designet sikrer at systemet er skalerbart, enkelt å vedlikeholde og robust mot datatap.

Standardisering og abstraksjon

Siden systemet benytter ulike typer sensorer som opererer med vidt forskjellige kommunikasjonsstandarder, er hver sensornode programmert til å fungere som en oversetter. Rådata fra både analoge og digitale sensorer kan dekodes lokalt på noden og konverteres til et spesifisert, digitalt standardformat. Denne abstraksjonen gjør at hovedenheten kan behandle alle sensormoduler likt, uavhengig av hvilken spesifikk sensorteknologi som er montert på den enkelte modulen.

Modularitet og kalibrering

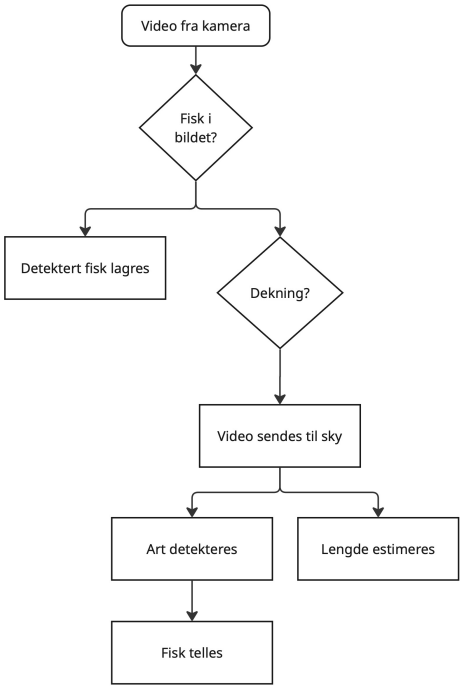
Behandlingslogikken er lagt opp slik at hver sensornode kan kalibreres individuelt under kontrollerte forhold før den integreres i det større systemet. Siden hver node selv håndterer omregningen fra råsignaler til ferdigbehandlede verdier, kan kalibreringsparametere lagres lokalt på noden. Dette gjør systemet svært fleksibelt; moduler kan byttes ut eller flyttes uten at det kreves rekonfigurering av hovedenhetens programvare.

Feilhåndtering og dataintegritet

Når de ferdigbehandlede dataene mottas av hovedenheten, er det dennes ansvar å sørge for sikker videreformidling til server-systemet. For å imøtekomme systemkrav 3.5 om frakobling fra 4g nett, er det implementert logikk for lokal lagring. Dersom opplasting til skyen feiler eller nettverkstilkoblingen er utilgjengelig, lagres dataene lokalt i en kø. Systemet vil automatisk forsøke å resende data når forbindelsen er gjenopprettet. Som en siste sikkerhetsmekanisme kan dataene også hentes ut manuelt fra det lokale minnet ved behov, noe som sikrer at kritiske miljødata aldri går tapt.

3.4.2 Bildebehandling

Som vist i Figur 4 er bildebehandlingen delt i flere steg: deteksjon av fisk lokalt på MPU, og artsgjenkjenning, lengdeestimat og telling på server. Figur 11 illustrerer dataflyten i bildebehandling.



Figur 11: Dataflyt for bildebehandling.

For å oppnå systemkrav 3.5 om lokal lagring er det hensiktsmessig å begrense datamengden som skal lagres lokalt. Dette muliggjør lavere kostnad siden lagringsplass kan utgjøre en stor kostnad. For å samtidig oppnå systemkrav 4.1-4.3 er det valgt å dele inn bildebehandling i to steg.

Lokal bildebehandling

En kildenær modell for bildebehandling kjøres på MPU-en for å filtrere de store mengdene data fra mange kamera videostrømmer til enkelte deteksjoner. Lokal gjenkjenning av objekter i bilder kan gjøres med et konvolusjonelt nevralt nettverk (CNN) trent på relevant datasett. Oppdelingen av deteksjon og resten av prediksjonene (art, lengde) vil muliggjøre bruken av algoritmer av forskjellig størrelser på hver prosesseringsenhet. Dette vil opprettholde presisjon

samtidig som krav for lagring blir lavere. En mindre algoritme kan kjøre på lav effekt lokalt på MPU-en og gjøre deteksjon av fisk i bildet kontinuerlig som video strømmes fra kameraene.

Avbrudd i mobilnettet vil ved denne inndelingen ikke påvirke detektert fisk. Ved å bare lagre de bildene det faktisk er en fisk, vil det i et slikt tilfelle spare lagringsplass. Datatrafikk blir også spart siden bare relevante bilder blir sendt til server. Systemkrav 5.1 om oppdateringer i sanntid kan på den måten overholdes også i områder der mobilnettet gir mindre gunstig opplastingshastighet.

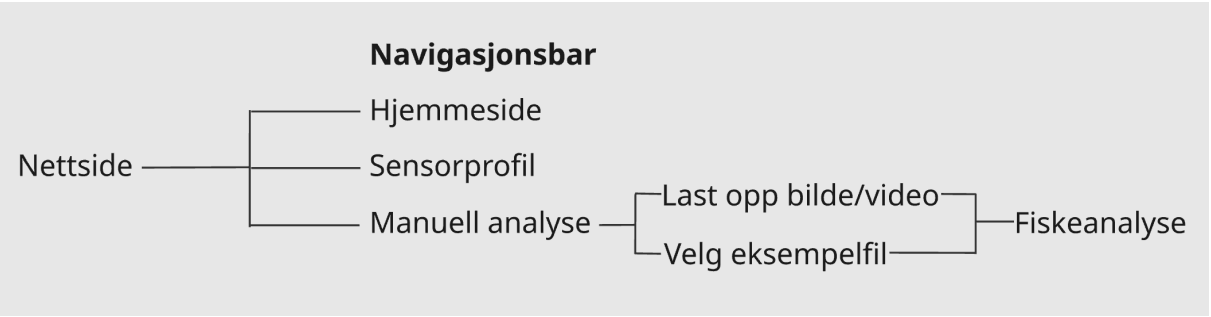
Bildebehandling på server

Større algoritmer kan kjøres på server og oppnå bedre presisjon for de komplekse prediksjonene (artsgjenkjenning og lengdeestimat). Større nevrale nettverk, og av annen type enn CNN er mulig, slik som en Vision Transformer (ViT) [9]. Denne ekstra regnekraften muliggjør bruken av en Multi-Task-arkitektur som kan predikere både art og lengde simultant. Serveren fungerer også som et viktig andrestegs-filter, feildeteksjoner sendt tidlgere kan vidre undersøkes med tyngre modeller og frasortere falske positiver (som bølger eller skygger) som den mindre presise lokale modellen har sluppet gjennom. Til slutt kan serveren samle opp og vurdere flere bildeutklipp fra deteksjoner av nøyaktig samme fisk under passeringen, noe som sikrer et svært robust og nøyaktig sluttresultat gjennom flertallsbeslutninger og gjennomsnittsberegninger.

3.5 Brukergrensesnitt

3.5.1 Webapplikasjon og navigasjonsstruktur

For å gi brukeren tilgang til innsamlede data og systemstatus, er det designet et webbasert brukergrensesnitt. Valget falt på en nettside ettersom det gir en plattformuavhengig tilgang som fungerer like godt på mobile enheter som på PC ute i felt. For å sikre god brukervennlighet er nettsiden strukturert rundt en enkel og intuitiv navigasjonsflyt, som vist i Figur 12.

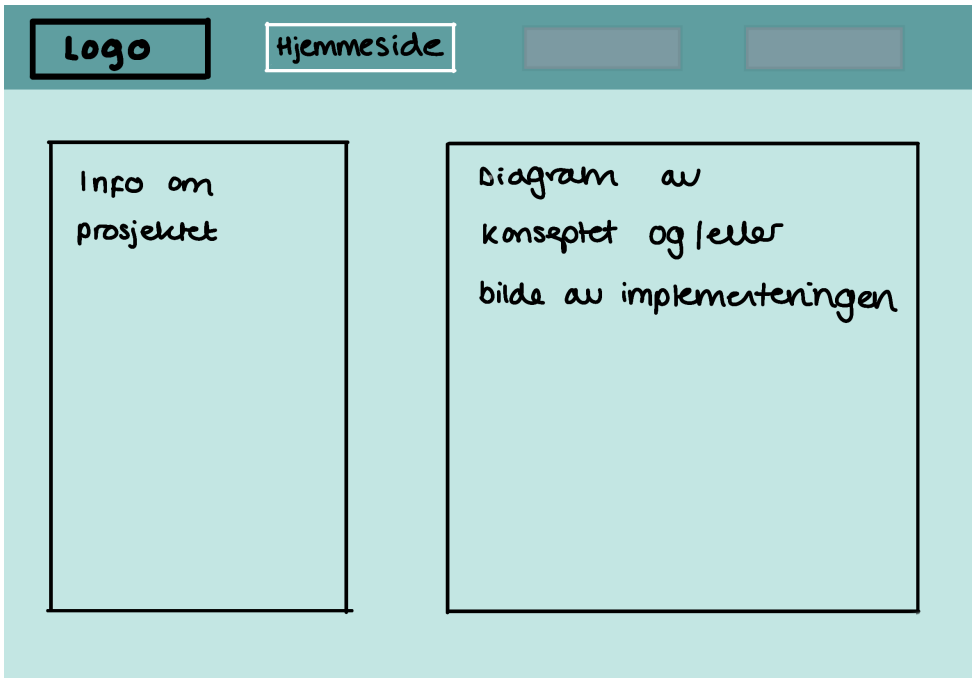


Figur 12: Frontend flytskjema

Basert på navigasjonsstrukturen er det utviklet egne brukergrensesnitt for hver hovedside. Designene viser hvordan informasjon organiseres visuelt, og hvordan arbeidsflyten støttes gjennom grensesnittet.

3.5.2 Hjemmeside

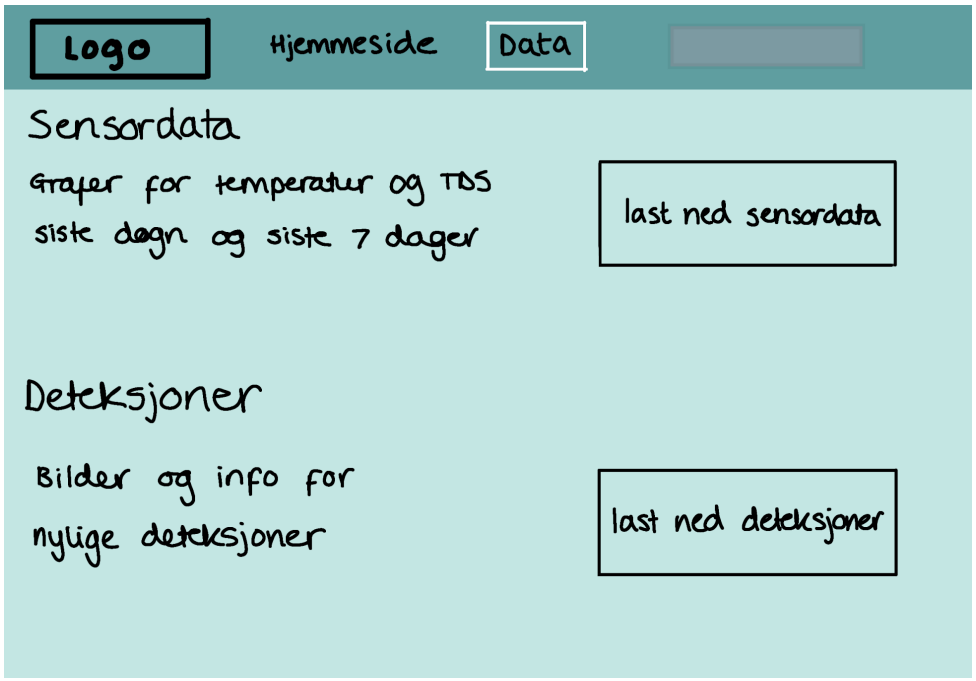
Figur 13 viser tenkt design for nettsidens hjemmeside. Den fungerer som et utgangspunkt for videre navigasjon, og gir brukeren tilgang til informasjon om prosjektet og prototypen. Navigasjonsbaren øverst legger opp til rask tilgang til de ulike funksjonene i webapplikasjonen. Innholdet på siden er delt inn i seksjoner for prosjektinformasjon og visuell presentasjonen av systemet, for å gi brukeren oversikt over konseptets formål og oppbygning.



Figur 13: Hjemmesidedesign

3.5.3 Presentasjon av miljødata og fiskeobservasjoner

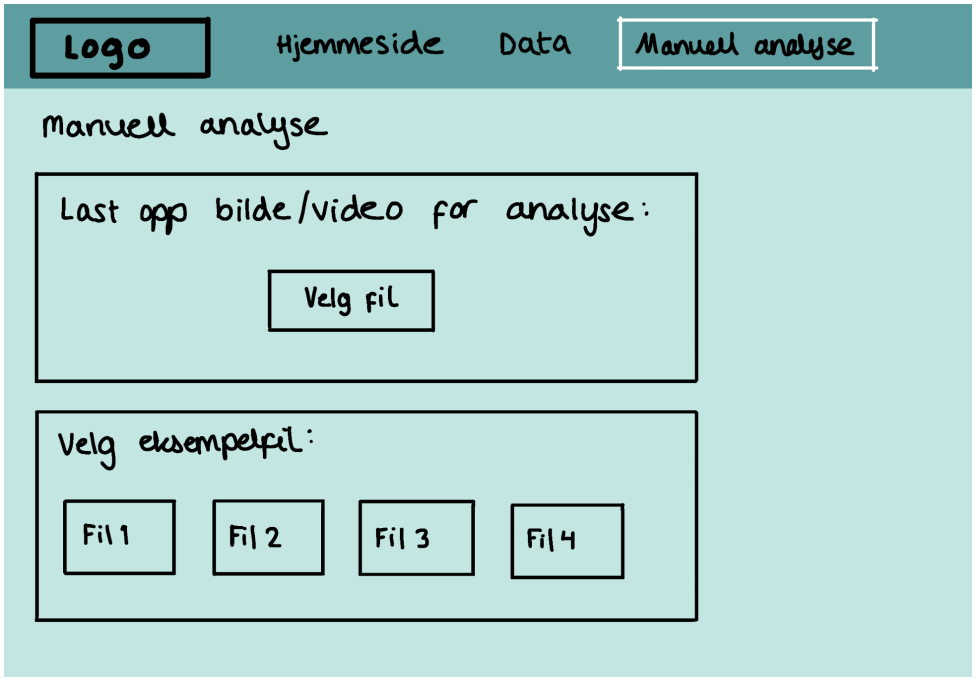
Figur 14 viser tenkt design av datasiden. Denne gir brukeren en samlet oversikt over både sensormålinger og deteksjonsdata fra kamerasystemet. Temperatur- og salinitetmålingene presenteres i separate grafer for siste døgn og siste uke (brukerkrav 2.1 og 2.2, systemkrav 5.2). Målingene er strukturert etter tidsintervall og visualiseres ved ulike dybder (brukerkrav 1.5 og 1.6), noe som gjør det mulig å sammenligne miljøforhold over tid og på forskjellige nivåer i vannet. Deteksjonsdataene viser registrerte hendelser og tilhørende klassifiseringer fra kamerasystemet. Dette gjør det mulig å analysere sammenhenger mellom miljøtilstanden i vannet og registrerte hendelser (systemkrav 5.1). Historisk data kan også lastes ned for videre analyse (brukerkrav 1.9, systemkrav 5.3).



Figur 14: Datasidedesign

3.5.4 Manuell analyse

Figur 15 viser tenkt design av manuell analyse. Denne gir brukeren mulighet til å laste opp egne bilder eller videoer for analyse, i tillegg til å teste systemet med eksempelfiler. Funksjonen gjør det mulig å teste analysemodellene uten at prototypen er ute i felt. Siden er strukturert slik at opplasting av egne filer og valg av eksempelfiler er tydelig separert. Når brukeren har valgt en fil, skal systemet presentere resultatene fra bildeanalysemodellene på en oversiktlig måte.



Figur 15: Manuell analyse design

4 Implementering

I denne seksjonen vil det beskrives hvordan systemet er implementert og realisert for de ulike komponentene. Det vil si hvordan prototype er satt sammen fysisk, hvordan kode er skrevet, og hvordan delsystemene er knyttet til hverandre.

4.1 Maskinvare

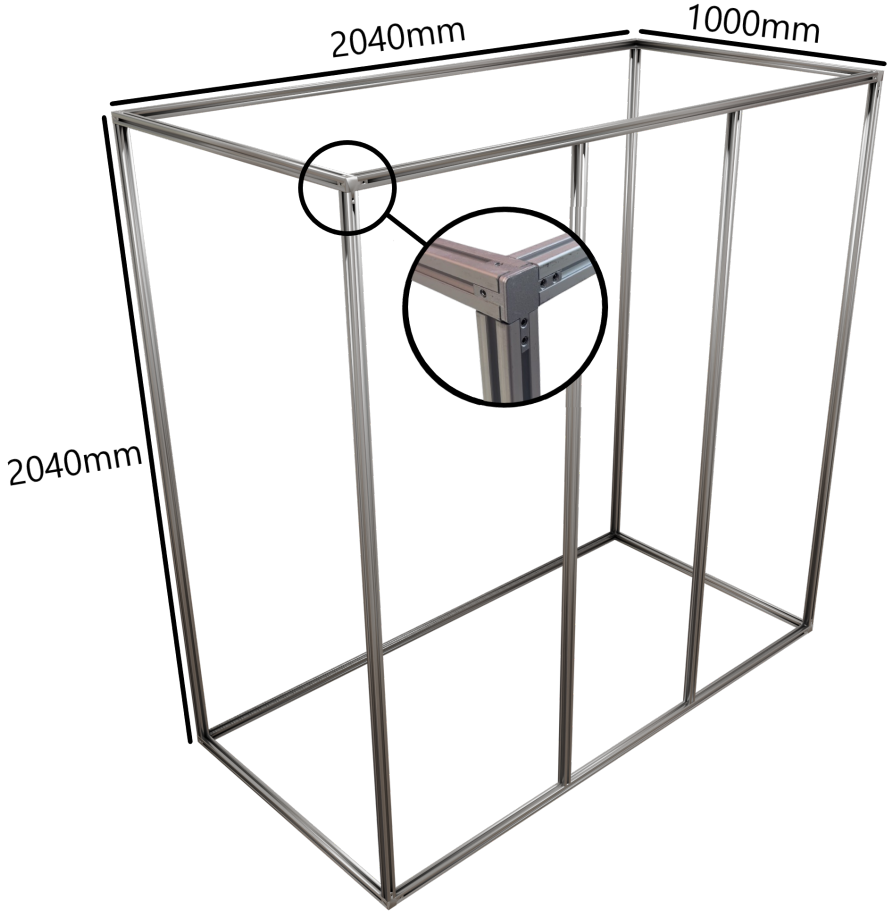
Maskinvareimplementasjonen utgjør det fysiske fundamentet for prototypen. Systemet er realisert med et sterkt fokus på robusthet for marine miljøer, kombinert med en modulær arkitektur som fasiliterer for enkelt vedlikehold, testing og fremtidig skalering.

Før de individuelle delsystemene detaljeres, gir Figur 16 en helhetlig oversikt over den ferdigstilte prototypen. Bildet illustrerer hvordan det mekaniske rammeverket, prosesseringskapslingen og sensornodene er fysisk integrert til en operativ enhet.

Figur 16: Komplett maskinvareimplementasjon av prototypen.

4.1.1 Fysisk utforming og mekanikk

Observasjonskammerets ytre struktur er konstruert av 3030-aluminiumsprofiler. Dette materialet er strategisk valgt grunnet dets høye stivhet-til-vekt-forhold og utmerkede gjenbrukbarhet. Rammeverket består av langsgående profiler på 1980mm og tverrgående profiler på 1000mm, noe som gir observasjonskammeret ytre dimensjoner på 2040 x 2040 x 1000mm. Denne utformingen sikrer et tilstrekkelig volumarial for bildeinnhenting, samtidig som fisken får en naturlig og fri passasje.



Figur 17: Metall grunn konstruksjon for prototypen, hjørnefestet satt i fokus

Som vist i Figur 17 utgjør 1000mm profilene hele bredden på observasjonkammeret, mens lengde og høyde profilene er montert ortogonalt på sideflatene til breddeprofilene. Dette er gjort for å øke stabilitet, samt holde sideveggene kvadratiske. Tykkelsen av de tverrgående profilene utgjør 30mm på hver side av de 1980mm profilene, denne faktoren summerer opp og skaper total dimensjonene våre på 2040mm.

For å optimalisere forholdene for bildeanalysen, er det montert en uniform, hvit bakvegg bestående av en stram presenning på motsatt side av kameraene. Den hvite flaten maksimerer lysrefleksjonen, sikrer høy kontrast og fjerner visuell støy fra det omliggende vannmiljøet, et kritisk premiss for stabil ytelse i systemets maskinlæringsmodell.

All sentral prosesserings- og kommunikasjonselektronikk er plassert i en spesialtilpasset kapsling montert på toppen av rammeverket. Kapslingen er konstruert i treverk for å minimere andelen plastkomponenter, samtidig som den gir nødvendig beskyttelse mot fuktighet og slitasje.

Figur 18: Boks for lagring av essensiele komponenter

4.1.2 Sentral prosesseringsenhet

Systemets sentrale kontrollnode er realisert med en **Raspberry Pi 4B**. Denne enheten fungerer som hjertet i dataflyten og er valgt på bakgrunn av sin kapasitet til å håndtere fire parallelle videostrømmer via USB 3.0. I tillegg til krevende videohåndtering, fungerer Raspberry Pi-en som master i bus-nettverket for innhenting av sensordata, og administrerer all opplasting av ferdigbehandlede pakkedata til skyen via et tilkoblet 4G-modem slik at systemkrav 3.3 tilfredsstilles.

4.1.3 Kamerasystem

For å sikre tilstrekkelig dekning av observasjonskammerets tverrsnitt, består kamerasystemet av en matrise med uavhengige USB-kameraer. Hvert kamera er montert i en egen kapsling bestående av to hovedkomponenter: en festemekanisme for montering på aluminiumsprofiler, og en transparent halvkule i akryl. Akrylkuppelen sørger for at kameramodulene er vanntette uten å begrense kameraenes synsfelt.

For å minimere optisk forvrengning og støy forårsaket av kapslingen, er akrylkuppelen posisjonert slik at dens geometriske senterpunkt sammenfaller med kameraobjektivets optiske senter. Denne konfigurasjonen sikrer at lysstrålene treffer akryllaget tilnærmet vinkelrett, noe som minimerer brytningsfeil i overgangen mellom luft, akryl og vann.

Da den optimale plasseringen av kameraene fremdeles er under uttesting, er kameramodulene utstyrt med en fleksibel monteringsmekanisme. Dette muliggjør trinnløs justering og installasjon på ønsket posisjon langs aluminiumsprofilene.

(a) Forside kameramodul **(b)** Bakside kameramodul

Figur 19: Oversikt over monterte kameramodulers utforming fra henholdsvis forside og bakside.

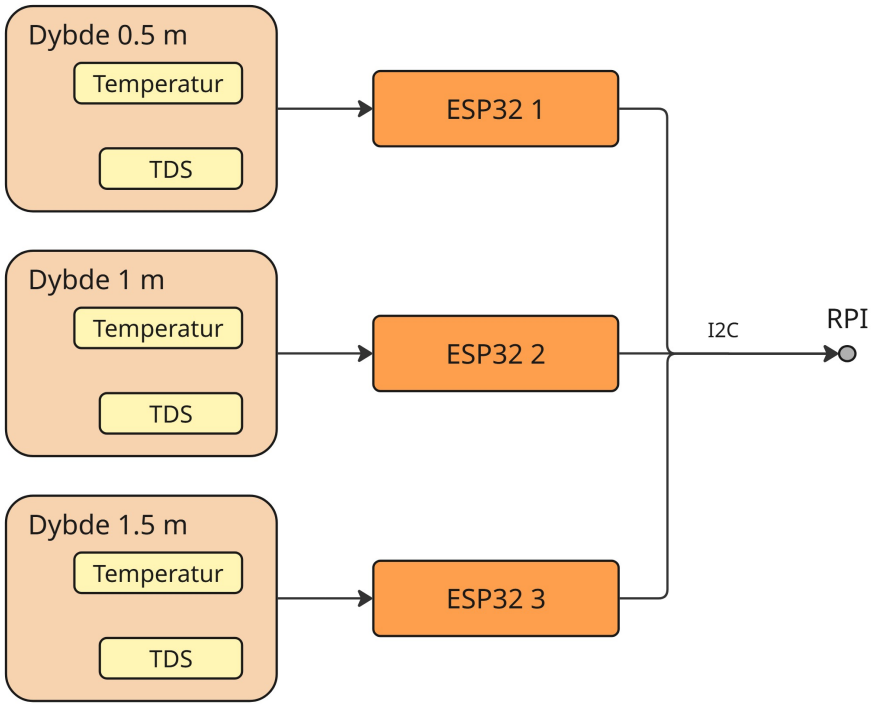
Et kritisk designvalg for denne implementeringen er at samtlige kameraer har innebygd maskinvarekoding for MJPEG. Dette forhindrer flaskehalser på USB-bussen og minimerer CPU-belastningen på Raspberry Pi-en betraktelig.

Kameraoppsettet består av:

- **1 stk. 8MP IMX179:** Et høyoppløselig kamera implementert primært for å fange detaljert visuell informasjon. Denne oppløsningen er nødvendig for å kunne skille oppdrettslaks fra villaks basert på visuelle markører, samt for nøyaktig deteksjon av lakselus.
- **3 stk. 2MP IMX290:** Spesialiserte lavlyskameraer (0.001 Lux) som komplementerer hovedkameraet ved å sikre kontinuerlig og pålitelig bildeinnhenting under krevende optiske forhold og i mørkere farvann.

4.1.4 Sensornettverk

Miljøovervåkningen i vannsøylen er bygget opp som en distribuert sensor-hub. Denne består av uavhengige sensornoder basert på ESP32-mikrokontrollere plassert i strategiske dyp. Disse er knyttet sammen til en Raspberry Pi via et I2C-bussnettverk (Inter-Integrated Circuit), som illustrert i Figur 20.



Figur 20: Systemarkitektur for sensor-hub med distribuert dybdeinnhenting.

Som vist i Figur 20, benyttes det tre uavhengige noder for å innhente data fra dybdene 0,5 m, 1,0 m og 1,5 m. Hver node er konfigurert som en I2C-slave med en fast, unik adresse. I2C-protokollen er valgt fordi implementeringen kun krever standard Serial Data (SDA) og Serial Clock (SCL) pinner på både ESP32 og Raspberry Pi, noe som forenkler kablingen i systemet.

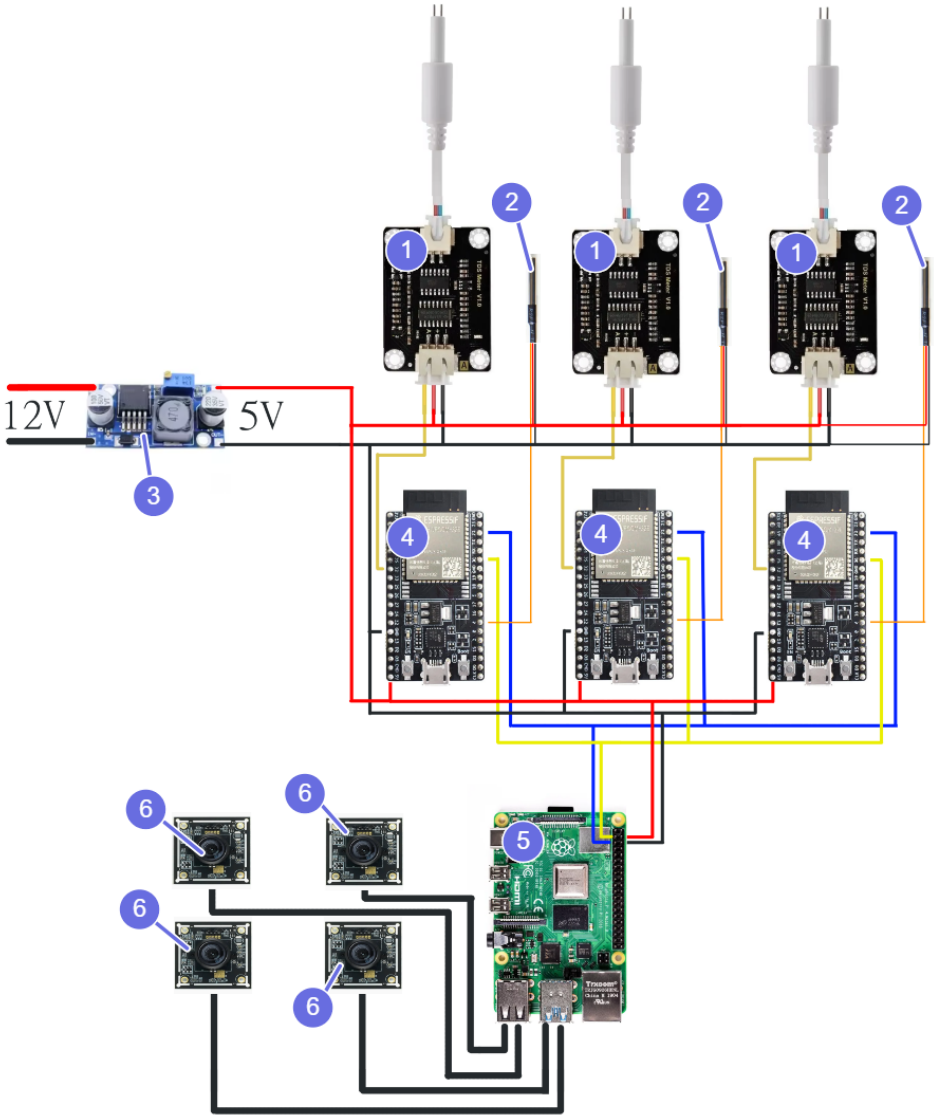
Til hver mikrokontroller er det koblet to sensortyper:

- **DS18B20 Temperatursensor:** En digital sensor som benytter 1-Wire-protokollen. Dette valget sikrer tapsfri dataoverføring selv over de lengre kabellengdene som kreves for å nå de ulike dybdenivåene i systemet.
- **Keyestudio TDS Sensor Meter V1.0:** En analog sensor som benyttes for måling av salinitet. Det analoge signalet digitaliseres av ESP32-enhetens innebygde analog-til-digital-omformer (ADC) før det overføres videre til hovedenheten.

Valget av en TDS-sensor (Total Dissolved Solids) fremfor en dedikert salinitetssensor er gjort på grunnlag av prosjektets budsjettrammer. Begge sensortypene baserer seg på det samme måleprinsippet, måling av vannets elektriske ledningsevne (konduktivitet). Ettersom salinitet i stor grad korrelerer med mengden oppløste stoffer, fungerer TDS-sensoren som en kostnadseffektiv og tilstrekkelig proxy for å demonstrere systemets evne til å overvåke endringer i vannmiljøet. For en prototype er denne løsningen vurdert som hensiktsmessig for å verifisere konseptet, selv om en spesialisert sensor ville vært nødvendig for vitenskapelig nøyaktighet i en ferdig løsning.

4.1.5 Koblingsskjema og strømforsyning

For å få en oversikt over systemets elektriske oppbygning og distribusjon av signaler, er det utarbeidet et koblingsskjema som vist i Figur 21. Dette illustrerer hvordan de sentrale komponentene er knyttet sammen for å sikre pålitelig drift og kommunikasjon.



Figur 21: Koblingsskjema som viser distribusjon av strøm og kommunikasjonslinjer mellom sentralenhet, noder og sensorer.

Tabell 4: Komponentoversikt for koblingsskjema (Figur 21).

Nr.	Komponent	Antall	Rolle i systemet
1	TDS-sensor	3	Måler vannkvalitet (salinitet).
2	Temperatursensor	3	Måler vanntemperatur.
3	Buck-konverter	1	Spenningsregulering (12V til 5V).
4	ESP32	3	Mikrokontroller / Datainnsamlingsnode.
5	Raspberry Pi 4B	1	Sentralenhet og gateway.
6	USB-kamera	4	Samler video data samt convertering til MJPEG.

Som illustrasjonen viser, tar strømforsyningen utgangspunkt i en 12V DC-kilde. For å tilpasse spenningen til elektronikken, benyttes en DC-DC buck-konverter som regulerer denne ned til en stabil 5V-linje. Denne linjen (markert med rødt) forsyner både Raspberry Pi, de tre ESP32-nodene og sensor-parene. Valget av en buck-konverter er gjort for kunne forsyne alle komponenter direkte med samme strøm-kilde i motsetning til å forsyne gjennom Raspberry pi-en.

En viktig detalj i koblingsskjemaet er den gjennomgående jordlinjen (svart), som sikrer felles jord (Common Ground) for alle enheter. Ved å koble alle komponentene til samme referanse, forhindres elektrisk støy som ellers kunne forstyrret datatrafikken på I2C-bussen (blå og gul linje) mellom sentralenheten og nodene.

For en mer detaljert oversikt over de spesifikke fysiske koblingspunktene som er benyttet, henvises det til Tabell 5.

Tabell 5: Oversikt over elektriske koblinger og pin-konfigurasjon i systemet.

Fra enhet	**Pinne/Signal**	**Til enhet**	**Beskrivelse**
RPi 4B	Pin 3 (SDA)	ESP32 Pin 21	I2C Datapinne (felles buss)
RPi 4B	Pin 5 (SCL)	ESP32 Pin 22	I2C Klokkepinne (felles buss)
RPi 4B	Pin 6 (GND)	ESP32 GND	Felles jordreferanse for alle enheter
ESP32	Pin 4 (Digital)	DS18B20	Data fra temperatursensor (1-Wire)
ESP32	Pin 32 (Analog)	TDS Sensor	Analogt signal fra salinitetsmåler
Buck Converter	Out (+5V)	RPi / ESP32	Felles spenningstilførsel (VCC)

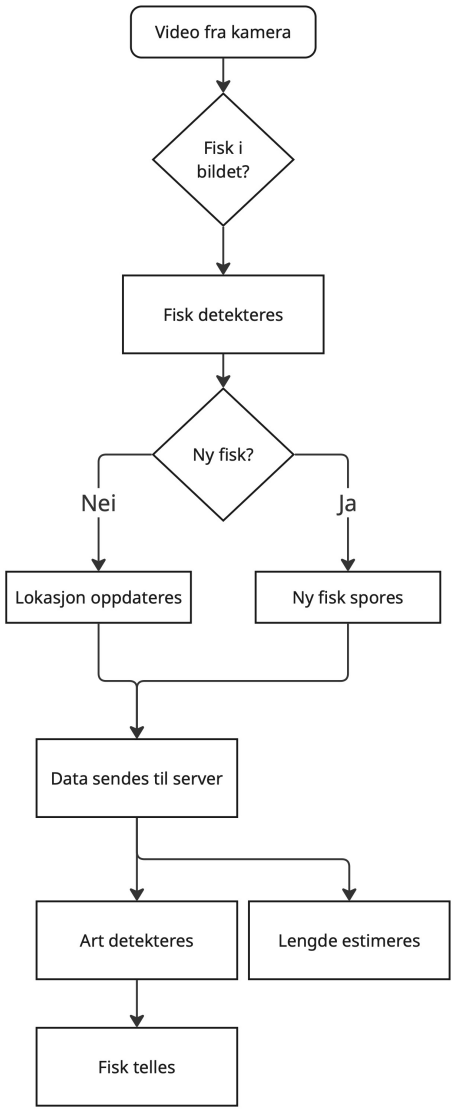
4.2 Programvare

Programvaren fungerer som systemets «nervesystem» og har som hovedoppgave å oversette fysiske signaler fra sensorene til forståelig informasjon for sluttbrukeren. For å oppnå en effektiv og robust løsning er programvarearkitekturen strukturert etter systemets overordnede datasti. Denne stien beskriver informasjonsflyten fra rådata hentes inn lokalt på enheten, via filtrering og prosessering, til den ender opp som ferdigbehandlede resultater i skyen.

Et av de mest kritiske punktene tidlig i denne datastien er håndteringen av visuelle data. Siden videostrømmer genererer store mengder informasjon, er systemet avhengig av å kunne identifisere relevante objekter umiddelbart for å unngå unødig belastning på nettverket. Dette bringer oss til det første sentrale steget i programvaren: bildegjenkjenning.

4.2.1 Bildegjenkjenning

Bildegjenkjenningen utføres i to steg, som beskrevet i seksjon 3.4.2 (Bildebehandling). Dataflyten for implementeringen er illustrert i Figur 22.



Figur 22: Dataflyt for bildeprosessering.

Den initielle deteksjonen utføres i sanntid lokalt på Raspberry Pi-en med en CNN. Her genereres det både en avgrensingsboks og sporingsdata. Dette lagres lokalt og sendes videre over 4G. Ved hjelp av disse dataene utfører en mer avansert transformer algoritme på serveren prediksjon for art og lengde. Med sporingsdata for hver fisk og artsestimatet kan tellingen utføres per laksefisk med en enkel algoritme.

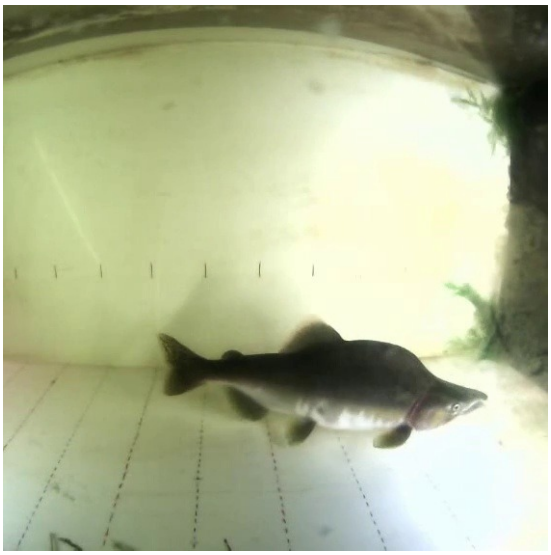
Datasett

De nevrale nettverkene benytter treningsdata fra prosjekter som ligner systemet beskrevet i denne rapporten. Dataene er hentet fra observasjonskammer i elver Drevja og Fjærevassdraget. Disse datasettene er fremskaffet i samarbeid med forsker Jan Davidsen. I disse prosjektene er

villaks, ørret og pukkellaks detektert og telt ved hjelp av nevrale nettverk. Den fysiske implementasjonen samsvarer godt med vår løsning; fisken ledes inn i et observasjonskammer med hvit bakgrunn for å sikre klare bilder og optimal kontrast.

Eksemepilder fra Drevja og Fjærevassdraget er vist under i hendholdsvis Figur 23a og Figur 23b.

(a) Tre laks i observasjonskammer i Drevja.



(b) Pukkellaks i observasjonskammer i Fjærevassdraget.

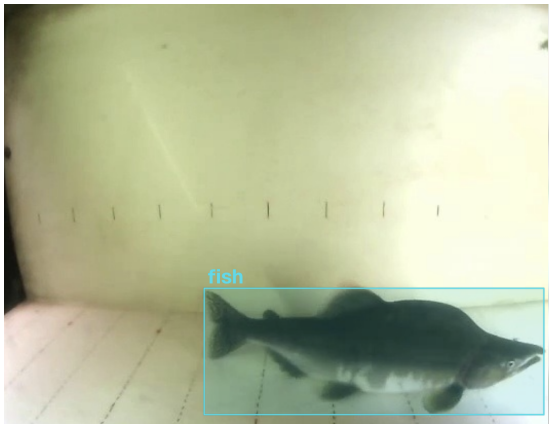
Siden vårt system skal plasseres i et sund, må det forventes noe annerledes lysforhold og potensielt mer grumsete vann. Bruk av kunstig belysning er også begrenset av systemets batterikapasitet.

Bildene er ekstrahert fra kontinuerlige videoopptak gjort over flere måneder. Fra rådataene er tidspunkter med fiskepasseringer detektert av en maskinlæringsmodell (ML-modell), for deretter å bli manuelt validert av en ekspert. Dette sikrer et datasett av høy kvalitet, fritt for falske deteksjoner. Disse bildene har høy overføringsverdi til vår implementasjon. Miljøforskjellen mellom elv og sund er ikke vurdert som utslagsgivende for bildekvaliteten, og dataene er meget egnet for å trene våre egne ML-modeller. Systemet benytter to ulike ML-modeller med delte bruksområder, noe som krever to separate datasett utledet fra de samme rådataene.

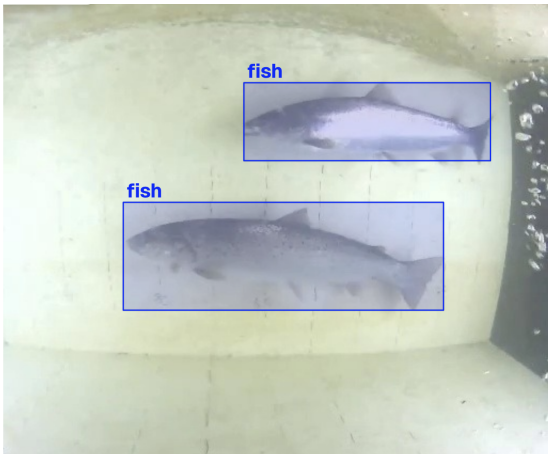
Datasett for fiskedeteksjon

Bilder med fisk ble ekstrahert, og en åpen kildekode-modell ble benyttet for å generere initielle avgrensingsbokser. Disse boksene ble deretter manuelt justert for å sikre nøyaktige treningsdata. Datasettet benytter kun én kategori: «fish», ettersom den lokale gjenkjenningen utelukkende har som oppgave å detektere tilstedeværelse av fisk, ikke bestemme art. Datasettet for fiskedeteksjon består av ~ 3000 bilder av forskjellig laksefisk i observasjonskammer.

Et utdrag fra datasettet er vist under i Figur 24b og Figur 24a.



(a) Pukkellaks i Fjærevassdraget med tilhørende avgrensningsboks.



(b) To fisker i observasjonskammer i Drevja. Avgrensingsbokser vises for begge med kategorien: «fish».

Når datasettet med kategoriserte avgrensningsbokser var etablert, ble modellen for Raspberry Pi-en trent på disse. Ettersom den skybaserte artsgjenkjenningen analyserer utklipp sendt fra Raspberry Pi-en, er beste fremgangsmåte å trene algoritmene direkte på slike bildeutklipp.

Datasett for arts- og lengdeestimering

For å etablere et datasett for arts- og lengdeestimering, ble fiskedeteksjonsmodellen kjørt på de innhentede videoene i kombinasjon med de manuelle artsklassifiseringene fra ekspert. Algoritmen lagret bildeutklipp og tidspunkter for alle deteksjoner. Ved å kryssreferere disse tidspunktene med den ekspertvaliderte dataen, kunne utklippene merkes med riktig art og lengde. Dette resulterte i et omfattende datasett bestående av $\sim 200\,000$ bilder av tusenvis av fisk fra norske elver.

Siden modellen som brukes i produksjon også genererte treningsdagene, vil ikke særegene utklipp eller mønstere modeller lærer påvirke arts- og lengdeestimeringen i reelle scenarioer. En solid modell for den lokale fiskedeteksjonen er et kritisk premiss for både høy datakvalitet og operativ nøyaktighet.

Lokal objektdeteksjon på Raspberry Pi

Objektdetekteringsalgoritmen Ultralytics YOLO26n [10] er implementert for lokal gjenkjenning av fisk på Raspberry Pi-en. Dette er en lettvektsmodell optimalisert for lavt energiforbruk. Det nevrale nettverket er forhåndstrent på Microsofts COCO-datasett (Common Objects in Context) [11]. Den kompakte modellstørrelsen ble valgt for å minimere prosesseringstiden per bilde og muliggjøre sanntidsdeteksjon. Nettverket ble deretter finjustert med prosjektets egne datasett. Under drift analyseres alle innkommende bilder, og modellen predikerer avgrensingsbokser dersom fisk detekteres.

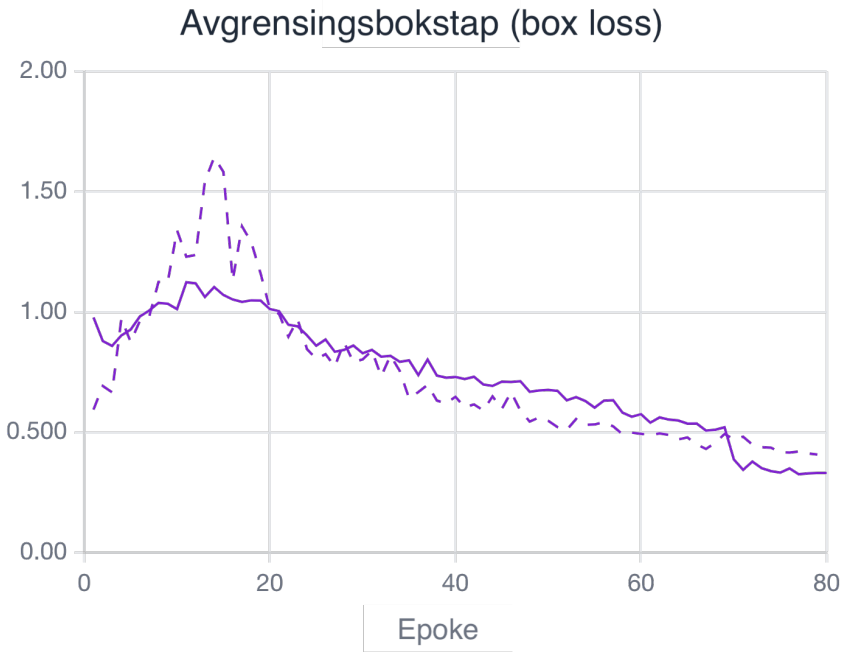
YOLO-trening

Treningsprosessen ble utført i skyen via Ultralytics' plattform. Dette kompenserte for manglende tilgang på dedikert maskinvare og gjorde det mulig å teste ulike hyperparametere og modeller effektivt gjennom treningsprosessen.

Den viktigste indikatoren for god ytelse i dette tilfellet vil være modellens evne til å peke på alle objekter i det relevante bildet. Det er ønskelig at modellen heller peker ut feil objekt noen ganger enn at den er mister en fisk. Utklipp sendes uansett videre til server, og vil derfor dobbeltsjekkes. For å oppnå best mulig ytelse er det valgt en rekke hyperparametre under trening.

Ultralytics biblioteket inneholder en rekke verktøy for datasetutvidelse. Datautvidelse vil gjøre modellen mer robust i tilfeller der lysforhold, grums, fiskestørrelse eller annet ikke er representert i datasettet. Det er tatt i bruk datautvidelse i form av rotasjon, speiling, skalering, mm. En parameter er spesielt viktig i dette tilfellet: mosaikk av treningsdata. Under kjøretid er bilder fra de fire kameraene sydd sammen i ett 2x2 gitter. Modellen utfører deteksjoner på alle bildene simultant. Mosaic hyperparameteret sammenfletter bilder i en 2x2 rute under trening. Dette forbedrer modellens evne til å håndtere flere objektdeteksjoner samtidig betraktelig.

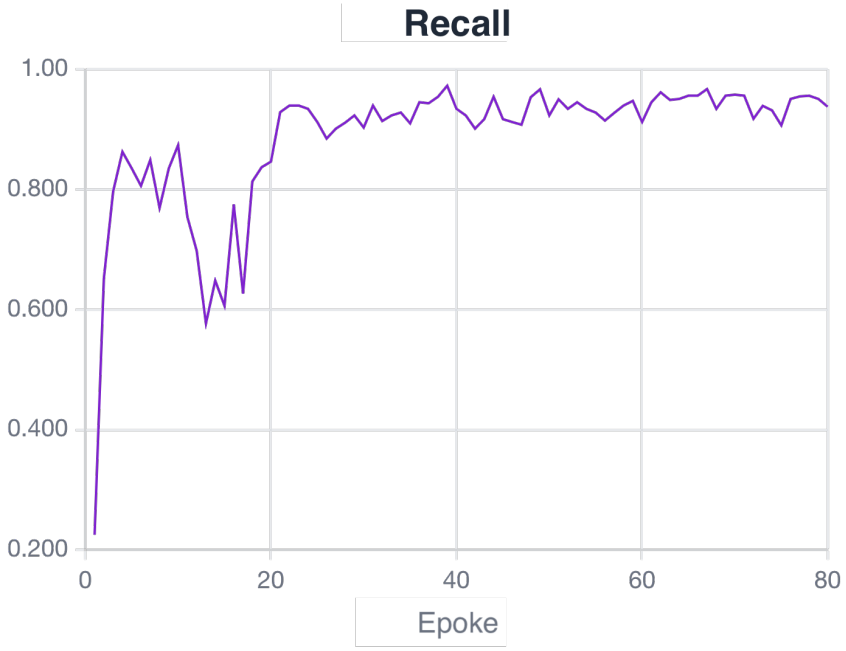
Utviklingen i avgrensingsbokstap (box loss) under trening er vist under i Figur 25. Dette er et mål på hvor mye prediksjonene bommer på de faktiske avgrensingsboksen datasettet. Modellen ble trent over 80 epoker (epochs).



Figur 25: Avgrensingsbokstap under trening. Stiplet linje viser tap på valideringsdata, heltrukken linje viser tap på treningsdata.

For å unngå at modellen overtilpasser til treningsdataen (overfitting) er avgrensingsbokstapet monitorert under trening. Dersom treningstapet (heltrukken linje) blir betydelig lavere enn valideringstapet (stiplet linje), tyder det på overfitting [12]. Overtilpasning skylders lite variasjon i datasettet. Siden det tilgjengelige datsettet kun inneholder ~ 3000 bilder fra 2 lokasjoner, var det nødvendig å begrense antall treningsepoker. Fra figuren synes det at valideringstapet øker noe de siste 10 epokene mens treningstapet synker. Modellen ble derfor stoppet under trening etter 80 epoker.

En metrikk som er viktig for modellen i dette spesifikke bruksområde er i hvor stor grad modellen klarer å detektere en fisk dersom det faktisk er en fisk i bildet (recall) [13]. Det er ønskelig med lite falske negativer ($< 1\%$) fra systemkrav 4.1, altså høy recall. Recall utviklingen er vist under i Figur 26.



Figur 26: Utviklingen av recall under trening.

Siden artsklassifiseringen senere kan dobbeltsjekke om det er en fisk i bildet er det ønskelig å ha en modell som er skråsikker. Med flere treningsepoker ville modellen ha oppnådd bedre recall, men på bekostning av mulig overtilpasning. Modellen har oppnådd en recall verdi på 93.8%. Dette tilsier falske negativer i mer enn 6% av tilfeller. Dersom det kan antas at recall verdien er representativ til reelle forhold vil deteksjon allikevel fungere godt. Siden det i realiteten vil tas flere bilder av samme fisk i tiden den passerer, vil systemet under kjøretid kunne detektere og senere telle fisk mye oftere enn i 93% av tilfeller. I praksis kan falske negativer nærme seg $\approx 0\%$ dersom fisken er synlig gjennom hele passasjen (gitt antakelsen over).

YOLO-implementering

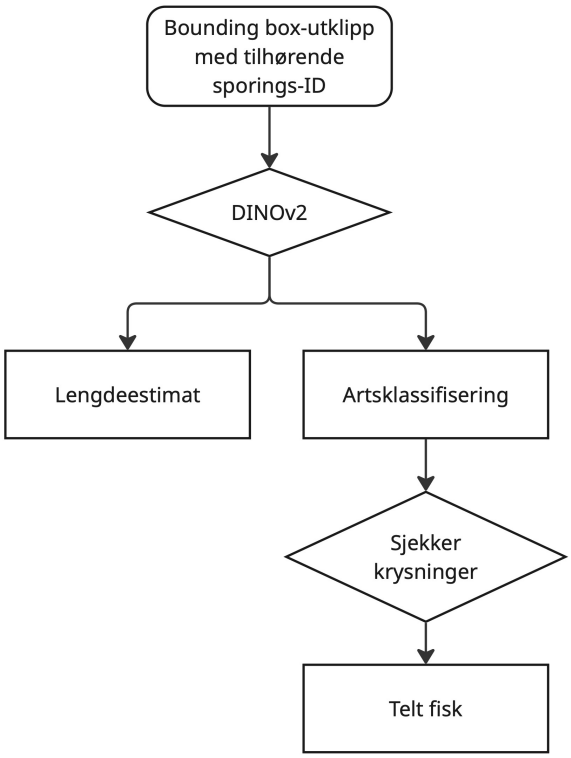
Etter trening ble YOLO-modellen kompilert til NCNN-formatet for å optimalisere kjøretiden på Raspberry Pi-en. I tillegg ble videostrømmene fra de fire kameraene sydd sammen til ett felles bilde, slik at modellen kunne analysere hele deteksjonsområdet simultant. Dette forenklet sporingen av fisk på tvers av kameraenes synsfelt betraktelig.

Ultralytics-biblioteket for Python tilbyr innebygd objektsporing via algoritmen BoT-SORT [14]. Sporingsdataene benyttes til å telle fisk ved å overvåke objektets x-koordinat relativt til en definert passasjegrense; en telling registreres når grensen krysses.

Gjennom sporingen tildeles hver fisk en unik sporings-ID. Denne ID-en sendes til serveren sammen med fiskens posisjonsdata i deteksjonsområdet. Sporingsdataene og den unike ID-en er essensielle, da de lar serveren aggregere data fra flere utklipp for å oppnå sikrere estimater av art og lengde. Ved positiv deteksjon lagres avgrensningsboksene og sporingsdataene midlertidig lokalt, før de overføres til serveren.

Bildeprosessering på server

Artsklassifisering og lengdeestimat utføres på serveren ved hjelp av Meta AI sin DINOv2-modell (ViT) [15]. For å kunne utføre begge disse komplekse oppgavene samtidig, er modellen bygget med en Multi-Task-arkitektur [16]. Estimeringene fra serveren blir deretter koblet sammen med sporingsdataene fra den lokale YOLO-modellen for å gi en endelig og nøyaktig telling av hver enkelt art som svømmer gjennom systemet. Dataflyten er vist under i Figur 27.



Figur 27: Dataflyt for bildeprosessering på server.

Datasett og «Ingen fisk»-strategi

For å trene servermodellen ble det etablert et stort og variert datasett med utklipte bilder av fisk fra de innsamlede videoene. Et velkjent problem i overvåking av lakseelver er imidlertid at fordelingen av fiskearter er svært ubalansert. Datasettet vårt gjenspeiler denne skjevheten:

- Laks: $\sim$ 50 000 bilder
- Ingen fisk: $\sim$ 20 000 bilder
- Ørret: $\sim$ 100 000 bilder
- Pukkellaks: $\sim$ 6 000 bilder
- Røye: $\sim$ 6 000 bilder

Kategorien «Ingen fisk» spiller en essensiell rolle i systemets robusthet. I stedet for å kreve at den lette YOLO-modellen på Raspberry Pi-en skal være 100 % sikker før den sender et bilde, lar vi den operere med en lavere terskel for gjenkjenning. Det er langt bedre å sende ett bilde

for mye over nettverket enn ett for lite. Dersom YOLO-modellen plukker opp en skygge, en grein eller bølger som ligner på fisk, sendes dette til serveren. Serveren, som har betydelig mer datakraft, bruker «Ingen fisk»-klassen for å gjenkjenne og filtrere ut disse feildeteksjonene.

Effektiv finjustering med LoRA

For å tilpasse den omfattende DINOv2-modellen til vårt spesifikke datasett uten å kreve enorme beregningsressurser, benyttes Low-Rank Adaptation (LoRA) [17]. Tradisjonell finjustering krever oppdatering av alle parametere i modellen, noe som er svært kostbart. LoRA baserer seg på hypotesen om at endringene i vektene under læring har en lav "indre dimensjon".

For en gitt vektmatrise $W_0 \in \mathbb{R}^{d \times k}$ fryses de opprinnelige vektene, og oppdateringen representeres ved et dekomponert produkt av to lav-rangs matriser, A og B:

$$W = W_0 + \Delta W = W_0 + BA \tag{1}$$

hvor $B \in \mathbb{R}^{d \times r}$ og $A \in \mathbb{R}^{r \times k}$, og rangen $r \ll \min(d, k)$. Dette reduserer antallet trenbare parametere drastisk, samtidig som modellen beholder den dype semantiske forståelsen fra den forhåndstrente DINOv2-rygggraden.

Matematisk formulering av tapsfunksjonen

Siden systemet må løse både en klassifiseringsoppgave (art) og en regresjonsoppgave (lengde) samtidig, benyttes en sammensatt tapsfunksjon.

For å håndtere den store ubalansen i artsfordelingen (eksempelvis 10 000 bilder av ørret mot 6 000 av røye), brukes Focal Loss [18]. Dette er en videreføring av Cross-Entropy som nedvekter bidraget fra enkle eksempler og tvinger modellen til å fokusere på de vanskelige tilfellene (minoritetsartene). Tapsfunksjonen for artsbestemmelse defineres som:

$$\mathcal{L}_{art} = -\alpha_t (1 - p_t)^{\gamma} \log(p_t) \tag{2}$$

hvor p_t er modellens estimerte sannsynlighet for riktig art, α_t er en vektingsfaktor for å balansere klassene, og fokuseringsparameteren γ (her satt til 2.0) kontrollerer vekten av vanskelige eksempler.

For lengdeestimering benyttes Huber Loss [19] bygd på $L2$-tap, som fungerer som et kvadratisk tap nær null, men som er lineært for større feil. Dette gjør modellen mer robust mot ekstremverdier enn vanlig $L2$-tap:

$$\mathcal{L}_{lengde} = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{for } |y - \hat{y}| \leq \delta \\ \delta(|y - \hat{y}| - \frac{1}{2}\delta) & \text{ellers} \end{cases} \tag{3}$$

hvor y er den sanne lengden og $\hat{y}$ er den predikerte verdien. Siden ikke alle bilder i datasettet har annotert lengde, brukes en binær maske $M \in \{0, 1\}$ slik at bilder uten lengdedata ikke påvirker gradientoppdateringen for regresjonshodet.

Den totale tapsfunksjonen som minimeres under trening er den vektede summen:

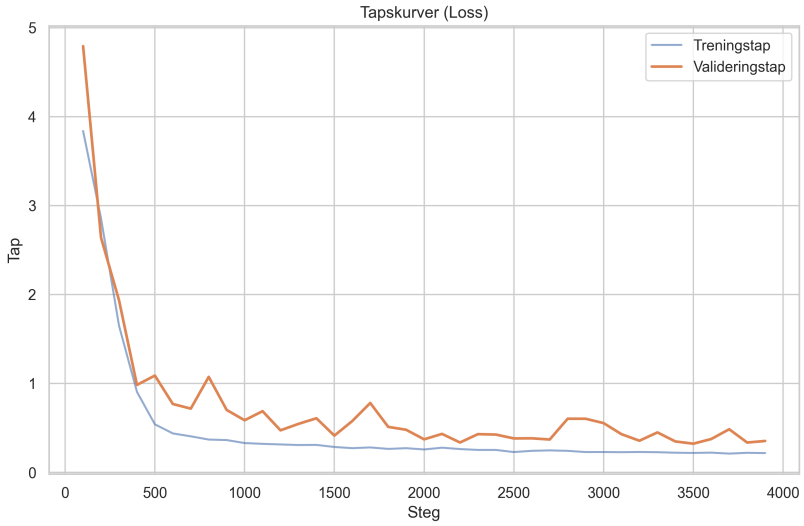
$$\mathcal{L}_{total} = \mathcal{L}_{art} + \lambda \cdot (M \cdot \mathcal{L}_{lengde}) \tag{4}$$

Her er skaleringsfaktoren λ satt til 0,05 for å sikre at de to oppgavene læres i et balansert tempo, uten at den numerisk større regresjonsfeilen dominerer over klassifiseringen.

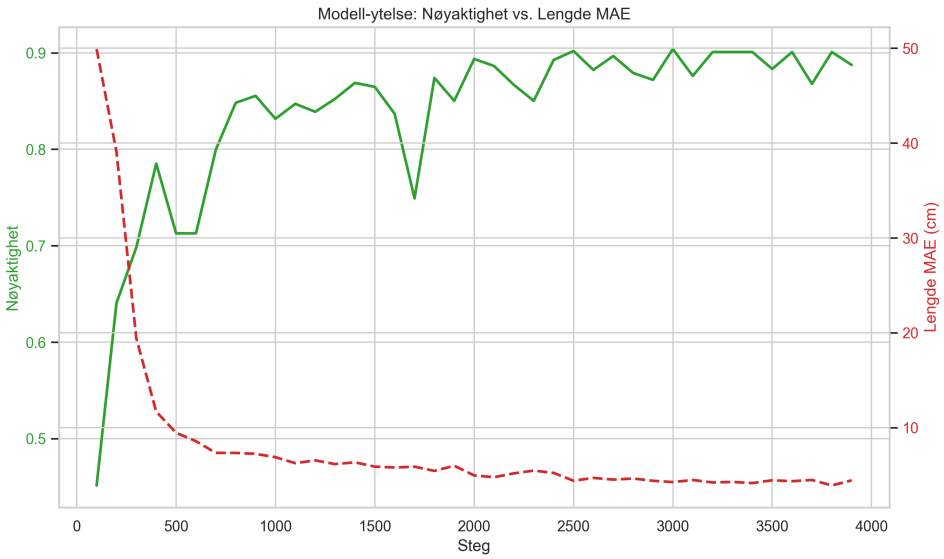
Trening og Ytelse

Treningen ble evaluert fortløpende, og resultatene viser at arkitekturen håndterer de komplekse oppgavene svært godt. I Figur 28 kan vi se at trenings- og valideringstapet stabiliserer seg tidlig, med et valideringstap på rundt 0.3 og et treningstap på rundt 0.2, noe som indikerer god generalisering uten overtilpasning.

Utviklingen i nøyaktighet er vist i Figur 29. Nøyaktigheten for artsklassifisering flater ut på imponerende ~ 90 %. Dette er et svært godt resultat tatt i betraktning at mange av utklippene under reelle forhold er delvis skjult, ute av fokus, eller kun viser halen på fisken. For regresjonsoppgaven stabiliserte modellen seg på en gjennomsnittlig absolutt feil (MAE) på omlag 3-4 cm. Dette gir forvaltningen presise nok estimater til å vurdere sammensetningen i laksebestanden.



Figur 28: Utvikling av trening- og valideringstap over tid.



Figur 29: Utvikling av klassifiseringsnøyaktighet og MAE for lengde.

Deployment og Kjøretid

I produksjon utgjør servermodellen det siste og avgjørende steget i datastien. Når YOLO-modellen på Raspberry Pi-en detekterer et objekt den tror er en fisk, sendes koordinatene for avgrensningsboksen, en ID-nummer (fra BoT-SORT-trackeren) og selve bildeutklippet over 4G-nettverket.

Ved ankomst på serveren prosesseres bildet av DINOv2-modellen. Det returneres da en sannsynlighetsfordeling av de forskjellige artene, samt et lengdeestimat. Siden vi sporer fisken med en unik sporings-ID over flere videobilder, kan serveren samle opp flere estimater av den samme fisken mens den svømmer gjennom kammeret. Dette gjør at vi kan bruke flertallsbeslutninger og gjennomsnittsberegninger over flere bilder av samme fisk for å garantere riktig art og lengde før den endelige tellingen registreres i databasen.

4.2.2 Sensorer

Fullstendig kildekode for mikrokontrollerne og sentralenheten finnes i Tillegg A. Original koden for data-innhenting til temperatur og TDS sensoren er hentet fra produsenten av sensoren og funnet i kilde [20] for temperatur og kilde [21] for TDS sensoren.

Implementeringen fokuserer på en pålitelig avlesning av sensorens interne minne over 1-Wire-protokollen. Den logiske prosessen for temperaturmåling er beskrevet i Kode 1.

```
FUNKSJON hentTemperatur():
    1. Skann bussen for å finne sensorens adresse
    2. Utfør CRC-sjekk for å verifisere feilfri adresseoverføring
    3. Send kommando: "Start analog-til-digital konvertering" (0x44)
    4. Vent til målingen er lagret i sensorens minne
    5. Les rådata fra minnet (0xBE)
    6. Kombiner mottatte bytes til et 16-bits heltall
    7. Divider med skaleringsfaktor 16.0 for Celsius-format
    8. RETURNER temperatur
```

Kode 1: Pseudokode for digital temperaturmåling.

TDS-sensoren leverer et analogt signal som må behandles for å fjerne elektrisk støy og kompensere for temperatursvingninger i vannet. Implementeringen benytter et glidende gjennomsnittsfilter for å sikre stabile verdier, som beskrevet i Kode 2.

```
FUNKSJON hentTDS(gjeldende_temperatur):
    1. Les analog verdi fra ADC-pinne (0-4095)
    2. Legg verdien inn i en sirkulær buffer (gjennomsnittsfilter)
    3. Sorter bufferen og hent ut medianverdien for å fjerne ekstremmålinger
    4. Konverter medianverdi til spenning (VREF / 4095.0)
    5. Beregn kompenasjonsfaktor basert på gjeldende_temperatur
    6. Bruk polynomisk formel for å beregne TDS-verdi fra kompensert spenning
    7. RETURNER kalibrert TDS-verdi
```

Kode 2: Pseudokode for innhenting og filtrering av TDS-data.

I2C-kommunikasjon

Kommunikasjonen mellom mikrokontrollerne og sentralenheten er implementert som en forespørsel-svar-modell (polling) [22].

På mikrokontroller (Slave): Dataene pakkes i et standardisert JSON-format. Når sentralenheten etterspør data via en OnRequest-event, sendes den nyeste JSON-pakken som en bytestrøm.

På sentralenhet (Master): Siden I2C-overføringer kan inneholde ugyldige tegn ved start og slutt av bufferen, er det implementert en renseprosess for å sikre gyldig JSON-struktur før deserialisering.

```
FUNKSJON lesDataFraNode(adresse):
    1. Åpne I2C-forbindelse og les 128 bytes fra slave-adresse
    2. RENSING: Behold kun tegn i det synlige ASCII-området (vask bort støy)
    3. STRING-PARSING: Identifiser posisjonen til '{' og '}'
    4. HVIS gyldig JSON-struktur er funnet:
        a. Dekod tekststrengen til et dataobjekt
        b. Legg til tidsstempel fra systemklokken
        c. RETURNER ferdig formatert datapakke
    5. ELLERS: Loggfør tolkningsfeil og RETURNER null
```

Kode 3: Pseudokode for datahenting og rensing på sentralenhet.

4G-kommunikasjon og Web-overføring

Når dataene er ferdig behandlet og tidsstemplet, sendes de til skyen via HTTP POST. Implementeringen fokuserer på systemets robusthet ved ustabil mobildekning.

```
FUNKSJON sendTilWeb(data):
    FORSØK:
        Send data til backend-URL med en tidsbegrensning (timeout) på 5 sekunder
        HVIS server svarer med status "200 OK":
            Bekreft vellykket overføring i loggen
        ELLERS:
            Skriv feilkode fra server til terminal for feilsøking
    FEILHÅNDTERING (Ved timeout eller manglende dekning):
        Fang opp unntaket slik at programmet ikke krasjer
        Lagre gjeldende sending og fortsett til neste måleperiode
```

Kode 4: Pseudokode for robust dataoverføring via 4G.

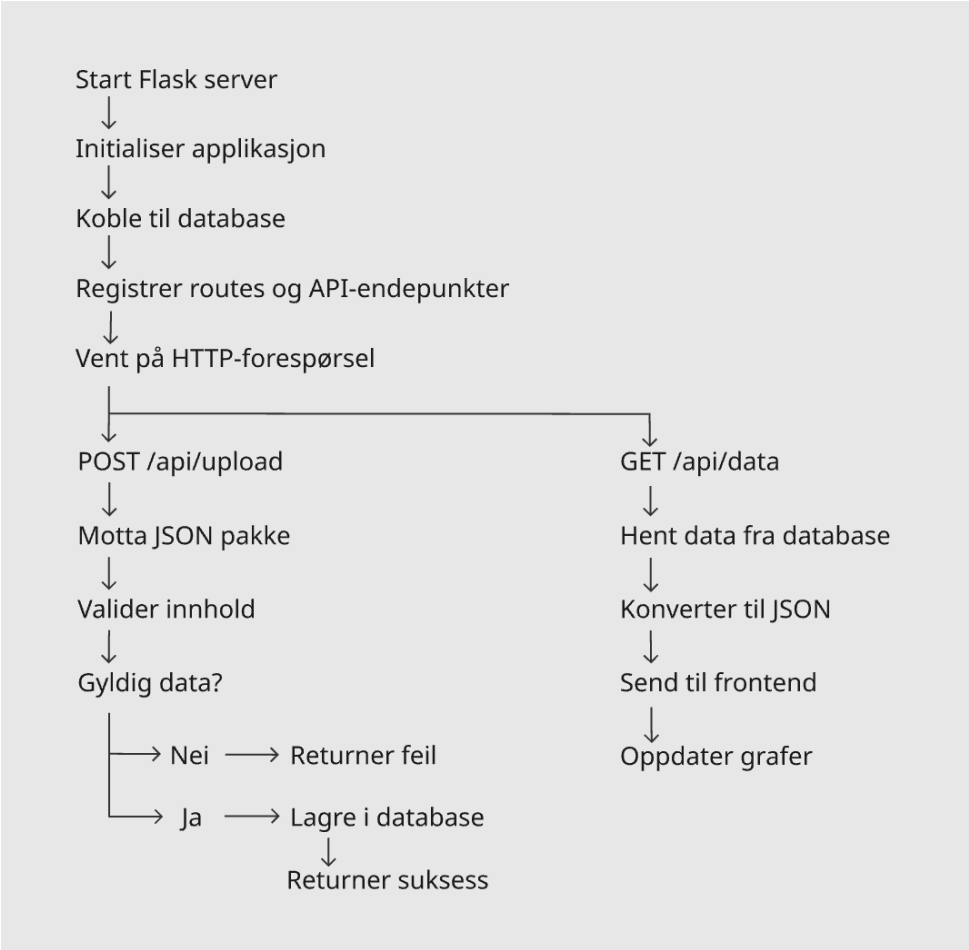
4.2.3 Nettside

Den fullstendige koden til nettsiden ligger i github, som man finner i Tillegg A. Implementasjonen av nettsiden er gjort ved programmering i kodespråkene Python, JavaScript (JS), HyperText Markup Language (HTML), og Cascading Style Sheets (CSS). Python er brukt til backend, API,

og databasehåndtering. JS er brukt til frontend-logikk som knapper og dynamisk oppdatering. HTML brukes til å definere strukturern til nettsiden, mens CSS brukes til utforming og styling.

Flask er brukt som backend-rammeverk for webapplikasjonen. Flask bygger på Web Server Gateway Interface (WSGI), som er en standard for kommunikasjon i Python. Den fungerer som en kallkonvensjon for å videresende forespørsler fra webservere til webapplikasjoner [23]. SQLAlchemy er et bibliotek som brukes for å hente og manipulere data i databaser. Biblioteket gjør det mulig å kommunisere med databasen gjennom Python-objekter istedet for å skrive SQL direkte [24]. API-endepunktene implementeres som Flask Blueprints som registreres i app/___init___.py ved hjelp av app.register_blueprint(). Dette kobler URL-rutene til de respektive Python-modulene som implementerer funksjonaliteten.

Figur 30 viser programflyten for nettsiden.



Figur 30: Nettsidens programflyt

Serveren er hendelsesstyrt: Den venter på forespørsler og utfører handlinger basert på hvilken URL som kalles. Data oppdateres periodisk, og hentes ved HTTP-forespørsler fra frontend, noe som reduserer belastning på serveren.

Tabell 6 bygger videre på Figur 30 ved å vise grunnlaget for backend strukturen mer detaljert. Koden er realisert i VSCode med versjonskontroll håndtert i GitHub.

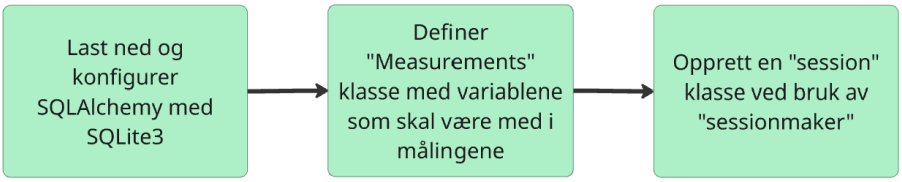
Struktur	Forklaring
.gitignore	Forteller github hvilke filer og mapper som skal ignoreres, slik at de ikke blir tracked, staged, eller committed til repository.
app.py	Kjører programmet
instance/	
database.db	Databasen skrevet i SQLite
app/	
___init___.py	Funksjonen create_app() er implementert - den tar inn blueprints som er definert i .py filene under routes/.
db.py	Én klasse for målingene og én for deteksjoner er definert. Databasen er initialisert.
routes/	
health.py	Implementerer protokoll som sjekker at serveren fungerer korrekt
home.py	Implementerer hjemmesiden
upload.py	Validerer input data og legger den isåfall i databasen
static/	
style.css	Gjør det mulig å endre utseende på nettsiden, som for eksempel fargen på teksten og bakgrunnen
templates/	
home.html	Skaper strukturen til hjemmesiden. Denne bruker base.html som en mal slik at navigasjonsbaren som ligger øverst på siden er lik på alle undersider
base.html	Mal for navigasjonsbar

Tabell 6: Grunnlag for backend strukturen

Oppsett av denne strukturen fulgte instruksene fra videoserien «Flask Tutorials» av Tech with Tim, spesielt følgende videoer: [25], [26], [27], [28], [29], [30]. Innholdet i .html og .css filene følger teknikkene fra videoen [31].

Nye undersider kan implementeres ved å opprette egne Python-filer for Flask-ruter, samt tilhørende HTML-maler. Når en ny rute er registrert, kan siden nås gjennom den tilhørende URL-en. For eksempel tar URL-en https://stimle.elektra.io/data brukeren til datasiden med sensorgrafer og deteksjoner. Hver av .html filene i implementasjonen arver fra base.html for å få samme navigasjonsbar. Den er basert på komponenter fra Bootstrap, som inneholder ferdige maler og elementer for webdesign [32]. De ulike HTML-filene inneholder også spesialisert innhold, som tekst, grafer og bilder tilpasset den enkelte siden. CSS-koden er hovedsakelig samlet i style.css.

Denne artikkelen [33] ble brukt aktivt i oppsettet av upload.py, som validerer data og lagrer den i databasen. Figur 31 viser overordnet fremgangsmåte. Når databasen er konfigurert med session, kan data hentes ved hjelp av query(). For å lagre data opprettes en ny måling, før session.add() og session.commit() benyttes for å skrive til databasen. Denne mekanismen benyttes blant annet i upload-funksjonen.



Figur 31: Overordnet prosess for SQLAlchemy oppsett [24]

Pseudokode for Funksjonen upload() vises i Listing 5. Den fungerer som mottakspunkt for sensordata fra Raspberry Pi. Oppgaven er å ta imot pakken, validere innholdet og lagre målingen i databasen. Dataene sendes som en pakke i JSON-format til backend via API-endepunktet /api/upload. Funksjonen mottar forespørselen og leser pakken, og konverterer JSON-innholdet til Python-struktur. Deretter valideres pakken for å sikre at verdiene eksisterer og har riktig datatype. Dersom pakken er gyldig, hentes relevante felt ut, og sensormålingene lagres i databasen.

```
FUNKSJON upload()
    Motta datapakke (JSON)

    Valider datapakken:
        Hvis ugyldig:
            returner feil

    Hent ut relevante verdier (pi_id, ts, sensorverdier)

    Hvis depth finnes i pakken:
        sett depth lik denne verdien
    Ellers:
        sett depth = 0

    Iterer gjennom alle sensorer:
        Lagre måling i databasen

    Returner suksessmelding
```

Kode 5: Pseudokode for upload()

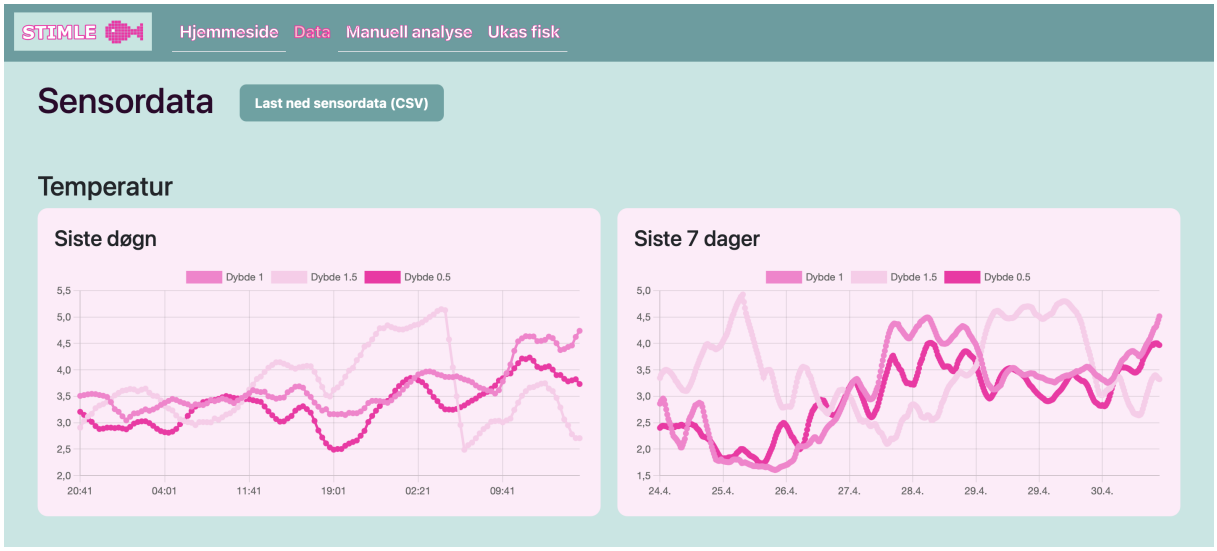
Data hentes fra databasen via API-endepunktet /api/data. Når dette endepunktet kalles, brukes funksjonen get_data() til å hente målingene. Disse konverteres til JSON-format, slik at de kan sendes til frontend. I frontend hentes dataene med fetch(), og behandles videre før de brukes til visualisering av målingene.

Som beskrevet i seksjon 3.5 (Nettside) skal dybdeprofilene for temperatur og salinitet visualiseres grafisk. De motatte dataene filtreres basert på tidsintervall, og deles inn i målinger fra siste døgn og siste uke. For hver sensortype tegnes det to grafer, en for hvert tidsintervall. Dette gir brukeren oversikt over både kortsiktige og langsiktige variasjoner. På grafene over den siste uka er det valgt å la x-aksen vise dato for når målingen er tatt, og ikke klokkeslett. Det er for å unngå uoversiktlige akser som er vanskelige å lese. Grafene over det siste døgnet derimot har klokkeslettet på x-aksen.

For å få en glattere kurve og gjøre det lettere å identifisere trender, benyttes Simple Moving Average (SMA) på grafene. Det er en metode hvor gjennomsnittet av et fast antall datapunkter over et bestemt tidsrom beregnes, og representeres som et punkt i en ny graf. Det er implementert en funksjon simpleMovingAverage(values, N), som tar inn målinger i values og antall datapunkter per tidsvindu i N. Funksjonen benytter en løkke som beregner gjennomsnittet ved formelen

$$SMA = \frac{x_1 + x_2 + ... + x_n}{n} \tag{5}$$

der n er antall datapunkter i hvert tidsvindu (tilsvarende parameter N), og vinduet forskyves med et steg for hver iterasjon [34].



Figur 32: Dataside

Figur 33: Dataside

For at brukeren skal kunne hente historisk data er det implementert to knapper på nettsiden. Når disse trykkes, lastes det ned en CSV-fil med tilhørende data, strukturert i tabeller. Tabellene er basert på klassene «Measurements» og «Detections», vist i Listing 6 og 7.

```
class Measurements(Base):

    id = Hvilket system
    pi_id = Hvilken raspberry pi
    sensor_name = Temperatur eller TDS (Total dissolved solids)
    ts = Tidsstempel
    sensor_value = Sensorverdien som er målt
    depth = Hvilken dybde
```

Kode 6: Pseudokode for klassen Measurements

```
class Detections(Base):

    id = Hvilket system
    pi_id = Hvilken raspberry pi
    fish_id = Hvilken fisk
    data = artssannsynlighet og fiskens lengde
    image_path = url til bildet av fisken
    ts = Tidsstempel
```

Kode 7: Pseudokode for klassen Detections

Når knappene aktiveres, kalles funksjonene detections_csv() og csv_download(), som genererer filene ved å hente data fra databasen. Figur 34 og 35 viser en rad i de to CSV-filene. Temperatur måles i grader, mens TDS måles i milligram per liter.

Pi-id	Timestamp	Dybde	Temperatur	TDS
1	2026-04-30 07:41:01	1.5	2.6037	901.14

Figur 34: CSV-fil for sensormålingene

ID	Pi-id	Fish-id	Data	Timestamp
1	−1	824236567	{'Species_data': {'Ukjent fisk': 0.5002, 'Ørret': 0.2186, 'Laks': 0.2104, 'Røye': 0.043, 'Pukkel laks': 0.02, 'Ingen fisk': 0.0078}, 'Fish_length': 12.0}	2026-05-01 09:13:54

Figur 35: CSV-fil for deteksjoner

Selve nettsiden kan bli sett på `https://stimle.elektra.io/`.

5 Verifikasjon, test og validering

Dette kapittelet dokumenterer den systematiske testingen av prototypen for å sikre at både tekniske systemkrav (verifikasjon) og overordnede brukerkrav (validering) er oppnådd. Siden systemet er i en tidlig utviklingsfase, er testene utført i kontrollerte omgivelser på land, men med metodikk som simulerer reelle forhold gjennom bruk av historisk videomateriale og fysiske fiskemodeller.

5.1 Verifikasjon av systemkrav

5.1.1 Verifikasjonsplan

Verifikasjonsplanen av systemet opp mot de systemkravene kravene definert i Tabell 3. For å sikre en strukturert gjennomføring er verifiseringen delt inn i to faser: først testes systemets kjernefunksjonalitet (skal-krav), etterfulgt av en evaluering av tilleggsfunksjonalitet (bør-krav). Tabell 7 viser en verifikasjonsplan for skal kravene til systemet.

Tabell 7: Verifikasjonsplan for systemets skal-krav (MVP-fokus).

Nr.	Metode for verifikasjon	Verifikasjonsmål
2. Utforming		
2.6	Montere og demontere systemet uten skade på materialet	Montering/demontering skal ta under 3 timer
3. Databehandling		
3.1	Sammenligningstest i vannbad mot kalibrert termometer og salinitetsmåler.	Avvik på temperatur og TDS innenfor ± 15%
3.2	Inspeksjon av montert prototype og sjekk av loggdata.	Bekreftet data fra tre distinkte dyp (0.5m, 1m, 1.5m)
3.3	Kontroll av database-oppdatering via 4G-tilkobling.	Vellykket datapakke mottatt i skyen uten WiFi-støtte
3.4	Inspeksjon for kontinuitet av datastrømmen på nettside.	Stabil datautveksling mellom noder og sentralenhet
3.5	Inspeksjon av sporet-ID etter brudd på nett.	Verifisert utklipp og ID er send etter gjenoppretting av 4G
4. Bildedeteksjon		
4.1	Kjøring av testvideoer med kjent antall passeringer.	Falske negativer $< 1\%$
4.2	Klassifiseringstest med bilder av ulike arter.	Korrekt skille mellom laks, ørret og pukkellaks i 90% av tilfeller
4.3	Måling av referanseobjekt i kamerasonen.	Lengdeestimat innenfor ±20% av faktisk mål
5. Grensesnitt		
5.1	Sammenligning av bildebehandlingslogg mot sanntidstall i GUI.	Korrekt antall fisk vises i sanntidsdashboard
5.2	Brukertest av grafiske plott for temperatur og salinitet.	Dybdeprofiler visualiseres grafisk og korrekt
5.3	Funksjonstest av eksportfunksjon i nettleser.	Mottatt fil inneholder korrekte historiske data

Ettersom nåværende prototype bare er bygget for tester på land vil mesteparten av bør-testene ikke utføres på denne. Bør-testene vil bli mer aktuelle for en fremtidig versjon.

Tabell 8: Verifikasjonsplan for systemets bør-krav.

Nr.	Metode for verifikasjon	Verifikasjonsmål
1. Energihåndtering		
1.1	Belastningstest med fulladet batteribank.	Systemet opererer i $>$ 72 timer
1.2	Måling av ladeeffekt fra solcellepanel under varierende lys.	Bekreftet lading til batteribank
2. Utforming		
2.1	Kontrollmåling av rammekonstruksjon.	Bredde innenfor 100 ± 20 cm
2.2	Kontrollmåling av rammekonstruksjon.	Lengde/Høyde mellom 200 og 400 cm
2.3	Stabilitetstest i åpent vann.	Systemet flyter stabilt uten å kantre
2.4	Test av fysisk ledesystem med simulerte objekter.	Objekter ledes suksessfullt til kamerasonen
2.5	Trykktest eller nedsenking av kapsling over tid.	Verifisert fravær av fuktighet (IP67)

5.1.2 Gjennomføring og testresultater

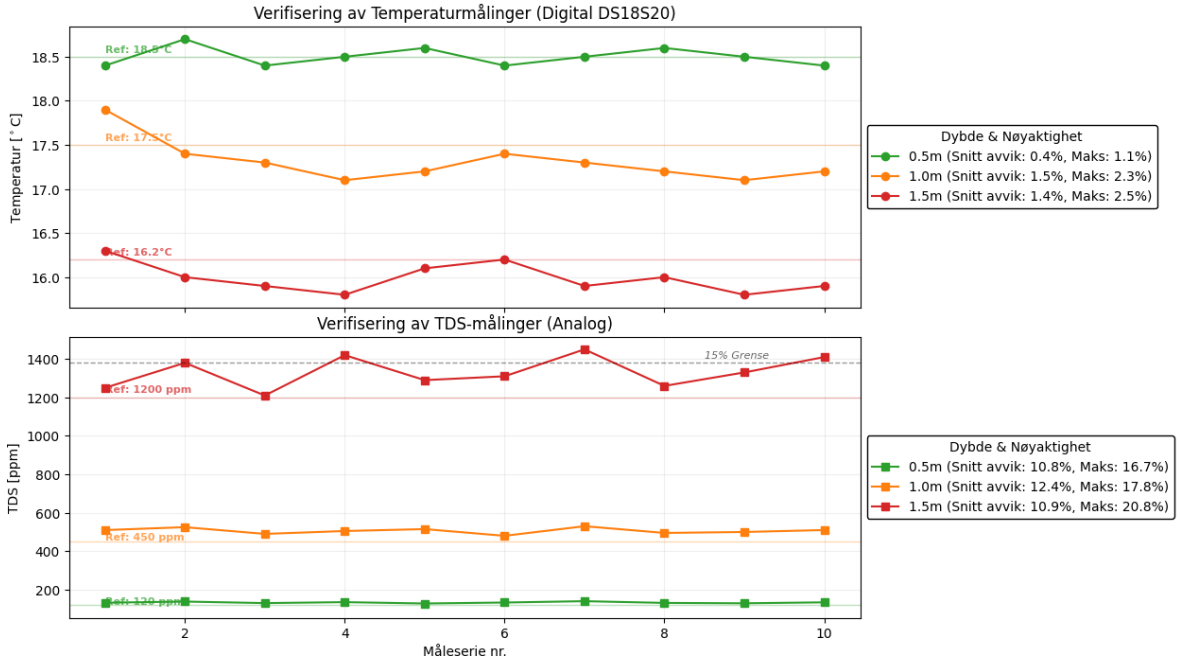
Test av utforming (skal)

Test 2.6 ble gjennomført ved å ta tiden på hvor lang tid det tok å montere systemet fra basekomponenter til det fysiske rammeverket var montert. Dette tok **2 timer og 14 minutter**, slik at systemkrav 2.6 er **oppnådd**.

Test av databehandling

Test 3.1 ble utført ved å sammenligne seks måleserier (tre for temperatur og tre for TDS) mot kalibrerte referanseinstrumenter. Hver måleserie bestod av 10 enkeltmålinger tatt med fem minutters intervall. Som referanse for temperatur ble det benyttet et digitalt termometer fra Biltema [35], og for salinitet ble det benyttet en HM Digital TDS-3 [36] håndholdt måler.

For å verifisere spennet i TDS-sensoren ble det benyttet vannprøver med varierende saltinnhold (fra kranvann til konsentrert saltoppløsning) ved romtemperatur. Temperatursensorene ble testet i vann med temperaturer mellom 16 °C og 19 °C. Punktresultatene fra målingene er illustrert i Figur 36.

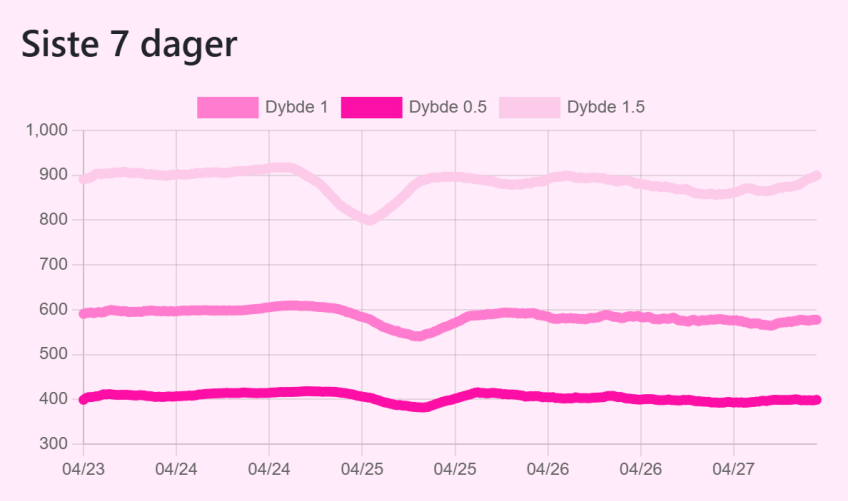


Figur 36: Verifisering av temperatursensorer (DS18S20) og TDS-sensor (Keystudio V1.0) mot referanseverdier for tre dybdenivåer.

Fra Figur 36 ser man at temperatursensorene i hvert dybdenivå er svært presise og ligger godt innenfor kravet på ±15 %. For TDS-sensoren, som benyttes som en indikator for salinitet, er resultatene mer varierende. Selv om gjennomsnittlig avvik for flere av seriene ligger innenfor marginen, viser dataene at samtlige dybdenivåer hadde enkeltmålinger med et maksimalt avvik som oversteg 15 %.

Dette skyldes trolig kombinasjonen av elektrisk støy i de analoge kretsene og behovet for mer omfattende kalibrering av sensoren. Det er verdt å merke seg at avviket var tilstede selv ved bruk av kranvann, som har relativt lave verdier av TDS sammenlignet med sjøvann. Basert på dette kan det konkluderes med at test 3.1 **delvis oppnådde** systemkravet, men at sensoren krever forbedret skjerming og kalibrering for pålitelig drift i felt.

Ved inspeksjon av nettsiden vist i Figur 37, observeres det at data fra dybdepunktene på 0,5 m, 1 m og 1,5 m sendes kontinuerlig med 60 sekunders intervaller. Dette bekrefter vellykket data-overføring via 4G-forbindelse (3.3) og en stabil datastrøm (3.4). Verdiene som vises i sanntid samsvarer med sensorenes faktiske målinger (3.2). Dette bekrefter at systemet fungerer som tilsiktet, og det konkluderes med at test nr. 3.2, 3.3 og 3.4 **oppfyller** systemkravene.



Figur 37: Skjermbilde av nettsiden som viser kontinuerlig mottak av TDS-data fra tre distinkte dybder.

For å verifisere systemets robusthet mot nettverksutfall (krav 3.5), ble 4G-forbindelsen manuelt brutt mens systemet var i drift. Under utfallstiden ble det simulert deteksjoner av fisk. Ved gjenoppretting av forbindelsen ble det observert at akkumulerte data, inkludert arts-ID og bildeutklipp, ble lastet opp til nettsiden med korrekte tidsstempler. Dette bekrefter at systemet håndterer frakobling ved å bufre data lokalt på Raspberry Pi-enheten, og systemkrav 3.5 anses derfor som **oppnådd**.

Test av bildedeteksjon

Ettersom systemet ikke ble plassert i vann, har bildedeteksjonen blitt validert ved bruk av alternative metoder. Den rene programvaredelen ble testet ved hjelp av videoklipp fra eksisterende systemer under tilsvarende forhold. For å sikre et pålitelig resultat, ble det utelukkende brukt videoer som ikke inngikk i modellenes treningsdata. I tillegg benyttet vi to fysiske fiskemodeller av ulik art og lengde for å kunne utføre en fullstendig systemtest av både maskinvare og programvare på land.

Fiskedeteksjon

For å teste programvarens evne til å detektere fisk, ble test 4.1 utført. Ved bruk av 50 testklipp som til sammen inneholdt 120 fiskepasseringer, klarte systemet å detektere 120 fisker (100% nøyaktighet). Dette svært gode resultatet skyldes systemets sporingsfunksjon. Algoritmen følger fisken gjennom hele videoforløpet, noe som gjør at den fanges opp selv om den skulle være uskarp eller vanskelig å detektere på enkeltbilder i løpet av passeringen. Videre testing vil være grensetilfelle-testing, som testing hvor deler av kameratene er tildekket eller under grumsete vann og lite lys. Disse testene vil finne hvor godt systemet vil funke under dårlige forhold og redusert nøyaktighet. Testing hvor stim av fisk svømmer forbi kreves også for å teste systemets nøyaktighet i slike senarioer. For disse videre testene kreves et større datasett.

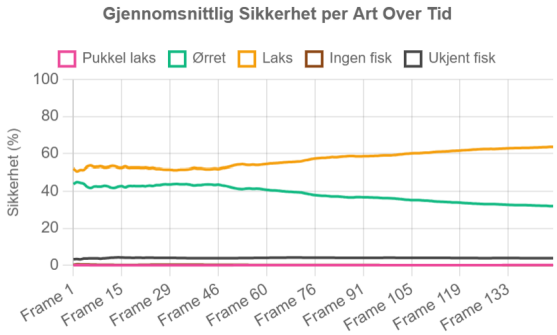
I tillegg ble det utført en rekke manuelle tester ved bruk av de fysiske fiskemodellene. Disse testene bekreftet resultatene fra test 4.1. Dette tilsier at systemkrav 4.1 er **oppnådd**.

Arts- og lengdedeteksjon

Testingen av DINOv2-modellen ble utført etter samme prinsipp. Først ble test 4.2 (klassifisering) gjennomført ved å analysere valideringssettet. På isolerte bildeutklipp oppnådde modellen en nøyaktighet på 90%. Gitt at systemet klassifiserer mellom fem ulike klasser, og at flere av utsnittene var uskarpe eller av lav bildekvalitet, anses dette som et meget lovende resultat. Når hele systemet ble tatt i bruk, og alle bildeutklipp av en og samme fisk ble aggregert for artsbestemmelse, klassifiserte systemet 118 av 120 fisker riktig (98% nøyaktighet). Valideringssettet inneholder representanter for alle de fem artene. Under ser du et eksmpel på en ørret som svømmer forbi som vår modell tilsier at er en laks.

Figur 38: Fisk klassifisert som ørret av ekspert

Vår modell gir på fisken som svømmer over en 32% sannsynlighet for ørret, og dobbelt så sannsynlig med 64% for laks. Under kan du se en graf over fiskens art sannsynlighet, hvor den frøst er usikker over fiskens art, mellom ørret og laks, før den feilklassifiserer arten til slutt som laks.

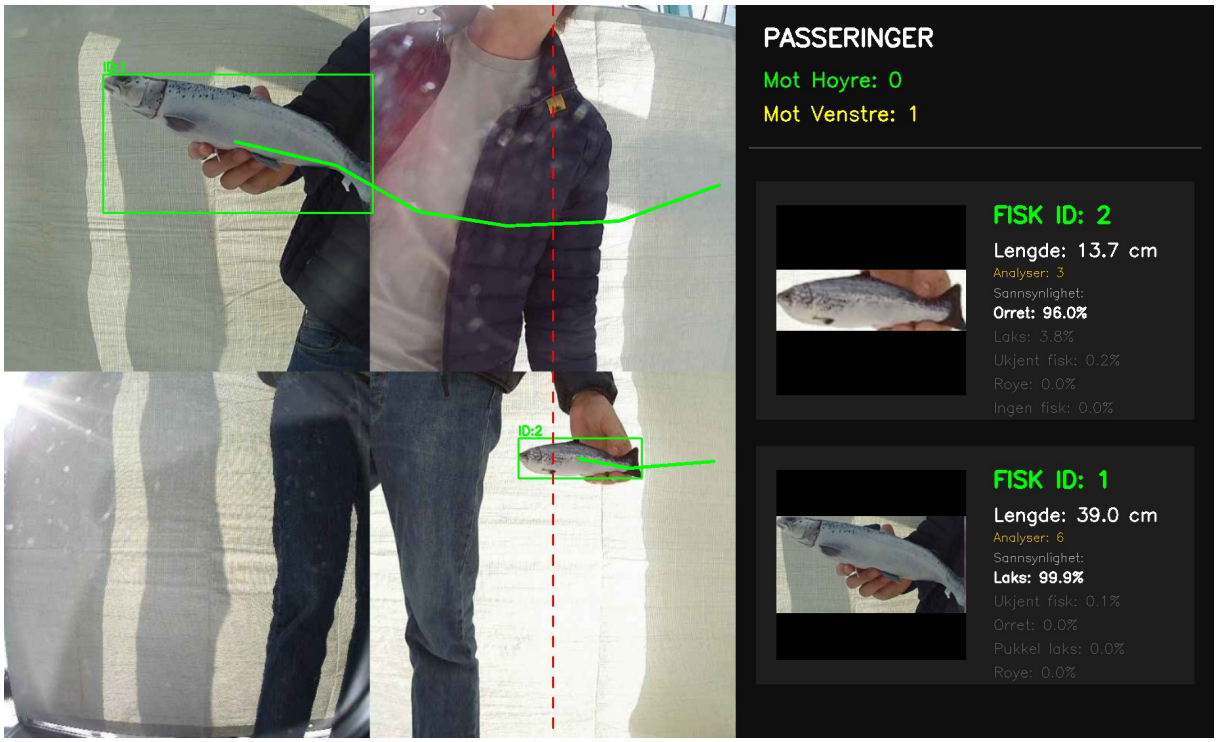


Figur 39: Graf over sannsynlighet til fiskeart for feilklassifisert fisk

Nøyaktigheten på 90% på bildeutklipp tilsier at systemkrav 4.2 akkuratt er **oppnådd**.

For lengdeestimering oppnådde modellen en gjennomsnittlig absolutt feil på 4,6cm på valideringssettet, ettersom gjennomsnitt lengden på dette datasettet er rundt 42cm har vi en nøyaktighet på rundt ±9%, disse resultatene er gitt singulerte utklipp og ikke samling av flere utklipp for hver fisk.

Lengdeestimeringen ble også testet på de fysiske fiskemodellene, i henhold til test 4.3. For å forenkle avlesingen av data under de fysiske testene, ble det utviklet en egen visualisering. Et eksempel på dette er vist i Figur 40.



Figur 40: Full test av system ved bruk av fysiske fiskemodeller

Visualiseringen viser fiskens svømmebane (grønn linje), telinepunktet for passering (rød stiplet linje), samt utklippene som sendes til arts- og lengdedeteksjon (til høyre i bildet). I tillegg vises sannsynlighetsfordelingen for de ulike artene knyttet til den spesifikke sporings-ID-en. Siden systemet benytter samtlige utklipp av en gitt fisk for å forbedre nøyaktigheten, vises også det totale antallet registrerte utklipp for hver fisk (markert i gult under ID-en). Systemkrav 4.3 er **oppnådd**.

Test av Grensesnitt

Grensesnittet ble verifisert ved å utføre planlagte fiskepasseringer og sjekke dashboardet i sanntid. Resultatene viste en forsinkelse på ca. 0,7 sekunder fra deteksjon til bildet var synlig på web.

Test 5.2 ble utført ved at 5 personer gikk inn på nettsiden og vurderte grafene. Tilbakemelding var at grafene er oversiktlige og lette å lese. Dybdeprofilene visualiseres grafisk og korrekt, som nevnt i underseksjonen «Test av databehandling».

Test 5.3 ble utført ved å laste ned CSV-filene. CSV-eksporten for sensordata ble testet og bekreftet å inneholde korrekt formaterte kolonner for tidsstempel, temperatur og TDS-verdier. CSV-eksporten for deteksjoner inneholder korrekt formaterte kolonner for tidsstempel, og fiske-id. En av kolonnene inneholder både sannsynlighetsdata og fiskens størrelse. For å gjøre fila mer oversiktlig og mer nyttig for videre bruk, burde denne blitt sortert slik at størrelsen på fisken havnet i en egen kolonne. Kravene er **oppnådd**.

Test av Energihåndtering

Test 1.1 og 1.2 er ikke utført siden energihåndtering er irrelevant for implementeringen. Inklusjonen av batteri og solcelle er nedprioritert i både design og implementering, og kravene er derfor **ikke oppnådd**.

Test av Utforming (bør)

Test 2.1 og 2.2 ble utført ved fysisk kontrollmåling av rammekonstruksjonen. Resultatene viste en bredde, lengde og høyde på henholdsvis 100 cm, 200 cm og 200 cm. Det bemerkes at lengden og høyden er justert til nedre del av kravintervallet for å optimalisere lagringsmuligheter og redusere materialkostnader. Rammekonstruksjonen er utført i aluminiumsprofiler, noe som gir en god kombinasjon av lav vekt og høy korrosjonsbestandighet. Systemkrav 2.1 og 2.2 er derfor **oppnådd**.

Ettersom systemet i denne fasen ikke er testet i vann, er test 2.3 og 2.4 ikke gjennomført; systemkrav 2.3 og 2.4 er følgelig **ikke oppnådd**. Av samme årsak ble det valgt å utsette fullstendig vanntetting av kapslingen, slik at systemkrav 2.5 heller **ikke er oppnådd** per dags dato

Tabell 9: Statusoversikt for systemets verifisering (Bør- og Skal-krav).

Bør-krav (ferdig produkt)		**Skal-krav (MVP)**	
Nr.	**Status**	**Nr.**	**Status**
1.1	Ikke Oppnådd	3.1	Delvis Oppnådd
1.2	Ikke Oppnådd	3.2	Oppnådd
2.1	Oppnådd	3.3	Oppnådd
2.2	Oppnådd	3.4	Oppnådd
2.3	Ikke Oppnådd	3.5	Oppnådd
2.4	Ikke Oppnådd	4.1	Oppnådd
2.5	Ikke Oppnådd	4.2	Oppnådd
		4.3	Oppnådd
		5.1	Oppnådd
		5.2	Oppnådd
		5.3	Oppnådd

5.2 Validering av brukerkrav

5.2.1 Valideringsmetode og resultater

For å validere konseptet uten fysisk utsettelse i vann, er det benyttet evaluering av 10 eksperter og interessepersoner (biologer, forskere, ...) samt brukertesting av systemets programvare og grensesnitt. Planen for validering tar sikte på å bekrefte at systemet gir reell verdi for sluttbrukere i forvaltningen. Valideringen gjennomføres ved at testpersoner får demonstrert systemets funksjonalitet gjennom GUI og historiske datasett, etterfulgt av et spørreskjema der hvert punkt validerer de overordnede brukerkravene definert i Tabell 1:

- **Brukerkrav 1.1–1.4 (Identifisering):** På en skala fra 1 til 5, i hvilken grad gir systemets evne til å telle og skille mellom laks, pukkellaks, ørret og sjørøye?

- **Brukerkrav 1.5–1.7 (Miljødata):** På en skala fra 1 til 5, i hvilken grad vurderes systemets evne til innhenting av temperatur, salinitet og havstrøm?
- **Brukerkrav 1.8–1.9 (Datatilgjengelighet):** På en skala fra 1 til 5, hvor tilfredsstillende er løsningen for lagring og tilgang til historisk data for ettertidens analyser?
- **Brukerkrav 2.1–2.2 (Grensesnitt):** På en skala fra 1 til 5, i hvilken grad er presentasjonen av sanntidsdata forenlig med eksisterende arbeidsflyt og behov for oversikt?
- **Brukerkrav 3.1–3.3 (Utforming og Implementering):** På en skala fra 1 til 5, til hvilken grad vurderes systemet som enkelt å implementere og vedlikeholde?
- **Brukerkrav 4.1 (Fiskevelferd):** På en skala fra 1 til 5, i hvilken grad gir konseptet (fysisk utforming og metodikk) tiltro til at den norske laksefisken ivaretas på en forsvarlig måte?

Tabell 10 viser gjennomsnittsresultatene fra denne evalueringen oppsummert.

Tabell 10: Resultater fra validering av brukerkrav (gjennomsnittsscore).

Nr.	Brukerkrav i fokus	Resultat (1–5)
1.1–1.4	Automatisk identifisering og telling av arter	4.5
1.5–1.7	Innhenting av miljødata (temp, salinitet, strøm)	3.8
1.8–1.9	Lagring og tilgjengelighet av historisk data	4.8
2.1–2.2	Brukergrensesnitt og sanntidsoppdatering	4.7
3.1–3.3	Vedlikehold, kostnad og implementering	3.2
4.1	Ivaretakelse av fiskevelferd	4.7

5.2.2 Drøfting av valideringsresultater

Resultatene fra valideringen viser en gjennomgående høy score på systemets kjernefunksjonalitet, men avdekker også forbedringspotensial knyttet til fysisk implementering og sensornøyaktighet.

- **Identifisering og telling (1.1–1.4):** Med en score på 4,5 bekreftes det at maskinsynsmodellen utfører identifiseringen med høy treffsikkerhet. Valideringen avdekket imidlertid at modellen i enkelte grensetilfeller kan utvise usikkerhet eller feilklassifisering. Dette skyldes trolig variabler i videomaterialet, og tyder på at ytterligere trening på mer varierte datasett vil være nødvendig for å oppnå fullstendig pålitelighet i felt.
- **Miljødata (1.5–1.7):** Scoren på 3,8 reflekterer tekniske begrensninger i den nåværende prototypen. Tilbakemeldingene indikerte at innsamling fra tre dybdepunkter (én måling per 50 cm) gir begrenset vertikal oppløsning i vannsøylen, og dermed ikke er tilstrekkelig for å generere en fullverdig dybdeprofil. En ønsket oppløsning ville vært omtrent én måling per 20 cm, noe som ville gitt et mer detaljert bilde av variasjoner i temperatur og salinitet mellom ulike vannlag. Videre ble det påpekt at bruk av TDS-sensor som indirekte mål på salinitet introduserer usikkerhet i målingene, ettersom metoden er mer følsom for støy og variasjoner i ledningsevne enn dedikerte salinitetssensorer.
- **Datatilgjengelighet og grensesnitt (1.8–2.2):** Disse punktene oppnådde de høyeste scorene (4,7 og 4,8). Dette begrunnes med at løsningen for lagring og visualisering på

nettsiden er svært intuitiv. Ekspertene trakk frem at den grafiske fremstillingen av korrelasjonen mellom fiskepasseringer og miljødata over tid gir en oversikt som er direkte forenlig med moderne forvaltningsbehov.

- **Implementering og vedlikehold (3.1–3.3):** Den laveste scoren (3,2) er knyttet til systemets drift. Siden prototypen i denne fasen ikke opererer på batteri eller solcellepaneler, vurderes implementering i avsidesliggende sund som utfordrende. For å forbedre dette resultatet, er overgang til egen energiforsyning et kritisk punkt.
- **Fiskevelferd (4.1):** En score på 4,7 viser stor tiltro til konseptets utforming for veldferd. Ved å benytte en romslig rammekonstruksjon sikres det at fisken har god plass ved passering, noe som minimerer stress og fysisk påvirkning.

Samlet sett validerer resultatene at konseptet løser de viktigste brukerbehovene knyttet til overvåking og datainnsamling, forutsatt at energiforsyning og sensorpresisjon videreutvikles i neste iterasjon.

6 Videreutvikling

Under utviklingen ble databehandling, deteksjon og brukergrensesnitt prioritert. Dermed gjenstår det flere muligheter for videreutvikling, særlig innen energihåndtering og fysisk utforming. De viktigste videreutviklingene presenteres i dette kapittelet.

6.1 Energihåndtering

En viktig videreutvikling for å oppnå autonom drift er å oppnå systemkrav 1.1 og 1.2 om batteridrift og solcelle-lading slik som det originale konseptet la frem i Figur 2. Under utviklingen er det allerede tatt hensyn til energibruk ved å ta i bruk en energieffektiv Raspberry Pi til prosessering, og å ta i bruk optimaliserte algoritmer for lokal bildegjenkjenning for å spare prosesseringskaft, og dermed oppnå lavere effekt.

På batteri siden er systemet allerede lagt opp til å direkte kunne kobles på en DC-spenning mellom 5,2-40V. Derav den ønskede spenningen vi så for oss ligger rundt $33,3\,\mathrm{V}$ ettersom det var dette vi har tilgengelig fra andre prosjekter. En høy spenning tillater oss også å kjøre led driverene direkte fra denne spenningen og dermed kjøre flere LED i serie.

Å kjøre LED-striper med led i serie tillater lett konstruksjon av led striper med spesifiserte LED-er for norsk farvann.

Med introduksjon av solcellepanel på toppen av systemet vil kjøretiden bli forlenget i stor grad, trolig i den grad at brukerkrav 3.1 vil være overholdt (systemet skal kreve lite vedlikehold).

6.2 Konstruksjon

Mer utvikling i konstruksjonen kreves for å kunne levere et produktet som er klar til bruk. I design og implementering er det valgt å nedprioritere krav om et vanntett system. Ved videre utvikling må alle kontakter/ledninger være gjort saltvann sikre. Kamerainnkapslingene må også være testet og verifisert for gjennomtrengning av vann ved dybden de er plassert. Videre må innkapslingen som rommer mikrokontrollerne være utviklet ytterligere for å tåle vannsprut. Dette innebærer å konstruere den av et annet materiale enn treverk da dette ikke egner seg i miljøet det er tenkt. Kapslingen er imidlertid plassert taktisk på toppen av systemet og må derfor ikke tåle full nedsenkning i vannet.

Et ytterligere valg som er tatt under design/implementeringsprosessen er å ikke bruke tid på flyteegenskapen til systemet. Flyteelementer som kan være relevante er standard bøyer/fendere. Disse kan trolig festes på systemet slik det er konstruert nå uten ytterligere modifiseringer.

Der konstruksjonen mangler mest utviklingen er ledenettet for å lede fisken inn i systemet slik som det er lagt frem i konseptet i Figur 2. Her er det ingen åpenbar løsning, da dette trenger ytterligere utvikling og testing for å kunne fungere i en reell installasjon. Hensyn å ta her er at fisken ikke skal bli påvirket i stor grad av ledenettet. Dersom fisken reagerer på å bli ført et sted den ikke vil, og derfor snur, vil dette påvirke hvor representativ dataen som er hentet inn er for bestanden av laksefisk i området. Dette er et punkt hvor det vil kreves ekspertise innenfor fiskeoppførsel. Videre ved et ferdig produkt kunne det vært nødvendig for å øke arealet på sideveggene til observasjonsystemet. Dette hadde medført at systemet ville detektert flere laksefisk som svømmer lenger unna brakkvann enn det som klarer nå. I tillegg ville sikkerheten av art på fisen som svømmer gjennom økes med en lengre detekteringstid.

6.3 Bildegjenkjenning

Den største flaskehalsen for systemets ytelse ved lokal deteksjon er prosesseringskaft. Kameraene som er montert støtter opptil 4K oppløsning men med kjøring på rasberry pi sin CPU er det nødvendig å skalere ned bildene til 960p oppløsning. Mosaikk løsningen som er valgt er på grunna av hensyn til prosesseringskaft, og hastighet under kjøring, men vil også ytterligere redusere oppløsningen til hver enkelt fisk i bildet. En videreutvikling som enkelt hadde økt ytelsen til den initielle deteksjonen er å kjøre modellen på dedikert ML-hardware, eksempelvis en Nvidia Jetson Nano Super modul. Mer prosesseringskaft muliggjør bruk bilder av høyere oppløsning. I mindre ideelle forhold, slik som ved mye grums i vannet, eller lavt lys vil dette være viktig for god ytelse. Mer prosesseringskaft kan også kjøre en større modell med flere parametre, noe som med mer trenings-data kan oppnå bedre generalisering. Kostnaden og energibehovet vil derimot øke i takt med økt prosesseringskaft, noe som vil være en viktig avveining i et ferdig produkt.

Ytelsen til artsgjenkjenning er begrenset av datasettet som er tilgjengelig. Ved videre utvikling er den viktigste forbedringen å få mer jevnt fordelt data. Slik det er nå er villaksen overrepresentert i stor grad, noe som trolig påvirker ytelsen.

Dersom det ved videre utvikling sees nødvendig å oppnå bedre ytelse for lengdeestimat finnes det alternative løsninger som kan være nødvendig å implementere. Installering av et stereokamera har potensiale til å forbedre lengdeestimatet betydelig.

6.4 Sensorikk

Protypen har 3 dybdeplasseringer for et MVP. I et ferdigutviklet produkt er det nødvendig med fler sensorplasseringer. Type sensor er i prototypen valgt med hensyn til kostnad, med utvidet budsjett er det mulig å ta i bruk ordentlige salinitetssensorer, bedre kalibrerte temperatursensorer. Det er også mulig å implementere ytterligere funksjonalitet slik som måling av havstrøm og tidevann, dersom dette vil være relevant for en forsker/forvalter.

7 Konklusjon og anbefalinger

En funksjonell prototype for automatisert overvåking av fiskebestander er utviklet og teknisk verifisert. Løsningen integrerer kontinuerlig miljøovervåking med automatisk deteksjon og analyse av fiskepasseringer. Ved å korrelere temperatur- og salinitetsdata fra flere dybdenivåer med artsbestemmelse av fire ulike fiskearter, legger systemet grunnlaget for å analysere sammenhenger mellom miljøforhold og vandringsmønstre.

Prototypen tilfredsstiller de definerte brukerkravene knyttet til funksjon, grensesnitt, fysisk utforming og fiskevelferd. Evalueringen viser at 10 av 11 **skal-krav** for en MVP er fullt ut oppfylt, mens det gjenværende kravet er delvis oppfylt. Av de syv definerte **bør-kravene** er 2 av 7 oppfylt i nåværende versjon. Resultatene bekrefter at prototypen fungerer som en vellykket demonstrasjon av konseptet, selv om systemet krever videreutvikling før det kan installeres som et autonomt produkt i felt.

Videre arbeid bør fokusere på å klargjøre systemet for drift i krevende omgivelser. Dette inkluderer full integrasjon av planlagt tilleggsfunksjonalitet som solcellepanel, batteribank, flyteelementer og ledesystem. I tillegg anbefales det å øke systemets prosesseringskraft og oppgradere sensorene for å forbedre både datakvalitet og presisjon i miljømålingene.

8 Arbeidsfordeling

Samtlige gruppemedlemmer har bidratt aktivt til planlegging, strukturering, skriving og korrekturlesing av den avsluttende rapporten. Utover selve rapportskrivingen har prosjektet vært inndelt med følgende hovedansvarsområder for å sikre god fremdrift:

Andreas Heyerdahl Hovedansvar for prosjektets sensorsystem. Dette innebærer utvelgelse, kalibrering og implementering av de fysiske sensorene, som for eksempel TDS-sensoren. Han har arbeidet med å sikre stabil datainnsamling og overføring av sensordata til mikrokontrolleren. I tillegg har han gjennomført omfattende testing for å kartlegge sensorenes nøyaktighet og håndtere utfordringer knyttet til for eksempel støy og feilmarginer i vann.

Andreas Lindeman Delt hovedansvar for den fysiske maskinvaren og utviklingen av artsgjenkjenningsmodellen (DINOv2). På maskinvaresiden har han arbeidet med det fysiske oppsettet av kamerasystemet, sammenkobling av komponenter og strømstyring. Parallelt har han hatt ansvar for å tilpasse og validere klassifiseringsmodellen for å sikre høy nøyaktighet ved artsbestemmelse, samt integrert denne funksjonaliteten mot systemets videostrøm.

Athena McQueen Hovedansvar for utformingen og utviklingen av systemets nettside, med et særskilt fokus på frontend-arkitektur og brukeropplevelse (UX/UI). Arbeidet hennes har bestått i å bygge et intuitivt og responsivt dashboard som visualiserer innsamlet data. Hun har implementert løsninger for å presentere sanntidsdata, grafer over sensormålinger, historikk over fiskepasseringer og bildeutklipp på en oversiktlig og brukervennlig måte.

Christin Bertelsen Hovedansvar for nettsidens backend-infrastruktur og databasehåndtering. Hennes oppgave har vært å etablere en sikker og effektiv dataflyt mellom maskinlæringssystemet, databasen (SQL) og nettsidens frontend. Dette inkluderer utvikling av API-er for å ta imot sensordata og deteksjonsresultater (som fiskeart og lengde), samt strukturering av databasen for å sikre rask og pålitelig lagring av historikk og overvåkningsdata.

Martin Steinsvoll Rikardsen Hovedansvar for det overordnede maskinlæringssystemet og bildebehandlingsarkitekturen. Dette inkluderer utvikling, trening og optimalisering av deteksjonsmodellen for lokal gjenkjenning av fisk og sporing (tracking) av deres svømmebane. Han har også arbeidet i dybden med logikken og matematikken bak estimering av fiskens lengde, og sørget for at hele system-pipelinen fungerer sømløst fra rått videoopptak til ferdig prosesserte data klare for backenden.

Referanser

[1] Miljødirektoratet. «Laks.» Miljødirektoratet.no. Hentet fra: https://www.miljodirektoratet.no/ansvarsomrader/arter-naturtyper/fiske/laks/ (Lastet ned: 13.04.2026).

[2] Havforskningsinstituttet. «Pukkellaks.» Hi.no. Hentet fra: https://www.hi.no/hi/temasider/arter/pukkellaks (Lastet ned: 15.04.2026).

[3] Norsk Institutt for Naturforskning. «Pukkellaks.» Nina.no. Hentet fra: https://www.nina.no/pukkellaks (Lastet ned: 30.04.2026).

[4] Mohn Technology. «Mohn Technology.» Mohntechnology.no. Hentet fra: https://www.mohntechnology.no (Lastet ned: 20.04.2026).

[5] ScaleAQ. «Kamerasystem.» Scaleaq.no. Hentet fra: https://scaleaq.no/produktkategorier/foring-og-overvaking/kamerasystemer/ (Lastet ned: 20.04.2026).

[6] NORCE Research. «LFI - Laboratorium for ferskvannsøkologi og innlandsfiske.» Norceresearch.no. Hentet fra: https://www.norceresearch.no/forskergruppe/lfi-laboratorium-for-ferskvannsokologi-og-innlandsfiske (Lastet ned: 20.04.2026).

[7] Troll Systems. «Troll Systems.» Trollsystems.no. Hentet fra: https://www.trollsystems.no (Lastet ned: 20.04.2026).

[8] Huawei. «Saving Norway's Endangered Atlantic Salmon.» Huawei.com. Hentet fra: https://www.huawei.com/en/tech4all/stories/saving-the-atlantic-salmon-in-norway (Lastet ned: 23.04.2026).

[9] X. Zhai, A. Kolesnikov, N. Houlsby, og L. Beyer, «Scaling Vision Transformers,» arXiv:2106.04560, 2021.

[10] Ultralytics. «Ultralytics YOLO26.» Docs.ultralytics.com. Hentet fra: https://docs.ultralytics.com/models/yolo26/ (Lastet ned: 23.03.2026).

[11] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, og P. Dollár, «Microsoft COCO: Common Objects in Context,» arXiv preprint arXiv:1405.0312, 2015.

[12] IBM. «What is overfitting?» ibm.com. Hentet fra: https://www.ibm.com/think/topics/overfitting (Lastet ned: 29.04.2026).

[13] Google. «Classification: Accuracy, recall, precision, and related metrics.» developers.google.com. Hentet fra: https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall (Lastet ned: 29.04.2026).

[14] N. Aharon, R. Orfaig, og B.-Z. Bobrovsky. «BoT-SORT.» Github.com. Hentet fra: https://github.com/NirAharon/BoT-SORT (Lastet ned: 26.03.2026).

[15] Meta. «Meta AI, DINOv2.» Metademolab.com. Hentet fra: https://dinov2.metademolab.com/ (Lastet ned: 22.03.2026).

[16] I. Berrios. «Multi-task Architectures.» Medium.com. Hentet fra: https://medium.com/data-science/multi-task-architectures-9bee2e080456 (Lastet ned: 22.03.2026).

[17] E. J. Hu, Y. Sharma, R. Lukanov, N. Holzenberger, G. Ke, D. Hazarika, J.-Y. Wu, L.-W. Chen, B. Ahmed, og J. Gao, «LoRA: Low-Rank Adaptation of Large Language Models,» arXiv preprint arXiv:2106.09685, 2021.

[18] T.-Y. Lin, P. Goyal, R. Girshick, K. He, og P. Dollár, «Focal Loss for Dense Object Detection,» arXiv preprint arXiv:1708.02002, 2017.

[19] PyTorch. «HuberLoss.» Docs.pytorch.org. Hentet fra: `https://docs.pytorch.org/docs/stable/generated/torch.nn.modules.loss.HuberLoss.html` (Lastet ned: 23.03.2026).

[20] DFRobot. «DS18B20 Water Proof Temperature Probe, DFR0198 Datasheet.» Farnell.com. Hentet fra: `https://www.farnell.com/datasheets/4376672.pdf` (Lastet ned: 07.05.2026).

[21] DFRobot. «Gravity: Analog TDS Sensor / Meter for Arduino, DFR0300 Datasheet.» Farnell.com. Hentet fra: `https://www.farnell.com/datasheets/4376692.pdf` (Lastet ned: 07.05.2026).

[22] Texas Instruments. «A Basic Guide to I2C.» TI.com. Hentet fra: `https://www.ti.com/lit/an/sbaa565/sbaa565.pdf` (Lastet ned: 14.04.2026).

[23] Wikipedia. «Flask (web framework).» Wikipedia.org. Hentet fra: `https://en.wikipedia.org/w/index.php?title=Flask_(web_framework)&oldid=1331301896` (Lastet ned: 30.04.2026).

[24] Wikipedia. «SQLAlchemy.» Wikipedia.org. Hentet fra: `https://en.wikipedia.org/w/index.php?title=SQLAlchemy&oldid=1294724957` (Lastet ned: 30.04.2026).

[25] Tech With Tim. *Flask Tutorial #1 - How to Make Websites with Python.* (27.10.2019). Sett: 12.03.2026. [Online video]. Hentet fra: `https://www.youtube.com/watch?v=mqhxxeeTbu0`

[26] Tech With Tim. *Flask Tutorial #2 - HTML Templates.* (28.10.2019). Sett: 12.03.2026. [Online video]. Hentet fra: `https://www.youtube.com/watch?v=xIgPMguqyws`

[27] Tech With Tim. *Flask Tutorial #3 - Adding Bootstrap and Template Inheritance.* (30.10.2019). Sett: 12.03.2026. [Online video]. Hentet fra: `https://www.youtube.com/watch?v=4nzI4RKwb5I&list=PLzMcBGfZo4-n4vJJybUVV3Un_NFS5EOgX&index=3`

[28] Tech With Tim. *Flask Tutorial #4 - HTTP Methods (GET/POST) & Retrieving Form Data.* (2.11.2019). Sett: 12.03.2026. [Online video]. Hentet fra: `https://www.youtube.com/watch?v=9MHYHgh4jYc`

[29] Tech With Tim. *Flask Tutorial #9 - Static Files (Custom CSS, Images & Javascript).* (18.11.2019). Sett: 12.03.2026. [Online video]. Hentet fra: `https://www.youtube.com/watch?v=tXpFERibRaU`

[30] Tech With Tim. *Flask Tutorial #10 - Blueprints & Using Multiple Python Files.* (19.11.2019). Sett: 12.03.2026. [Online video]. Hentet fra: `https://www.youtube.com/watch?v=WteIH6J9v64`

[31] Bring Your Own Laptop. *Free Course: Beginner Web Design using HTML5, CSS3 & Visual Studio Code - YouTube.* (29.06.2019). Sett: 12.03.2026. [Online video]. Hentet fra: `https://www.youtube.com/watch?v=C5QFHp1oAws&list=PLwOL3j6-8a7sG7bcQHWGaPFtLkkLNZyS4&index=14`

[32] Bootstrap. «Navbar.» Getbootstrap.com. Hentet fra: `https://getbootstrap.com/docs/5.3/components/navbar/` (Lastet ned: 12.03.2026).

[33] Maskarapriyanshu. «A Beginner's Guide to SQLAlchemy: Getting Started with Python's Powerful ORM.» Medium.com. Hentet fra: https://medium.com/@maskarapriyanshu/a-beginners-guide-to-sqlalchemy-getting-started-with-python-s-powerful-orm-ab8428f7074b (Lastet ned: 25.03.2026).

[34] J. Fernando. «Moving Average (MA): Purpose, Uses, Formula, and Examples.» Investopedia.com. Hentet fra: https://www.investopedia.com/terms/m/movingaverage.asp (Lastet ned: 19.03.2026).

[35] Biltema. «Grilltermometer, 300 °C.» Biltema.no. Hentet fra: https://www.biltema.no/fritid/grill/grilltilbehor/grilltermometer-300-c-2000041065 (Lastet ned: 07.05.2026).

[36] Fruugo. «TDS-3 Digital Water Quality Tester.» Fruugonorge.com. Hentet fra: https://www.fruugonorge.com/tds-3-digital-water-quality-tester/p-474182919-990374272 (Lastet ned: 07.05.2026).

A Vedlegg: Kildekode (GitHub)

For å sikre full innsyn i systemets programvare og tekniske utførelse, er all kildekode tilgjengelig via organisasjonen **Elsys-Gruppe-1** på GitHub: `https://github.com/Elsys-Gruppe-1`. Koden er fordelt på tre repositories som ivaretar ulike deler av systemarkitekturen.

1. Sensor Side Dette lagringsstedet inneholder programvaren som kjører lokalt på systemets fysiske maskinvare. Dette inkluderer lavnivå-kode for ESP32 som håndterer sensorinnhenting, I2C-kommunikasjon mellom enhetene, samt den lokale YOLO-implementeringen som sørger for sanntidsdeteksjon og uthenting av bildeutsnitt på Raspberry Pi.

2. Server Side Dette repoet omfatter systemets sentrale infrastruktur for datahåndtering og presentasjon. Det består av to hovedkomponenter: en webserver som drifter brukergrensesnittet (GUI), og en prosesseringsklient som fungerer som bindeleddet mellom sensorenhetene i felt og databasen i skyen.

3. Remote Processing Her ligger systemets tyngste logikk for bildeanalyse og maskinlæring. Lagringsstedet inneholder en avansert pipelinesom mottar data fra sensorenhetene og utfører nøyaktig artsklassifisering ved bruk av DINOv2-transformere. Repoet dekker hele løpet fra sporing av fiske-ID til smart mellomlagring av prosesserte videofiler for optimal ressursbruk.

Ai verktøy er benyttet for rettskriving og feilsøking av koding.