Høgskulen på Vestlandet

Bacheloroppgave

ELE350

Predefinert informasjon

Startdato:	08-05-2023 09:00 CEST	**Termin:**	2023 VÅR
Sluttdato:	22-05-2023 14:00 CEST	**Vurderingsform:**	Norsk 6-trinns skala (A-F)
Eksamensform:	Bacheloroppgave		
Flowkode:	203 ELE350 1 O 2023 VÅR		
Intern sensor:	Torkel Bjarte Larsson		

Deltaker

Navn:	Jari Miicael Andreas Inermo
Kandidatnr.:	243
HVL-id:	590390@hvl.no

Informasjon fra deltaker

Egenerklæring *:	Ja
Inneholder besvarelsen konfidensielt materiale?:	Nei
Jeg bekrefter at jeg har registrert oppgavetittelen på norsk og engelsk i StudentWeb og vet at denne vil stå på vitnemålet mitt *:	Ja

Gruppe

Gruppenavn:	BO23EH-04 HVL Autodrone 2023
Gruppenummer:	45
Andre medlemmer i gruppen:	Njål Almenning

Jeg godkjenner avtalen om publisering av bacheloroppgaven min *

Ja

Er bacheloroppgaven skrevet som del av et større forskningsprosjekt ved HVL? *

Nei

Høgskulen på Vestlandet

BACHELOROPPGAVE

BO23EH-04 HVL Autodrone 2023

Jari Miicael Andreas Inermo 243

Njål Almenning 320

Bachelor Automatisering med robotikk

Høgskulen på Vestlandet, Campus Haugesund

22/05-2023

Jeg bekrefter at arbeidet er selvstendig utarbeidet, og at referanser/kildehenvisninger til alle

kilder som er brukt i arbeidet er oppgitt, *jf. Forskrift om studium og eksamen ved Høgskulen på Vestlandet, § 12-1.*

1 Forord

Denne Autodrone bacheloroppgaven ble gitt til oss av HVL (Høgskulen på Vestlandet). Vi hadde selv sett noen andre skrive en bacheloroppgave på autodronen våren 2022, og vi kunne se autodronen i gangen utenfor elektrolabben. På høsten 2022 når vi var på søk etter bacheloroppgave fikk vi beskjed om at båten var ledig, og vi søkte dermed om å få ta denne bacheloroppgaven. Vi hadde et lite informasjonsmøte med Torkel Bjarte Larsson og Harald Spångberg, der vi ble informert om hva bacheloroppgaven gikk ut på. Vi fikk tilbud om å ta oppgaven, og vi takket ja. Oppgaven går ut på å fortsette arbeidet som den forrige gruppen hadde påbegynt våren 2022, og å forbedre autodronen ytterligere. Etter bacheloroppgaven er målet å delta på AutoDrone 2023 konkurransen, og representere HVL.

Vi skriver denne bacheloroppgaven i siste semester av en treårig bachelorgrad innen automatisering med robotikk på HVL campus Haugesund. Denne oppgaven var stor og med mange utfordringer, spesielt siden dronen hadde ligget lagret med fukt siden sommeren 2022, og det var korrupt data i mye av den digitale kopien fra den forrige gruppen. Det var og mye ny teknologi vi måtte lære oss, samt mye teori vi har satt oss inn i og prøvd å implementere. På Haugalandet har vi den siste tiden sett flere nye innovasjonsprosjekter som baserer seg på autonomi innen mange sektorer, og autonomi på havet er høyaktuelt som en bacheloroppgave.

Vi ønsker å takke alle som har hjulpet og støttet oss når vi har jobbet med denne oppgaven, vi vil også spesielt takke:

- Torkel Bjarte Larsson og Olav Skjølingstad, som var veilederne våre, og kom med mange gode innspill og tips.
- Linn Jeanette Myhren og Magnus Eide, som hadde oppgaven før oss, og som tok seg tid til å svare på spørsmål i starten.
- Høgskulen på Vestlandet som lot oss få velge autodronen som vår bacheloroppgave.
- Norwegian Test & Inspection og Origo Test & Inspection som sponset oss med 5’000,- kroner, til å dekke diverse utgifter vi har hatt i denne bacheloroppgaven.
- Jari Johannes Inermo, som lot oss få låne lokale til å jobbe med autodronen i, samt egen privat brygge for sjøsetting av dronen.

2 Sammendrag

Bacheloroppgaven går ut på å forbedre en HVL Haugesund sin autonome sjødrone, slik at den kan delta på en konkurranse i Horten månedsskifte Mai Juni. Denne oppgaven ble for første gang utført på HVL våren 2022, og vår oppgave er å videreføre deres arbeid, slik at vi kan få forbedret autodronen til de forskjellige konkurransegrenene. Konkurransen AutoDrone, er et samarbeid mellom HVL, NTNU og USN. Konkurransen holdes for tredje året i Norge, og gjennomføres i Horten. Konkurransen deles opp i fire grener.
Autodronen er en ubemannet båt, som er utstyrt med sensorer og trustere, den kan navigere både med fjernstyring via fjernkontroll, og med autonomi. Ved starten av oppgaven oppdaget vi at flere av komponentene hadde blitt ødelagt som følge av saltvann, noe det ble brukt en del tid på å rette opp i dette. Gjennom arbeidet med bacheloroppgaven har vi fokusert på tre hovedelementer: reparere dronen, forbedre objektdeteksjonen og forbedre lokaliseringen.
Vi hadde en tung start på oppgaven, da det var mye å sette seg inn i, og det forelå lite dokumentasjon om hvordan ting fungerte. Vi har derfor brukt tid på å dokumentere prosessene i denne bacheloroppgaven, slik at senere bachelorgrupper kan få en enklere start. Dette resulterte i en rekke prosedyrer som gjør det enklere å operere autodronen.
Under arbeidet med objektdeteksjonen oppdaget vi mangler i den forrige gruppens datasett. Det ble også opplyst om at de hadde problemer med å detektere ønskede objekter i realistisk setting. Derfor har vi laget et nytt og forbedret datasett til trening av objektdeteksjonen, noe som resulterte i en drastisk forbedring av deteksjon av bøyer på avstander opptil 20 meter.
I algoritmen for lokalisering av autodronen hadde den forrige bachelor gruppen kun basert seg på GPS-data for lokalisering av autodronen. Gjennom testing ble det klart at dette viste seg å være for unøyaktig. Dette forsøkte vi å forbedre ved å bruke *Sensor Fusion* av *GPS* og *IMU*-data for å estimere lokaliseringen av autodronen.

3 Summary

The bachelor's thesis is to improve an autonomous sea drone, so that it can participate in a competition at the turn of the month May & June. This task was carried out for the first time by HVL in spring 2022, and our task is to continue their work, so that we can improve the autodrone for the various competition branches. The AutoDrone competition is a collaboration between HVL, NTNU and USN. The competition is held for the third year in Norway and takes place at Horten. The competition is divided into four branches. The autodrone is an unmanned boat, which is equipped with sensors and thrusters, it can navigate both with remote control and by self-autonomy. When the task was started, the computer that handles the navigation was destroyed by salt water, it took some time to rectify this. We had a difficult start to the thesis, as there was a lot to get to grips with, and little documentation about how things worked, we have thus spent time documenting the processes in this bachelor's thesis, so that the next bachelor's group can have an easier start. The autonomy itself has been prepared by the previous bachelor group that had this autodrone, it was reported that it was not working as expected. Something that was discovered is that the previous group took a lot of pictures inside the university, at short distances of the buoys they were supposed to use, this caused that during the competition, they were not able to detect the buoys. We have used the autodrone a lot in the sea, to photograph buoys, so that we can improve the data set for object detection. During testing of the new weights for the object detection, we could conclude with a high increased improvement of the detection of the buoys up to 20 meters, as we lose some resolution of objects above 20 meters, which makes the detection somewhat more unreliable. In the algorithm for locating the autodrone, the previous bachelor's group had only relied on GPS readings for locating the autodrone, we have tested this out at sea, by driving to coordinates, but we see that an error margin increases over time. We thus put the autodrone on land, and made sure that it was completely stationary, and we could see that we got a margin of error of up to three meters over a minute, we have solved this by using Sensor Fusion to help reduce this rising margin of error. The Sensor Fusion method we have created uses GPS and IMU readings to estimate the location of the autodrone.

4 Ordliste

HVL = Høgskulen på Vestlandet

COLREG = Convention on the international regulations for preventing collisions at sea

ROS = Robot Operating System

GPS = Global Positioning System

IMU = Intertial Measurement Unit

LiDAR = Light Detection and Ranging

YOLO = You Only Look Once

USN = Universitetet I Sørøst-Norge

NTNU = Norges Teknisk-Naturvitenskaplige Universitet

HAT = Hardware Attached on Top

GPU = Graphic Processor Unit

CPU = Central Processing Unit

CEP = Circular Error Probable

5 Innholdsfortegnelse

6 Introduksjon

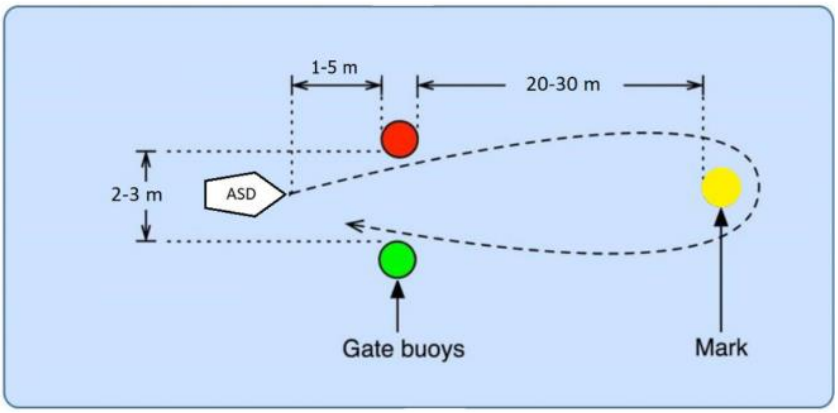
Bacheloroppgaven vår går ut på å fortsette arbeidet med HVL Haugesund sin autonome sjødrone. Målet med oppgaven er å forbedre sjødronen, og stille i Autodrone2023 som er en konkurranse som holdes i Horten fra 31.Mai til 1.Juni 2023. Vi vil videre gå igjennom hva selve konkurransen går ut på, samt gi en introduksjon av sjødronen og noen av de viktigste komponentene den innehar. I tillegg vil vi kort informere om ulike systemer og programvarer vi har brukt, slik at leseren får en bedre forståelse av arbeidet vårt under de ulike metode-kapitelene. Vi vil også bruke en del engelske ord og uttrykk ettersom det finnes mange fremmedord i dette emnet, og norske oversettelser ikke alltid godt nok representerer det vi vil beskrive i denne bacheloroppgaven. Disse fremmedordene vil vi skrive i kursiv tekst.

Figur 1: Autodronen under en sjøtest (Privat)

6.1 Autodrone 2023

Autodrone2023 er en konkurranse der ulike utdanningsinstitutter og profesjonelle aktører har mulighet til å delta i. Konkurransen går ut på å at en autonom sjødrone skal gjennomføre totalt fire ulike poenggivende øvelser. Øvelsene blir blant annet bedømt på tid, og evnen til å oppfylle den spesifikke øvelsens formål.
De fire ulike øvelsene er:

Speed Gate: Autodronen kjører gjennom en port, bestående av en grønn og en rød bøye, rundt en gul bøye og tilbake gjennom porten.

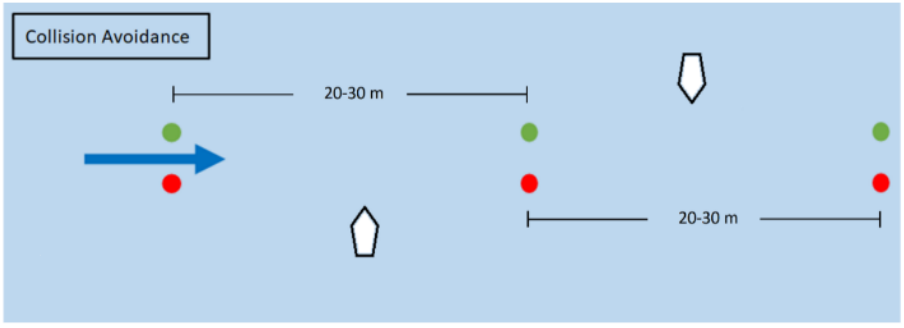


Figur 2: Illustrasjon av Speed Gate grenen [1]

Scoring	
Criteria	**Points**
Time criteria	80*1/place
Exit through gate	20
Round the mark	15
Reach Mark	10
Enter through gate	5

Figur 3: Poenggiving Speed Gate [1]

Collision Avoidance: Autodronen passerer gjennom porter, men blir møtt av trafikk, og må dermed kunne unngå kollisjon ved å følge «Konvensjon om internasjonale regler til forebygging av sammenstøt på sjøen» (COLREG).

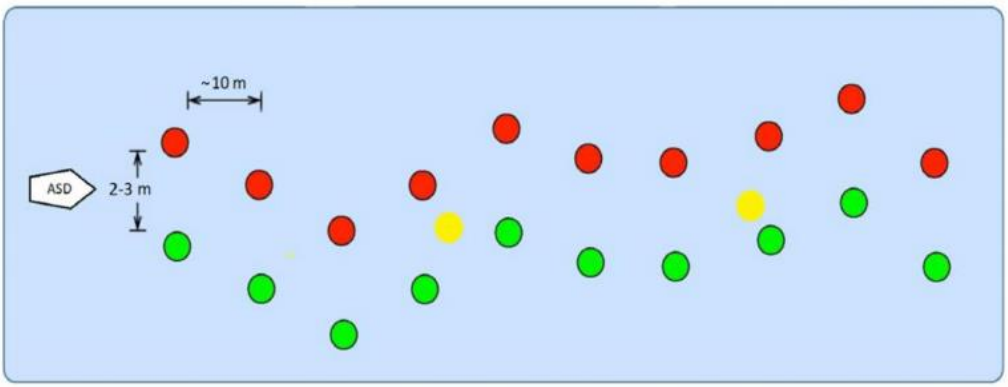


Figur 4: Illustrasjon av "Collision Avoidance [1]"

Points		
Gate	**Collision**	**COLREGs**
+10	-30	+30/rule followed

Figur 5: Poenggiving Collision avoidance [1]

Obstacle Challenge Mission, autodronen følger en kanal av porter, men møter hindringer underveis i form av gule bøyer. Den må kunne unngå disse og fremdeles følge portene til enden.

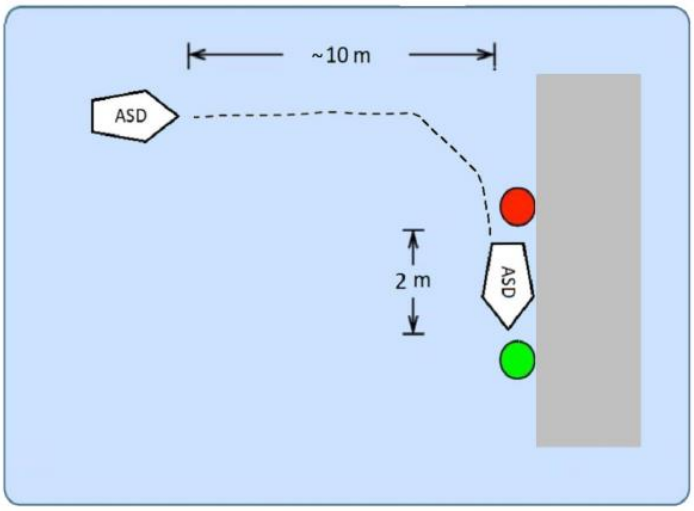


Figur 6: Illustrasjon av "Obstacle Challenge" [1]

Points		
Gate	**Obstacle**	**Time**
+5	-10	100 - 5/min

Figur 7: Poenggiving "Obstacle Challange"

Docking, autodronen skal kunne legge seg med barbord side til kai, mellom en rød og en grønn bøye, og holde seg der i 30 sekunder.



Figur 8: Illustrasjon av "Docking"

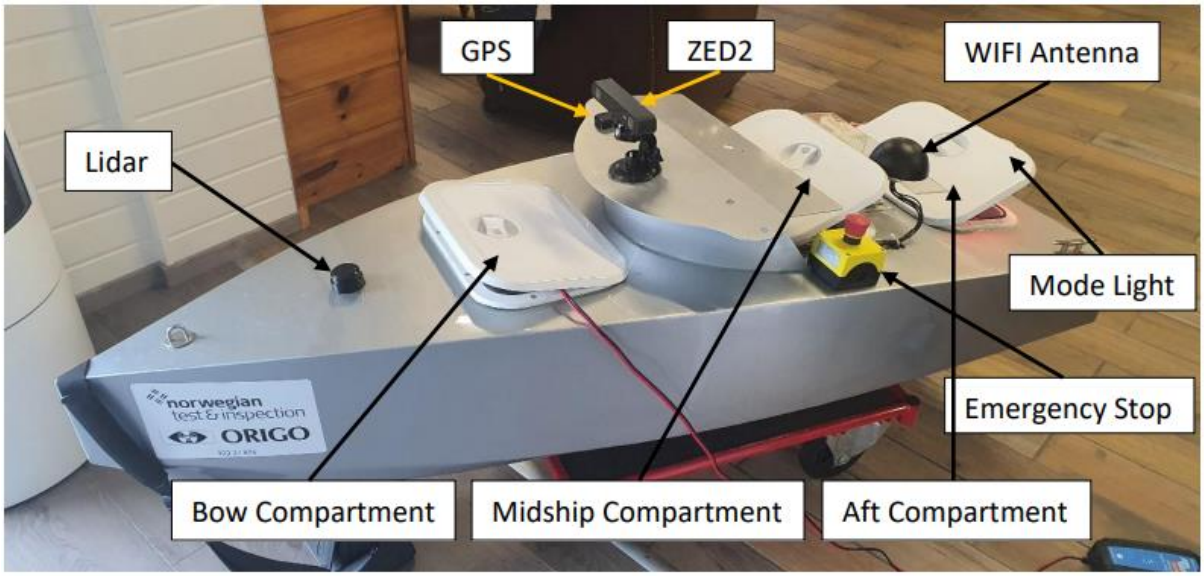
Points		
Dock Reached	**Docking correct**	**Docking time**
+10	+30	+ 2 /sec docked (max 60p)

Figur 9: Poenggiving "Docking"

6.2 HVL Haugesund autonome sjødrone

Sjødronen navigerer seg ved hjelp av to akter-monterte trustere, de står for både fremdrift og styring av dronen. Dronen kan kjøres autonomt, samt at man har mulighet til å styre dronen manuelt ved hjelp av en radiokontroller. Alle andre komponenter er følgende:

- **Modus-Lys** som indikerer hvilken modus dronen er i. Rødt lys indikerer at den ikke er i operasjon, gult lys indikerer at den blir styrt via radiokontrollen, blått lys indikerer at den er i autonom modus.
- **LiDAR** (*light detection and ranging*), som gir mulighet for å detektere avstand til objekter i 360 grader. Denne er foreløpig ikke tatt i bruk.
- **Ruter og WiFi-antenne** som håndterer intern kommunikasjon mellom Raspberry PI og Jetson Xavier, samt gir oss mulighet til å koble oss opp mot dronen over trådløst nettverk.
- **Nødstoppbryter** som kutter strømmen til trusterne.
- **Hovedstrømsbryter** som kutter strømmen til hele systemet.
- **Litium batterier.** Det er satt inn to stykk batterier som gir strøm til alle de ulike komponentene.
- **GPS-mottaker** som gir oss GPS-data.
- **Arduino Nano** er en mikrokontroller som har et sett med innganger og utganger. I vårt oppsett kommuniserer Arduino Nano med Raspberry PI over seriellport, som videre styrer fargen i moduslyset.



Figur 10: Oversikt over autodronen og dens hoveddeler (Foto: Privat)

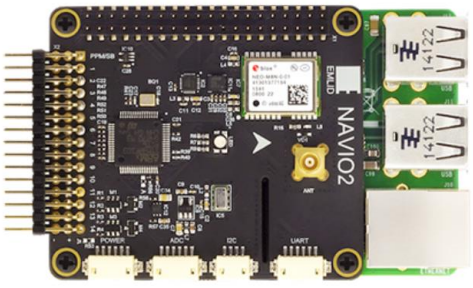
Raspberry Pi:

Raspberry Pi 3B+ er en mikroprosessor som kan kjøre et eget Linux basert operativsystem. Den er dermed en egen fullverdig *PC*, bare at den er av en miniatyr versjon, med litt større begrensninger. Den som vi bruker om bord på autodronen, kjører en egen *OS*, Raspberry Pi Buster, som er en pre-konfigurert versjon fra Emlid, som har innebygde funksjoner for Navio2 HAT med ROS allerede pre-installert.

Figur 11: Raspberry Pi 3B+ [2]

NAVIO2 HAT:

Navio2 HAT som er en *HAT (Hardware Attached on Top)*, denne monteres direkte på Raspberry Pi-en som vi har i autodronen. Dette gjør at disse to komponenten kan samhandle sømløst med hverandre, der Raspberry Pi-en tar jobben med å kjøre ROS, gjøre beregninger og kommunisere med overordnende komponenter, mens Navio2 fungerer mer som en *Input/Output* modul, som har innebygd *GPS* og *IMU*. Den kan også motta radiosignaler via radiomottaker, og styre trusterne på autodronen med pulsbreddemodulering.



Figur 12: Navio2 HAT [3]

NVIDIA Jetson Xavier NX:

Jetson Xavier er en type mikroprosessor som er på noen måter lik Raspberry Pi, der den er liten og kjører Linux basert OS, samt den også kjører ROS. Det som skiller Jetson seg fra Raspberry Pi, er at den er bygd med intensjon til å kunne håndtere spesifikke oppgaver bedre enn vanlige prosessorer. Da arkitekturen har lånt teknologi fra nyere grafikk kort, som gjør den usedvanlig god til å håndtere operasjoner som kunstig intelligens, maskinlæring og objektdeteksjon. Dette gjør den til et naturlig valg når vi skal bruke den til objekt deteksjon.

Figur 13: NVIDIA Jetson Xavier NX [4]

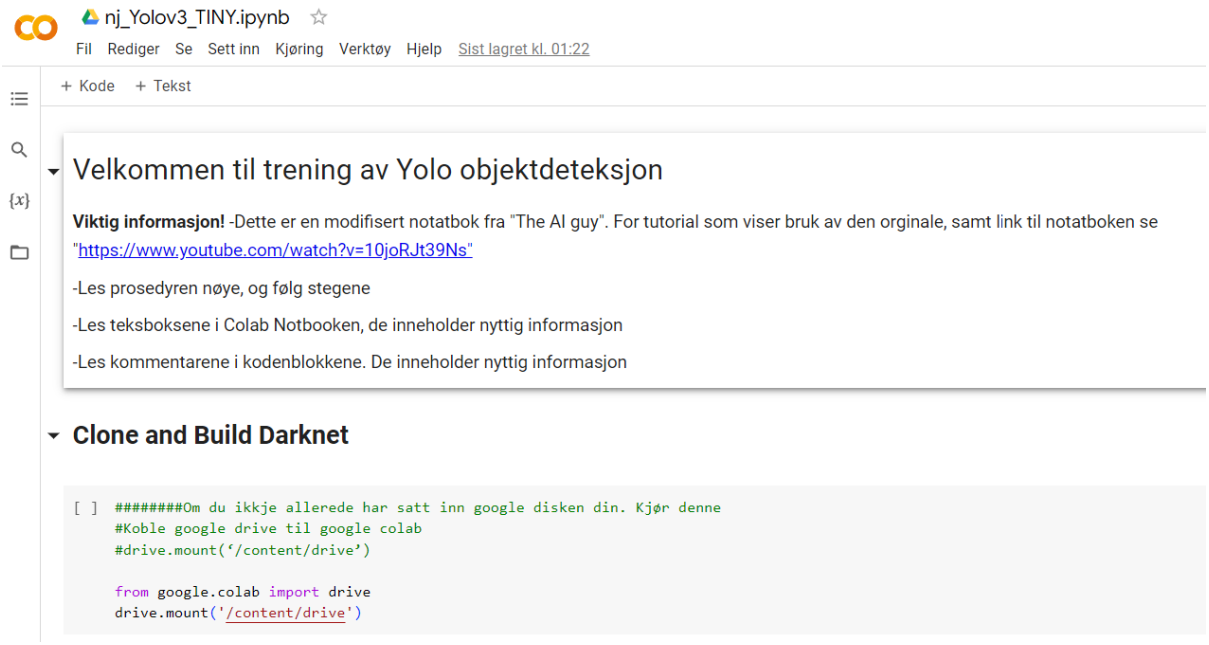
Zed2 stereokamera:

Zed2 Stereokameraet er en enhet som benytter seg av to kameraer, den kan dermed emulere det menneskelige syn, i den forstand at den kan lese av dybde i bilde ved å analysere endringer i bildene mellom de to kameraene. Den har også innebygd *IMU,* Magnetometer og barometer.

Figur 14: Zed2 Stereokamera [5]

6.3 Google Colab

Google Colab er en skybasert plattform som lar brukeren skrive og kjøre Python kode direkte i nettleseren. I tillegg gir Google Colab deg gratis tilgang til skybaserte GPU og CPU dataressurser. På mange måter kan man se på Google Colab som en svært kraftig virtuell datamaskin man kan benytte seg av gjennom nettleseren. Grunnet den resurskrevende prosessen med å trene opp objektdeteksjons algoritmer, har vi i arbeidet vårt benyttet oss av Google Colab. [6]



Figur 15: Skjermbilde av vår modifiserte Colab notatbok (Privat)

6.4 Yolo-V3 Tiny

YOLO (You Only Look Once) er enkelt forklart en objektdeteksjons algoritme som benytter seg av bilder som inngangsdata, og som ved hjelp av dype nevrale nettverk klarer å detektere objekter i bildene. YOLO V3 Tiny er en versjon av denne populære objektdeteksjons algoritmen som er optimalisert for å kunne behandle bildene raskt på begrensede ressurser. Dette går på kompromiss av presisjon og nøyaktighet, men som samtidig gjør den mer egnet til bruk der man er avhengig av å detektere objekter i sanntid. Derfor egner Yolo V3 Tiny seg godt til vårt bruk på den autonome sjødronen. [7]

6.5 ROS

ROS som er en forkortelse for Robot Operating System er et rammeverk med verktøy og biblioteker som kjører på toppen av et Linux-operativsystem. Det fungerer som et bindeledd mellom robotens maskinvare, og Linux-operativsystemet. Hensikten med ROS er å gjøre det enklere for utviklere å håndtere kommunikasjon, prosessere data og styre robotens maskinvare [8]. ROS består av en enorm mengde verktøy og funksjoner man kan benytte seg av for å utvikle roboter, noen av nøkkelfunksjonene er:

ROS Node:
ROS-noder er de individuelle programmene som utfører de ulike oppgavene i et ROS system. En node kan for eksempel ha ansvar for å motta data fra en spesifikk sensor, mens en annen node kan ha ansvar for å utføre beregninger på data fra alle systemets ulike sensorer. Fordelen ved bruk av ROS-noder, er at det gjør det enkelt å bygge et modulært system, der nodene kun har ansvar for spesifikke deler av systemet. Noder kan kommunisere med hverandre ved å publisere og abonnere på ulike ROS Topics.

Ros Topic:
Topics tillater ulike ROS-noder å publisere og abonnere på informasjonen som ligger på topicet. En node kan publisere informasjon til et bestemt topic, og andre noder kan abonnere på den samme topicen for å motta og behandle informasjonen. Dette muliggjør effektiv kommunikasjon og informasjonsutveksling mellom forskjellige deler av ROS-systemet.

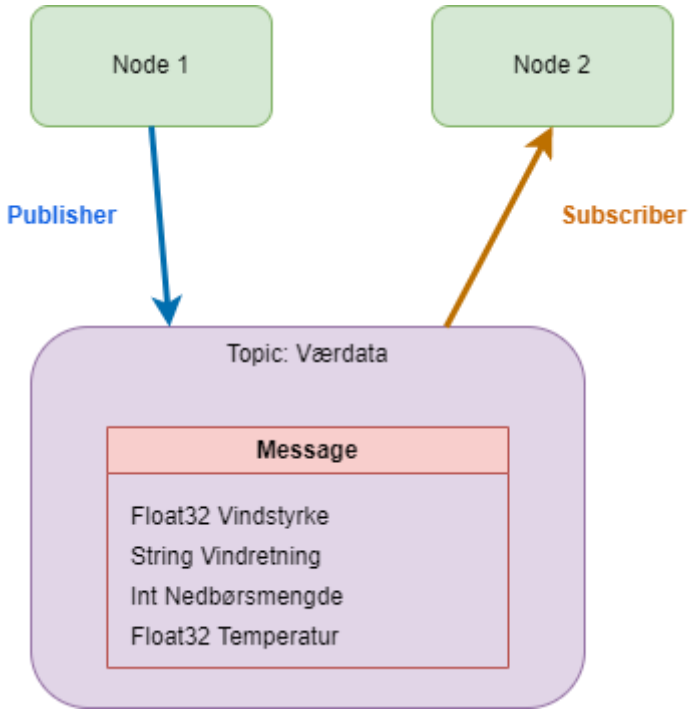
ROS Subscriber:
ROS subscriper er en funksjon som gjør det mulig å abonnere på informasjonen som ligger på et ROS Topic. Hver gang informasjonen på topicet blir oppdatert henter denne funksjonen den nyeste dataen og videreformidler dette til noden som benytter seg av subscriber funksjonen.

ROS Publisher:
ROS Publisher er en funksjon som gjør det mulig for en node å publisere ny informasjon på et ROS Topic. Si du har en node som får målinger fra tre ulike temperatursensorer. Noden

beregner gjennomsnittstemperaturen fra alle tre sensorene, for så å publisere gjennomsnittstemperaturen på et ROS Topic

ROS Message:

ROS Messages er datastrukturene som brukes for å sende informasjon mellom ROS-noder via topics. En ROS-Message kan være et sett av variabler eller felt som representerer ulike typer data, for eksempel tall, strenger eller egendefinerte datatyper. Man kan selv lage egendefinerte ROS Messages der man bestemmer strukturen på dataen som blir publisert på et ROS Topic. Messages er viktig for å få oversikt og struktur på hva slags informasjon som ligger på de ulike ROS Topicene. Et topic vil kun godta data som følger den gitte strukturen i ROS-Messagen den er satt opp med.



Figur 16: Illustrasjon av nøkkelfunksjoner i Ros (Privat)

En analogi som forklarer disse fem nøkkelfunksjonene er en gruppe personer som jobber med et prosjekt, der gruppen benytter seg av oppslagstavler for å formidle den nyeste tilgjengelige informasjonen. Hver oppslagstavle har en spesifikk struktur for hvordan informasjonen skal formidles. For eksempel kan vi spesifisere at oppslagstavlen for værdata skal ha vindstyrke på rad en, vindretning på rad to, nedbørsmengde på rad tre og temperatur på rad fire.

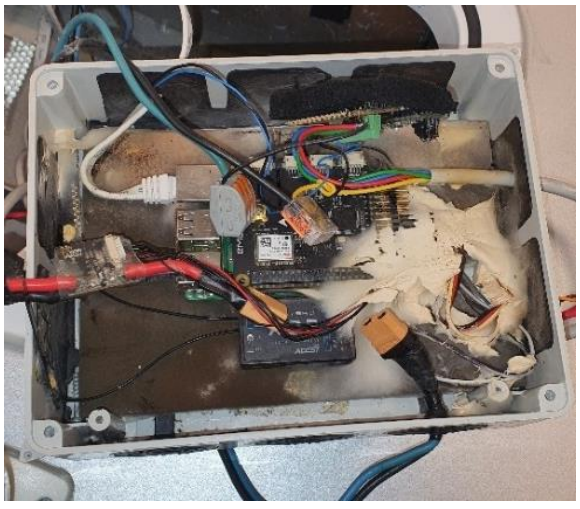
Alle som er interessert i et spesifikt emne kan henvende seg til tilhørende oppslagstavle og enten oppdatere oppslagstavlen med ny informasjon, eller ta med seg informasjonen fra tavlen til sin arbeidsstasjon og anvende den i sitt arbeid. Det er ingen direkte samtale mellom avsendere og mottakere, men informasjonen blir delt på en måte som er tilgjengelig for alle som er interessert i det spesifikke emnet.

I analogien vil de ulike personer som oppdaterer oppslagstavlen tilsvare ROS Publishers, de som leser informasjon og anvender det i sitt arbeid tilsvarer ROS Subscriber, arbeidsstasjonene tilsvarer ROS Nodes, oppslagstavlene tilsvarer ROS Topics, og de ulike oppsettene på oppslagstavlene tilsvarer ROS Messages.

7 Oppstartsproblemer

Da vi skulle hente dronen ut fra lageret fikk vi oss en stor overraskelse. Det viste seg at det hadde vært fukt i inne i skroget på dronen da den ble satt til vinterlagring. Problemet viste seg å være enda større da vi skrudde opp koblingsboksen der Raspberry Pi-en og Navio2 Hat'en stod. Koblingsboksen var full av mugg, og det var mye korrosjon på pinnene og kretskortet til Raspberry Pi'en. Det som hadde blitt ødelagt var Raspberry Pi'en, Navio2 Hat, Arduino Nano og hovedstrøms-bryteren.

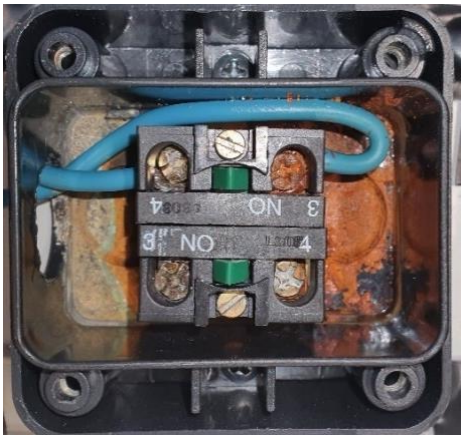
Dessuten vil vi nevne at koblingene var svært uoversiktlige og dårlig merket. Et annet problem vi fikk som følgefeil av alt som var ødelagt, var at vi manglet kildekoden som var installert på minnekortet. Det viste seg at den forrige gruppen hadde tatt sikkerhetskopi av alt som lå på Raspberry Pien, men dette var dessverre tatt i feil format. Når vi da fikk tilgang til sikkerhetskopien som var sky-lagret i OneDrive, så vi at sikkerhetskopien inneholdt mengder av tomme mapper og korrupte filer. Grunnen til dette er at når man tar sikkerhetskopi av et helt Linux system, inkludert selve operativsystemet klarer ikke Windows-baserte systemer å kjenne igjen mange av filene. Det samme problemet var til stede på en USB den ene veilederen hadde fått overrekt.



Figur 17: Bilde av fuktskadet koblingsboks (Privat)

7.1 Utbedring av oppstartsproblemer

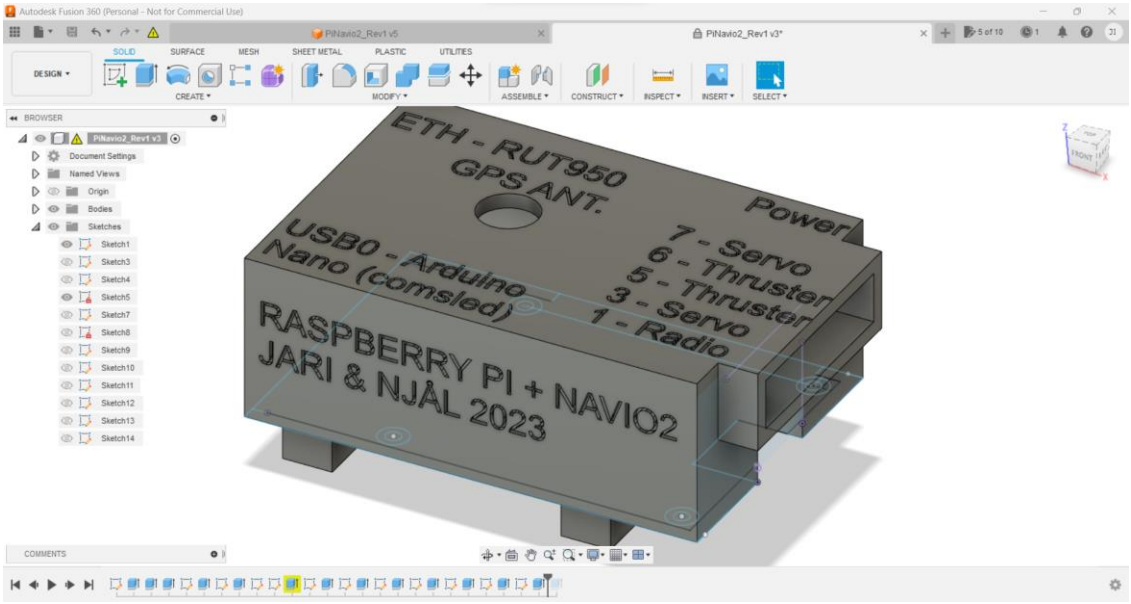
Det første vi måtte gjøre var å få på plass en ny Raspberry Pi og tilhørende Navio2-Hat, da dette er selve hjernen i systemet. Heldigvis hadde den ene veilederen en Raspberry Pi med Navio2 Hat liggende, så vi fikk byttet ut denne. Etter mye prøving og feiling fikk vi heldigvis til å ta en fullverdig sikkerhetskopi av det gamle minnekortet som stod i Raspberry Pi-en. Her må vi kunne si at vi hadde litt flaks både fordi minnekortet ikke hadde blitt ødelagt, men også da dette minnekortet plutselig sluttet å fungere på et senere tidspunkt. Da vi skulle skru på dronen oppdaget vi at den ikke fikk strøm, etter litt feilsøking kunne vi konstatere at hovedstrøms-bryteren ikke fungerte fordi kontakt-settet hadde rustet i stykker.



Figur 18: Bilde av defekt hovedstrømsbryter (Privat)

Etter at hovedstrømsbryter var byttet ut, fikk vi videre ikke til å koble oss opp mot nettverket, da passordet som var oppgitt i dokumentasjonen var feil. Derfor måtte vi resette ruteren, og sette opp hele nettverket på nytt. Da dette var gjort fikk vi kontakt med dronen, og ting så etter forholdene greit ut. Det siste som da gjenstod for å få dronene tilbake i «original» stand var å bytte ut Arduino Nano med tilhørende kretskort. Arduino Nano byttet vi ut, men problemet her var at vi ikke hadde tilgang til koden som lå på Arduinoen. Derfor måtte vi «*reverse engineer*» koden, slik at den kunne styre modus-lyset. Arduinoen kommuniserer via seriell port med Raspberry Pi, og mottar data om hvilken modus dronen er i. Den skal da endre farge på moduslyset tilsvarende den modusen den er i.

I tillegg 3D-printet vi to kapslinger som vi monterte Arduino Nano og Raspberry PI i. Dette skal beskytte disse to komponentene mot overslag og kortslutning, samt gjøre koblingsboksen mer oversiktlig.



Figur 19: Skjermbilde av Fusion 360, som vi bruker til 3d Modellering (Privat)

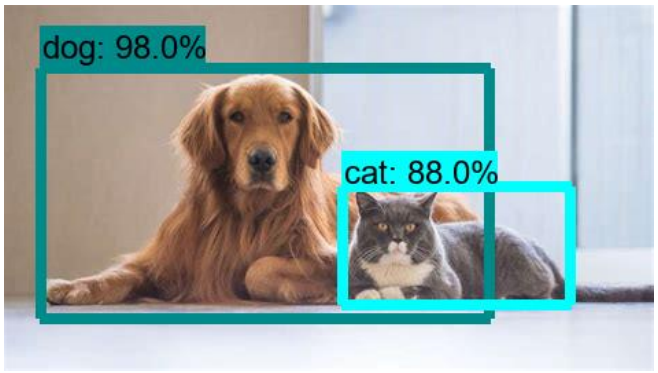
7.2 Resultat

Resultatet av dette arbeidet er først og fremst at vi fikk en fungerende sjødrone, som er i samme stand som da den ble satt til lagring sommeren 2022. Samtidig fikk vi sikret sjødronen bedre mot vanninntregning ved å tette lukene med silikon, samt sikret Raspberry Pi og Arduino Nano bedre mot overslag og kortslutning.

Figur 20: Bilde av utbedret koblingsboks (Privat)

8 Objektdeteksjon

Objektdeteksjon i maskinsyn sammenheng, handler om å lære en algoritme å identifisere og lokalisere spesifikke objekter i bilder. Objektdeteksjons algoritmen lærer å gjenkjenne forskjellige objekter, samtidig som den rammer dem inn slik at vi har mulighet til å anslå posisjonen til det ønskede objektet i bildet.



Figur 21: Eksempel objektdeteksjon [9]

For at dronen skal klare å navigere seg gjennom de ulike øvelsene er den helt avhengig av å bruke maskinsyn og en objektdeteksjons algoritme. Derfor er det montert et stereokamera på dronen. Stereokameraet bruker vi blant annet til å måle avstand til ønskede objekt, samt til å gi objektdeteksjonsalgoritmen sanntidsbilder å jobbe med.

Objekt deteksjonsalgoritmen vi valgte å bruke er Yolo V3-Tiny, grunnlaget for dette er at det er en anerkjent og mye brukt algoritme. Samtidig er denne godt tilpasset til bruk i ROS [10]. Dessuten er den også hurtig nok til å håndtere bildestrømmer i sanntid, noe som er avgjørende i vårt system [11]. I tillegg er denne algoritmen fra tidligere installert og tatt i bruk på sjødronen, noe som sparte oss for en del tid.

Prosessen for å få til en fungerende objekt-deteksjon kan deles opp i tre deler:

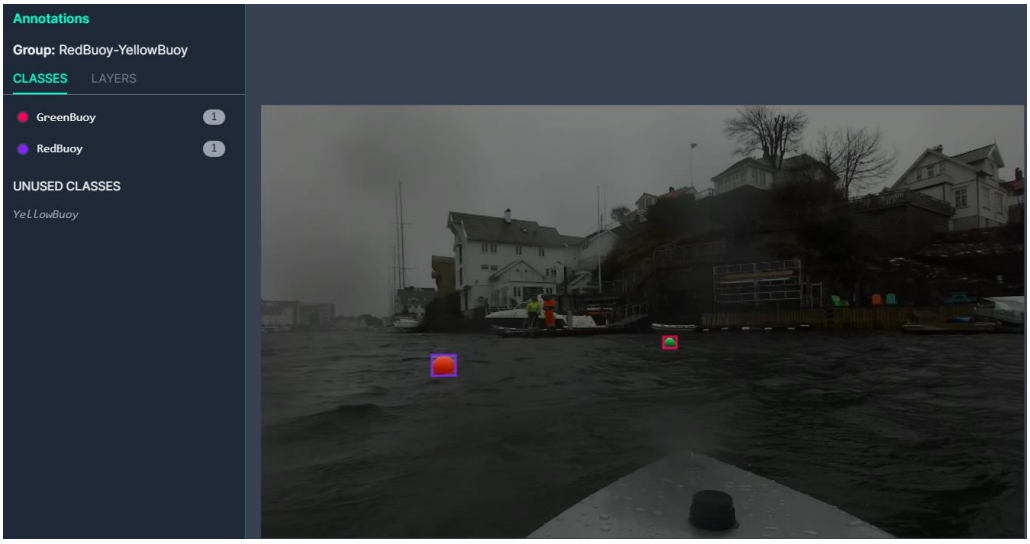
- Innhenting av data og generering av datasett. Her inngår det å samle inn gode bilder, for deretter å manuelt annotere bildene og generere et datasett.
- Trening og validering. Her inngår arbeidet med å bruke datasettet til å trene opp modellen.
- Implementering og testing.

Før vi går videre til metode, vil vi gjennomgå litt teori som omhandler datasett og trening av objekt-deteksjon på generelt grunnlag.

8.1 Teori Objekt-deteksjon

8.1.1 Viktigheten av et godt datasett

Et datasett for objekt deteksjon er i bunn og grunn bare en samling med bilder som inneholder et eller flere objekter man ønsker å detektere, og en tilhørende tekstfil for hvert bilde. Tekstfilen inneholder informasjon om hvilket objekt som befinner seg i bildet, samt plasseringen til objektet. I vårt tilfelle vil plasseringen av objektet være angitt av en firkant som omslutter det ønskede objektet, også kalt *bounding box*. Denne tekstfilen er et resultat av manuell annotering av hvert eneste bilde i datasettet. Å annotere et bilde, vil i denne sammenheng bety at en operatør bruker programvare til å manuelt tegne en *bounding box* rundt det ønskede objektet, for så å navnsette *bounding box* med korrekt objekt.



Figur 22: Skjermbilde av annotering i Roboflow (Privat)

Sluttresultat fra trening av en objektdeteksjons algoritme er helt avhengig av kvaliteten på datasettet, både antall bilder og innholdet i disse spiller en avgjørende rolle for at sluttresultatet skal bli bra [12]. Derfor er det helt essensielt at de innsamlede bildene er av god kvalitet, og samtidig inneholder variasjon i form av bilder tatt av objektet i ulike lysforhold, med varierende avstand, og med varierende posisjon og orientering. Uttrykket «*Garbage in, Garbage out*» er i høyeste grad gjeldende i denne sammenheng.

En annen viktig faktor som avgjør kvaliteten på datasettet, er hvor god annoteringen av bildene er. Dette betyr at *bounding box* skal ha korrekt navn og omslutte hele objektet, uten

å ta med for mye av bakgrunnen. I tillegg er det også viktig at datasettet ikke inneholder «tomme» *bounding box*-er, eller inneholder bilder der ønskede objekt er til stede, men det ikke er satt *bounding box* rundt. I en studie som omhandler utviklingen av en algoritme som skal detektere dårlig annotering av bilder i datasett til trening av objektdeteksjon, har de blant annet utført tester der de ser på effekten av dårlig annotering i datasett. Gruppen la til ulik støy i et datasett og sammenlignet det med resultatet fra det samme datasettet uten støy. De ulike formene for støy var blant annet å angi feil navn på *bounding box*-er, skalere opp/ned *bounding box*-er, forskyve *bounding box*-er, legge til tomme *bounding box*-er og å fjerne korrekt annoterte *bounding box*-er. Resultatet viste at alle formene for støy hadde en markant effekt på resultatet, og støyen som hadde størst påvirkning var *bounding box*-er med feil navnsetting [13].

«*if a person shows a pencil to a child and says it is a pen, then after some time, the child sees a pencil, the child will classify it as a pen*» [12, p. 7].

8.1.2 Trening av nevralt nettverk objekt-deteksjons algoritmer

Trening av objekt-deteksjonsalgoritmer betyr å mate en objekt-deteksjonsalgoritme med store mengder data, slik at den kan lære å kjenne igjen ønskede objekter. Gjennom trening lærer algoritmen å kjenne igjen en rekke egenskaper ved de ønskede objektene. Dette kan blant annet være form, farge eller tekstur. I treningsprosessen vil algoritmen kontinuerlig sammenligne seg selv mot «fasiten» i datasettet den trener på, for så å kontinuerlig justere sine parameter basert på feilene den gjør. På denne måten vil den gradvis oppnå en mindre og mindre feilmargin gjennom treningen. [14]

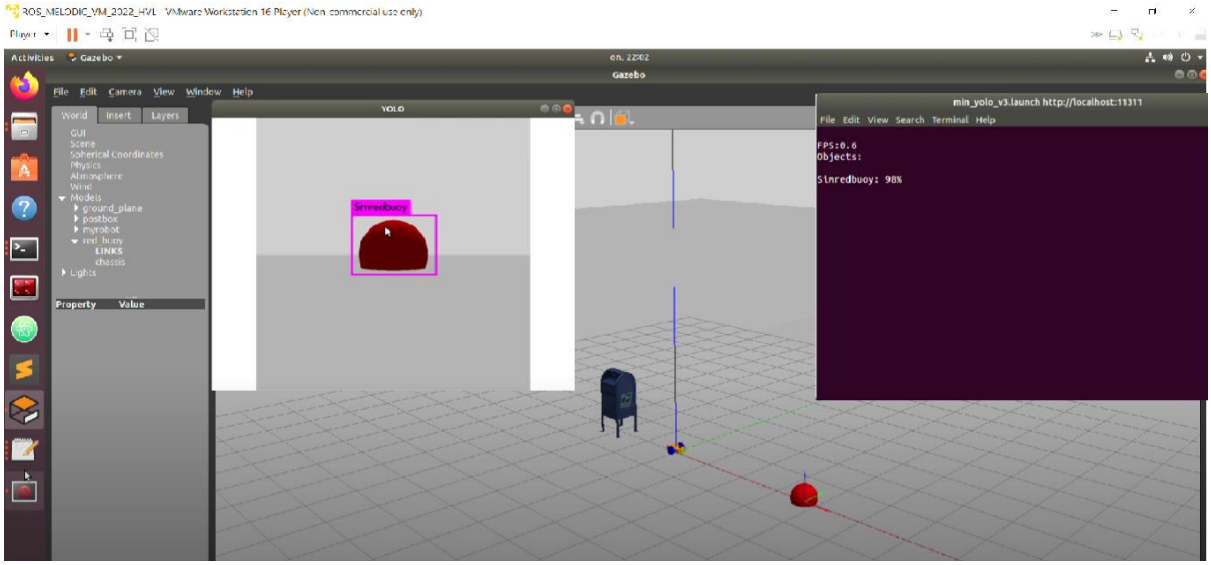
Sluttresultatet av en trening vil produsere en «vekt-fil». Man kan på mange måter si at vektfilen inneholder selve lærdommen og kunnskapen algoritmen har oppnådd under trening. På denne måten kan man ved å lagre vektfilen, flytte den og benytte seg av samme lærdom i ulike system. Grunnet den store mengden med data algoritmen må prosessere

under trening, krever dette kraftige beregningsressurser. Typisk er man avhengig av tilgang til GPU(Grafikkprosessor)-kraft når man trener, da disse er spesielt tilpasset til å behandle grafikkdata [15].

I tillegg til selve treningen bør man gjennomføre en validering av sluttresultatet. For å validere sluttresultatet bruker man et eget valideringsdatasett. Dette valideringsdatasettet er også annotert, men skal ikke inneholde de samme bildene som er brukt under treningen. Grunnen til at valideringsdatasettet skal inneholde ulike bilder, er for å unngå *overfitting*. Med *overfitting* i maskinlærings sammenheng, snakker vi om et fenomen der modellen kjenner igjen støyen fra treningsdatasettet i stedet for objektet selv. Derfor er det viktig at modellen valideres på andre bilder enn det den ble trent på, slik at man bedre sikrer et reelt resultat. Måten en validering blir utført på, er at man lar den ferdigtrente objektdeteksjons-algoritmen gå gjennom bildene i valideringsdatasettet og anslå hvor objektene er plassert. Siden bildene i valideringsdatasettet også er annotert, sammenlignes antakelsene mot annoteringene som ligger i datasettet. Selve grunnen til at man utfører en validering, er at man får en pekepinn på hvor godt sluttresultatet av treningen er.

8.2 Forbedring av objektdeteksjonen

Siden vi i starten ikke fikk muligheten til å benytte oss av systemet på dronen grunnet oppstartsproblemene drøftet i kapittelet «Oppstartsproblemer», bestemte vi oss for å utføre et testeksperiment. Her skulle vi gjennomføre trening og implementere en Yolo-V3 tiny modell i et simuleringsmiljø i ROS. På denne måten kunne vi sikre oss at vi hadde god kontroll på prosessen, frem til vi fikk mulighet til å gjøre dette i full skala. Under gjennomføringen av testprosjektet var det mye prøving og feiling. En stor andel av tiden gikk til problematikk knyttet til å få objekt-deteksjonen til å fungere i ROS. Til syvende og sist viste dette problemet seg å være tilknyttet at konfigurasjonsfiler som hadde blitt modifisert i Windows under treningsprosessen, hadde feil linjeendelse, noe som gjorde at ROS som er Linux basert ikke klarte å tolke disse. Dessuten fikk vi klargjort en Google Colab notatbok, og diverse prosedyrer som beskriver prosessen fra start til slutt. Dette gjorde at vi kunne gjennomføre de kommende treningene sømløst.



Figur 23: Skjermbilde av simulering av Objektdeteksjon i Gazebo (Privat)

Ettersom forrige bachelorgruppe fikk en objekt-deteksjonsmodell som tilsynelatende fungerte bra da de testet den på land, men som ikke fungerte da de stilte i konkurranse og dronen faktisk var i sjøen, måtte vi prøve å finne en løsning på dette. Vi fant ut at datasettet fjorårets gruppe hadde brukt til å trene objekt detektoren på, kun bestod av bilder som var tatt mens bøyene var på land, eller at bøyene var i sjøen, men bildet var tatt med mobiltelefon fra land. Etter en gjennomgang av det tidligere datasettet kom vi frem til noen hypoteser om hvorfor objekt detektoren ikke fungerte tilstrekkelig i konkurransen. Bildene i datasettet er i stor grad av bøyer som er plassert innendørs. Dette kan medføre en del problematikk da lysforhold og omgivelsene rundt bøyene ikke er representative for sanntidsbildene stereokameraet vil gi under konkurransen. En annen ting vi fant som kunne være problematisk med bildene av bøyer i sjøen i datasettet, er at bildene er tatt fra land. Dette medfører at vinkelen fra kameraet til bøyen ikke vil representere det kameraet på dronen kommer til å se.

Figur 24: Tidligere datasett innendørs [16]

Figur 25: Tidligere datasett utendørs [16]

For å videre underbygge at disse hypotesene kunne stemme, foretok vi en leting i studier og forskningsartikler for å se om vi kunne finne informasjon som bygde opp under våre hypoteser. Vi fant blant annet en studie gjennomført ved «Universety of Alicante» i 2018, som omhandlet datasett til bruk i autonome biler. Der kom det frem at det å legge til bilder fra UDacity datasettet forbedret både nøyaktigheten og robustheten til objekt-deteksjonen når den skulle detektere biler. Grunnen til at dette forbedret resultatet skyldes at alle bildene i UDacity datasettet er tatt fra bilens perspektiv, og derfor godt representerer hva et kamera montert på en bil vil se [17]. UDacity datasettet er en samling av videoklipp og bilder som brukes til å trene og evaluere objektdeteksjons-systemer for selvkjørende biler. [18]

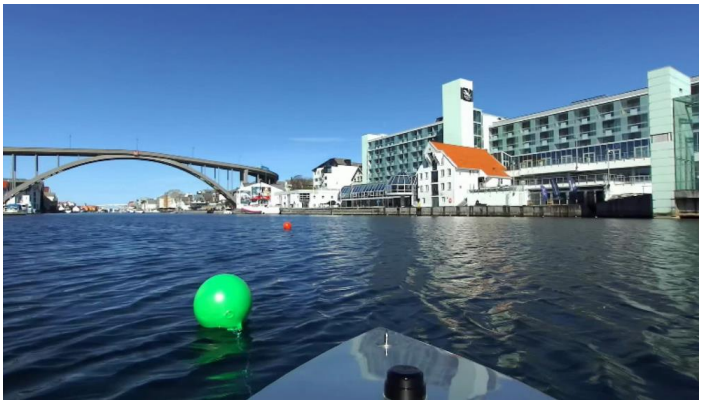
I en annen forskningsartikkel som omhandlet en metode for å automatisk kunne lage datasett til trening av objekt deteksjon, står det skrevet at den største hindringen for å anvende objekt deteksjon i praksis er å generere et spesifikt trenings datasett:

«*Numerous high-accuracy neural networks for object detection have been proposed in recent years. Creating a specific training dataset seems to be the main obstacle in putting them into practice.*» [19, p. 1].

Med dette konkluderte vi med en rekke forbedringstiltak før vi begynte å innhente bilder til vårt eget datasett:

- Bildene skal være tatt av kameraet på dronen, slik at vinkel og bildekvalitet samsvarer med det dronen vil se i en konkurranse setting.
- Bildene skal være tatt utendørs, med bøyene i sjøen. Dette gjør vi for å få mest mulig realistiske lysforhold og omgivelser.
- Vi tar bilder i ulikt vær slik at dronen skal kjenne igjen bøyene best mulig enten det er overskyet, regn eller klar himmel.

Figur 26: Foto tatt av autodrone under overskyet vær (Privat)



Figur 27: Foto tatt av autodronen med klar himmel og sol (Privat)

Selve innsamlingen av bilder viste seg å være en tidkrevende prosess i seg selv, da det ikke fantes en enkel metode å faktisk hente ut bildene fra kameraet i et lesbart format. Det første vi gjorde var å lage et Python-script som henter data som blir publisert på et rostopic, for deretter å konvertere dette til et lesbart format og lagre det. Dette fungerte egentlig ganske bra, men det største problemet var at begrensinger i *softwaren* og oppsettet gjorde at vi kun klarte å lagre 1-2 bilder per sekund. Deretter så vi at selve ZED2 kameraet hadde en innebygd funksjon som lagret rådataen til en fil. Rådataen fra kameraet inneholder bildestrømmen fra både fra venstre og høyre kamera, samt dybde informasjon og mye mer. Derfor måtte vi installere en programvare som heter ZED SDK, for videre å kunne hente ut videofilene [20].

Arbeidet med å lage et godt datasett er enormt tidkrevende. Først må vi sette bøyene ut i sjøen, for deretter å styre sjødronen via radiokontrollen samtidig som den tar opptak. Når rådataene er samlet inn, må vi konvertere dette til lesbare videofiler for deretter å splitte

videofilene inn i individuelle bilder. Siden opptaket av rådataen inneholder cirka 25 bilder per sekund, så sitter vi igjen med veldig mange bilder som er mer eller mindre identiske, eller som ikke er relevante å ha i datasettet. Videre må vi da gå igjennom disse bildene og plukke ut de vi skal ta med i datasettet.

Når vi så er ferdige med datainnsamlingen begynner enda en tidkrevende oppgave, nemlig annotering av bilder. Til annoteringsarbeidet brukte vi en nettside som heter Roboflow, her laster man opp bildene man vil annotere [21]. Selve annoteringen gjøres ved at man ved hjelp av funksjonene i Roboflow manuelt tegner en firkant rundt bøyene, og setter riktig navn på dem. Til sammen har vi annotert nesten 2000 bilder, som inneholder en eller flere bøyer. Etter annoteringen er fullført, er det viktig å ta en gjennomgang av de annoterte bildene, slik at man kan minimere sjansen for at det finnes annoteringer av dårlig kvalitet. Spesielt viktig er det å oppdage bøyer som er markert med feil navn. Etterfulgt av dette deler vi opp de annoterte bildene i to datasett, et som brukes til å trene objekt-deteksjons algoritmen på, og et som brukes til å validere treningen.
Roboflow har også innebygde funksjoner som gjør det mulig å øke antallet bilder i datasettet. Dette gjøres ved at man kan legge til modifikasjoner på bildene, i form av å legge til blant annet støy, endre kontrast, lysstyrke og fargemetning på bildene. Det blir da dannet kopier av de originale bildene, bare at det er lagt til modifikasjoner. I gratisversjonen av Roboflow kan man på denne måten tredoble antall annoterte bilder i datasettet.

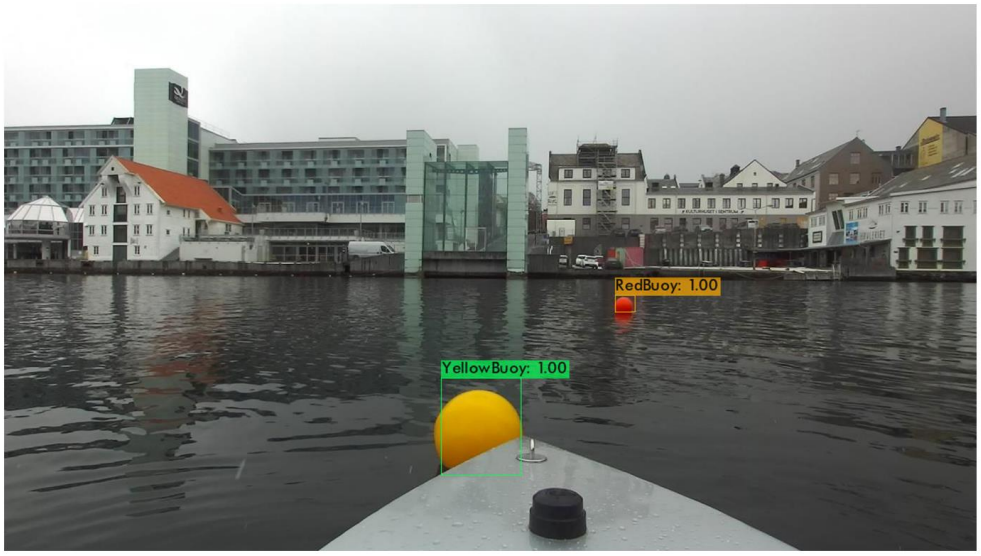
Selve treningen av objekt-deteksjons algoritmen gjøres i Google Colab. Google Colab er en sky-basert form for virtuelldatamaskin. Her kan du lage en notatbok der du har muligheten til å kjøre ulike script, og i tillegg får du tilgang til å låne CPU og GPU kraft. Dette gjør det mulig for oss å låne GPU kraft til å trene objektdeteksjons algoritmen helt gratis. «The AI guy» har laget en gjennomgangsvideo og en korresponderende Colab notatbok som viser hvordan man skal trene opp Yolo-V3 algoritmen [22]. Denne notatboken har vi modifisert og tilpasset til vårt bruk, slik at vi kan trene opp Yolo-V3 Tiny algoritmen.

Når treningen er gjennomført kjører vi en validering av modellen vår. Til dette bruker vi valideringsdatasettet. På denne måten får vi en pekepinn på for vellykket treningen var, og hvor godt modellen klarer å detektere de ønskede objektene. Resultatet av treningen ligger

nå i en vektfil. Denne overfører vi til sjødronen, og implementerer den i systemet slik at vi kan teste og ta i bruk objekt-deteksjonen.

8.3 Resultat

Sluttresultatet av arbeidet vårt viser at vi har fått til en objektdeteksjon som klarer å detektere bøyene under drift. Under fullskala testing, ser vi at dronen detekterer bøyene med forholdsvis høy sannsynlighet. Det oppstår noen problemer ved deteksjon av bøyene på avstander mellom 20 og 30 meter, hvor vi har tilfeller der bøyene får tildelt lav sannsynlighet eller blir feilklassifisert. På avstander over 30 meter er det vanskelig å detektere bøyene, og det er høy sjanse for feilklassifisering.



Figur 28: Resultat objektdeteksjon

Valideringen av de ferdigtrente vektfilene viser at objektdeteksjonen korrekt detekterer 73.6% av alle bøyene som er til stede i annoteringsdatasettet, dette kalles for *recall*. I tillegg viser resultatet av valideringen at av alle deteksjoner algoritmen gjør, så er 76% av dem korrekt, dette kalles for *precision.*

$$recall = \frac{Korrekt\ detkterte\ bøyer}{Antall\ bøyer\ til\ stede\ i\ datasettet} = \frac{369}{501} = 73.6\%$$

$$precision = \frac{Korrekt\ detektere\ bøyer}{Totalt\ antall\ deteksjoner} = \frac{369}{488} = 75.6\%$$

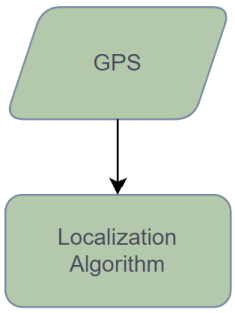
9 Lokalisering

Lokalisering i robotikk sammenheng, refererer til prosessen med å bestemme den nåværende posisjonen og orienteringen til en robot [23, pp. 151-160]. Målet med lokalisering er å oppnå nøyaktig posisjon og orientering i forhold til koordinatsystemet man beveger seg i. Dette gjør lokalisering til en av de viktigste oppgavene til roboten for at den skal kunne operere autonomt og bevege seg ute i den fysiske verden. På mange måter kan man si at robotens evne til å lokalisere seg selv, er selve grunnsteinen i en autonom robot.

En av de viktigste og mest utfordrende oppgavene å få på plass har vært lokalisering. Det er dette systemet som er ansvarlig for å estimere hvor autodronen er plassert geografisk.
Et system som bruker GPS, vil bruke et globalt koordinatsystem som dekker hele jordkloden, mens et system som kun baserer seg på gyroskop og akselerometer vil bruke et mer lokalt koordinatsystem.

9.1 Aktuelle lokaliserings metoder

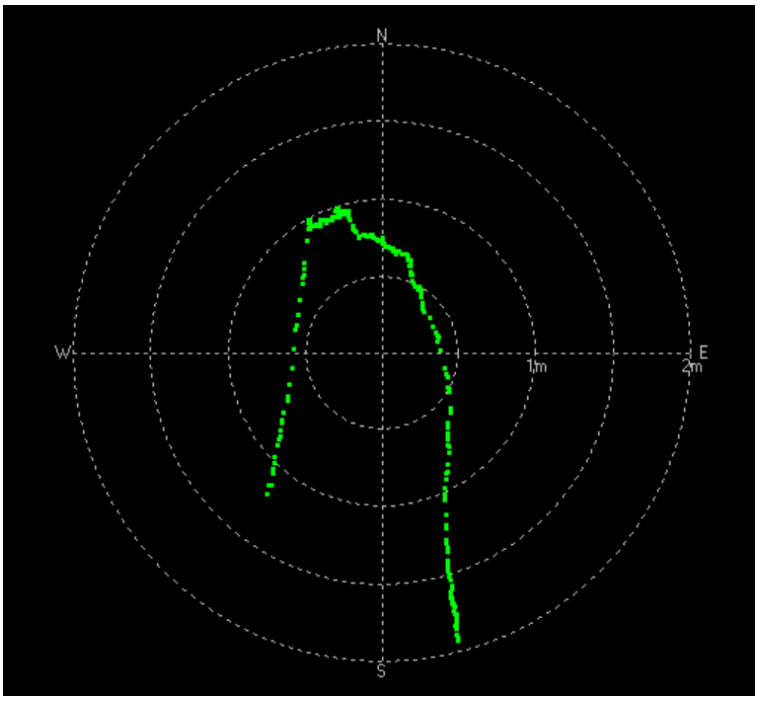
GPS Lokalisering: Lokaliserings metoden forrige bachelorgruppe benyttet seg av, brukte dette oppsettet for å anslå posisjonen til dronen. Her leses posisjonen fra GPS, og brukes så direkte til å posisjonere dronen.



Figur 29: Illustrasjon av GPS lokaliserings metode (Privat)

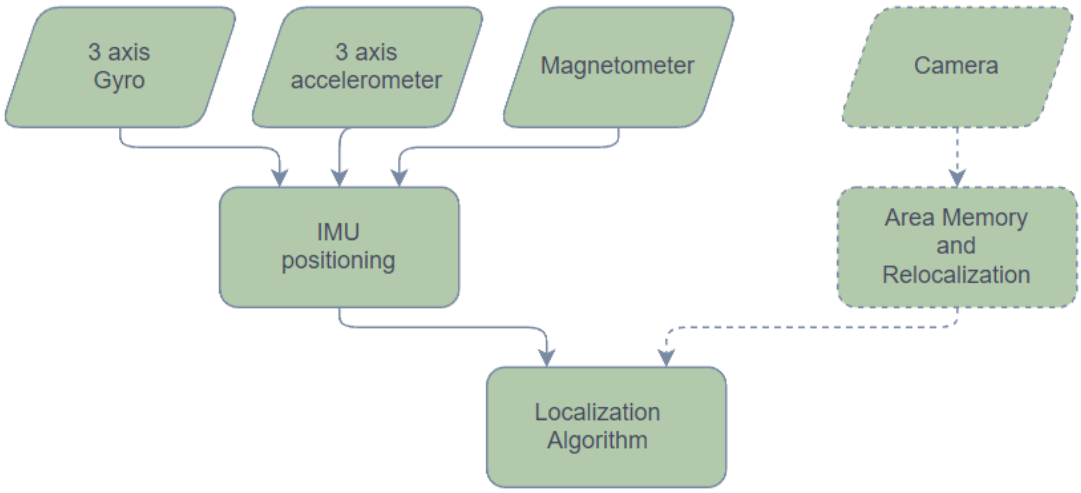
Ved testing da vi startet vår oppgave, fant vi at denne metoden har en høyere feilmargin enn det vi kunne tillate med hensyn til konkurranseforholdene. Da det er 2-3 meter imellom bøyene dronen skal passere gjennom, og vi har en målt feilmargin med radius på 2 meter på

et minutt når dronen er stasjonær.



Figur 30: Illustrasjon over GPS koordinater mottatt (Privat)

ZED2 Positional Tracking: Denne metoden bruker Zed2 kameraets innebygde sensorer for å gjøre estimater på hvor den har forflyttet seg. Sensorer brukt her er 3 akse gyroskop, 3 akse akselerometer og magnetometer, alt dette befinner seg i IMU sensoren.

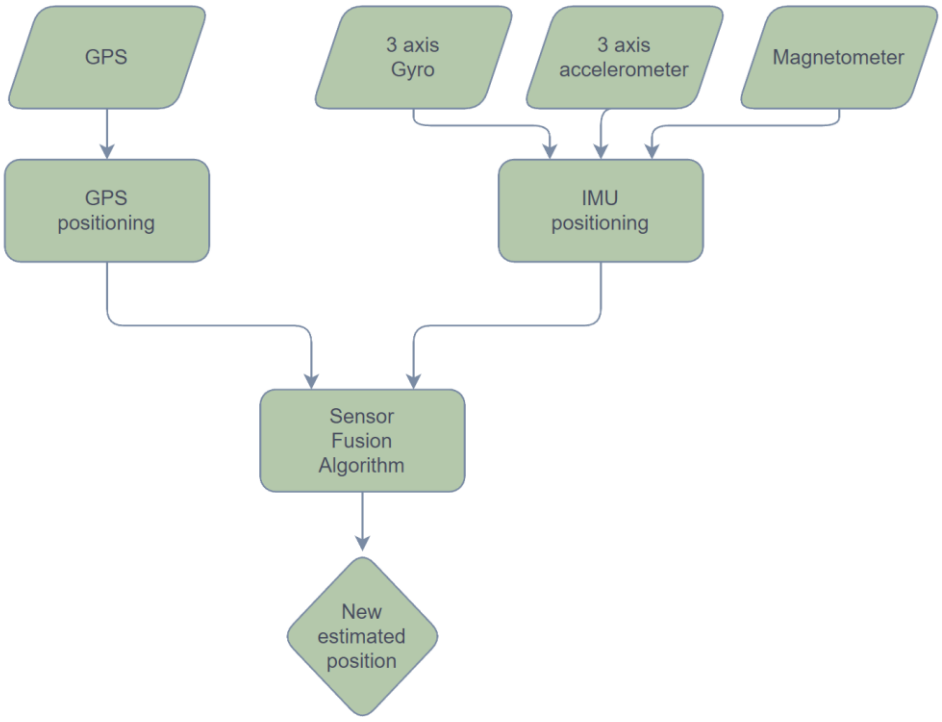


Figur 31: Illustrasjon av Positional Tracking metode (Privat)

Dette har vi sett har god suksess på land, men i sjøen har vi ikke prøvd denne metoden før. Denne metoden vil under perfekte forhold ha en liten feil margin, og etter vår erfaring

under testing, var disse for store til å kunne oppnå noen suksess. Metoden kan også inkludere områdeminne for relokasjon, der vi kan bruke kameraet for å gjenkjenne landemerker, slik at vi kan reposisjonere dronen til en kjent lokasjon om vi ser et kjent landemerke, og dermed fjerne den økende feilmarginen på lokaliseringen. Men under konkurranse forhold vil vi ikke ha noen landemerker vi kan bruke, ettersom vi befinner oss ute på sjøen, og den vil ha problemer med å stadfeste landmerker da vi har bevegelser i sjøen, som strøm og bølger.

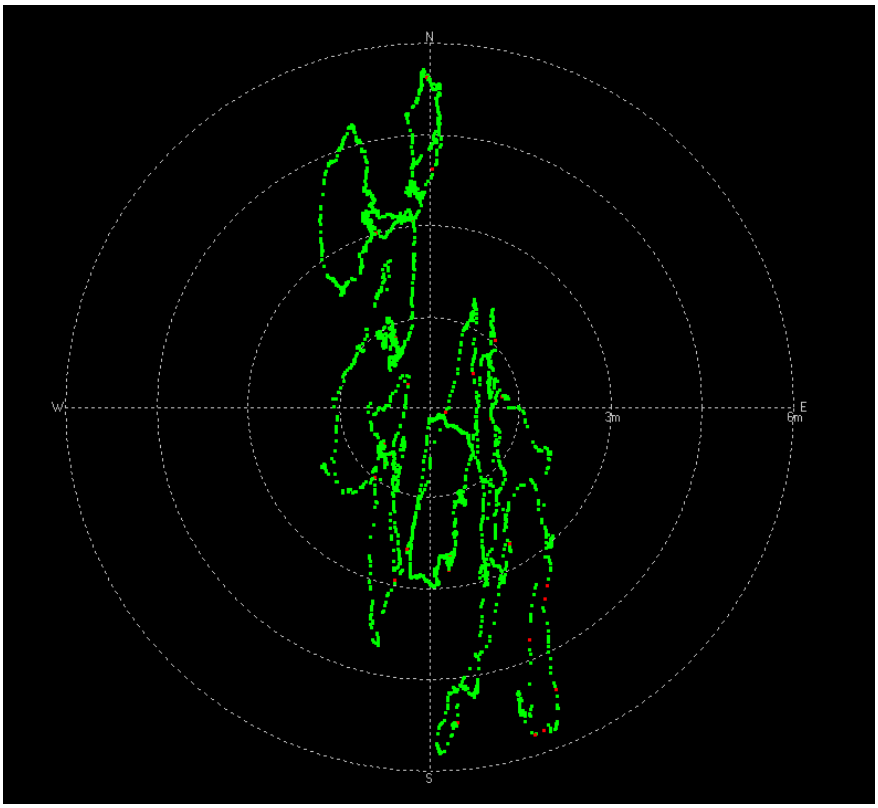
Sensor Fusion, IMU + GPS: Sensor fusion er en metode der vi kan bruke to eller flere typer lokaliserings sensorer, for å forbedre lokaliseringen av dronen. Her tar vi GPS koordinatene og legger de sammen med IMU bevegelse data, og anslår så ut ifra det, hvor dronen er. Vi velger å bruke en vektet metode her, da GPS koordinatene er mer nøyaktige, og vektlegger dermed disse dataene mer enn de vi får fra IMU sensoren. Vi får dermed mer presis lokalisering av dronen. Det er denne metoden vi har endt med å bruke i oppgaven.



Figur 32: Illustrasjon av Sensor Fusion (Privat)

9.2 GPS sensor drift

GPS sensoren som dronen bruker ligger på Navio2 kortet, selve GPS-modulen er en u-blox M8 GNSS mottaker. I starten av oppgaven la vi merke til at vi fikk en økende feil ved navigering til GPS-koordinater. Vi konkluderte med det at vi testet dette med å sette autodronen på land og logge endringer i posisjon over tid uten å bevege autodronen. Det vi da kunne se, var at etter et minutt, så fikk vi koordinater fra GPS, som viste oss at vi får en feilmargin på ca. tre meter. Etter ca. 10 minutt med logging, så hadde vi nesten 12 meter i avvik i ytterpunktene av loggingen. Dette er en svært høy feilmargin på posisjonering.



Figur 33: Illustrasjon over økende feil margin i mottatte GPS koordinater (Privat)

Testområdet vårt, der vi tester autodronen, er nede i Smedasundet i Haugesund. Det er relativt åpent mot himmelen, men det finns flate overflater fra bygg nærliggende til sjøen. Disse flate overflatene, kan skape refleksjoner av *GPS* signaler, som skaper støy i signalet, eller at signaler går tapt. I åpent farvann kan dette fenomenet reduseres en del, og vi kan få mer presist *GPS* koordinat. Siden dette er *GPS* kan vi aldri unngå feilmargin i posisjonering helt. Ved nærmere undersøkelse av M8, kan vi se at mange av begrensningene har blitt

forbedret på nyere versjoner, som i M10. Der M8 kun kan håndtere 3 konstellasjoner av satellitter for posisjonering, kan M10 håndtere 4 konstellasjoner av satellitter samtidig. Dette øker da presisjonen og senker usikkerheten med stor margin under krevende forhold. De nye M10 mottakerne har også økt stabilitet på forbindelse til satellitter, der vi ofte opplevde å miste *GPS* signal i veldig korte øyeblikk.

Table 1: Urban environment accuracy measurements

	Singapore urban canyon			Chicago urban canyon		
	4 GNSS	Δ	3 GNSS	4 GNSS	Δ	3 GNSS
2D position error (m), 50%	2.54	-5%	2.66	1.93	-21%	2.45
2D position error (m), 68%	3.6	-30%	5.14	3.79	-15%	4.44
2D position error (m), 95%	13.56	-35%	20.87	35.5	-14%	41.43
2D position error (m), 100%	27.28	-72%	98.57	122.6	-18%	148.9

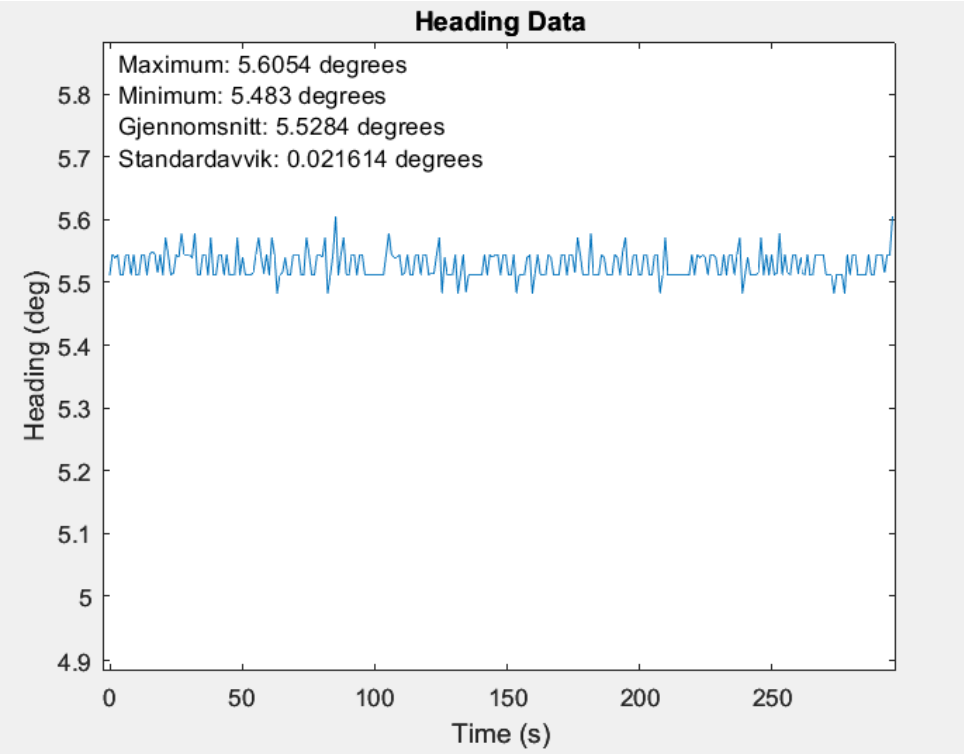
Figur 34: Tabell over feilmarginer av GPS i Urban Canyon [24]

Som vi kan se vi kan se i figuren over, er data innhentet i et krevende miljø, som er *Urban Canyon*, oversatt til «urban dal», som i en by med skyskrapere. [24] I figuren over vises resultatene i form av CEP, som er *Circular Error Probable*. Har vi en CEP50%, så måler vi diameteren til kun 50% av de nærmeste treffene til senter av alle treffene. Og måler vi så CEP100% så er det den største diameteren til alle punktene, og vi får dermed med alle de ytterste treffene med i målingen. Vi kan lese av i figuren at presisjonen målt med CEP50% at feilmarginen ikke er så stor målt i meter. Dersom man evaluerer målingen med CEP100% kan vi se at nøyaktigheten er redusert til en tredjedel. Vi kan ikke bytte fra M8 til M10 i denne bacheloren, men vi kan lære av å lese av hvilke svakheter M8 har, og hva den har som fungerer, slik at vi da kan bruke dette som underlag for hvordan dette kan brukes for å skape en relativt nøyaktig posisjonering. Vi har som sagt tidligere valgt å bruke *Sensor Fusion*, ved å bruke *GPS* og *IMU* data til å estimere posisjon, vi kan dermed vektlegge *GPS* signalet over *IMU*, ettersom *GPS* er mer presist enn *IMU*, men over tid så vil feil marginen i *GPS* øke, derfor vil vi bruke *IMU* data til å stabilisere denne feilmarginen til et minimum. Vi anbefaler at neste bachelorgruppe kjøper inn ny GPS, eller DGPS, for å redusere feilmarginen i lokaliseringen drastisk, slik at autodronen ikke blir for avhengig av andre sensorer for å kunne lokalisere seg selv.

9.3 Verifisering av magnetometer

Et magnetometer er en sensor som brukes til å måle magnetfeltet i sine omgivelser. Målingene forteller oss hvor sterkt magnetfeltet er, og i hvilken retning de peker. Derfor kan man sette det opp slik at magnetometeret kan brukes som et kompass.

For å få en bedre forståelse over hvor god mulighet vi har til å blant annet estimere vår egen orientering, samt å lokalisere bøyene i vårt koordinatsystem, gjennomførte vi et eksperiment for å se hvor stabile målinger vi fikk fra magnetometeret. Dette gjorde vi ved å plassere dronen stasjonert på land, og ved hjelp av et Python-script logget vi målingene fra magnetometeret hvert sekund. Siden dronen står stasjonært på land og ikke er i bevegelse vil vi på denne måten kunne se hvor stabile målinger vi får. Grafen under viser loggdata av magnetometeret da dronen var stasjonær på land. Eksperimentet er gjennomført over en periode på 5 minutter, der vi logget data fra magnetometeret hvert sekund.



Figur 35: Graf som viser Heading fra Magnetometer over tid (Privat)

Resultatet viser at magnetometeret kan gi oss stabile målinger som oppfyller kravene våre for tiltenkt formål.

10 Prosedyrer

Motivasjonen vår for å skrive en egen del om arbeidet vårt ved å lage prosedyrer, stammer fra at vi i startfasen av oppgaven, selv opplevde hvor viktig dette er. Ettersom sjødronen er et komplekst system der det har blitt uført flere forskjellige bacheloroppgaver opp gjennom årene, der samtlige har drevet med testing, utvikling, prototyping, prøving og feiling. Dette har ført til at hele systemet har blitt ganske rotete å sette seg inn i fra blanke ark. I tillegg til dette møtte vi som tidligere nevnt en del utfordringer siden noe maskinvare hadde blitt ødelagt mens den stod lagret. Noe som gjorde det enda vanskeligere å sette seg inn i, da det i tillegg var store problemer knyttet til å få sikkerhetskopiene til å fungere. Fra våre egne erfaringer i arbeidslivet, har vi i tillegg opplevd at standardiserte prosedyrer er en essensiell faktor for effektive arbeidsdager. Dessuten brukte vi selv våre instruksjoner og prosedyrer flittig for å gjennomføre visse arbeidsoppgaver, noe som resulterte i at vi var mer effektive. Et annet mål er også å gjøre det lettere for studenter som i ettertid skal jobbe med dronen, slik at de får en bedre innføring og en mer effektiv operasjon av dronen enn det vi gjorde.

10.1 Teori Prosedyrer

Mange av dagens bedrifter benytter seg av såkalte «Lean-verktøy». Disse verktøyene er metoder som bedriften benytter seg av for å få en mer produktiv bedrift, og de fokuserer på å finne områder bedriften kan forbedre seg på, og iverksette tiltak for å kontinuerlig forbedre seg. Det finnes mange ulike former og teknikker av Lean verktøy, og noen av de vanligste er blant annet 5S, Lean Six Sigma og Kaizen [25]. Vi skal ikke gå nærmere inn på disse Lean-verktøyene, men vi vil påpeke at de fleste av disse har til felles at de i en eller annen form innebærer å lage standardiserte prosedyrer og arbeids instruksjoner.

Det finnes en rekke grunner til hvorfor man bør lage gode arbeidsprosedyrer. Disse er blant annet:

- Standardisering vil øke effektivitet.
- Konsekvent sørge for at alle har tilgang til en metode som gir resultat
- Gjør det lettere for nye å sette seg inn i, og utføre arbeidet

- Standardisering hjelper på å holde oversikt, og sørge for at alle steg i prosessen blir gjennomført
- Standardisering legger til rette for å ytterligere forbedre og effektivisere arbeidsoppgavene.

[26], [27, pp. 319-340], [28]

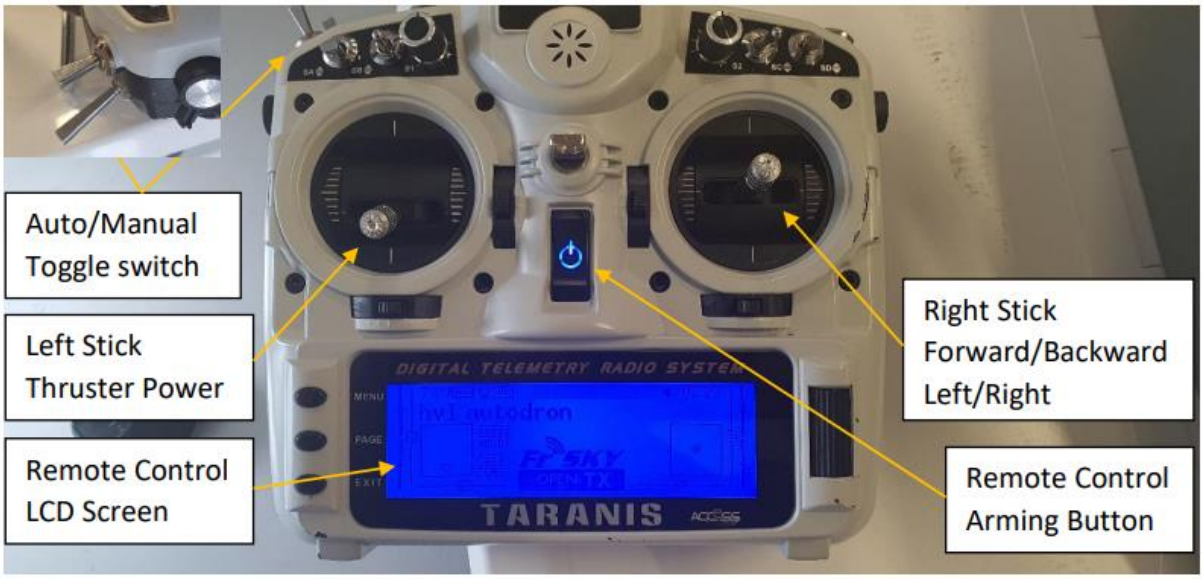
I en casestudie gjennomført ved et bilchassis produksjonslinje undersøkte de effekten av å innføre standardiserte arbeidsprosedyrer [29]. Resultatet viste blant annet at man reduserte tidsbruken med 15% og at nedetiden ble redusert med 9.6%. Her kan man trolig også legge til grunn at personellet tidligere hadde erfaring med arbeidet. Dessuten er arbeidet ved en produksjonslinje stort sett forholdsvis repetitivt, så det er god grunn til å tro at gode arbeidsprosedyrer vil ha en større effekt når man skal utføre oppgaver som er ukjente eller som man gjør sjeldent.

10.2 Arbeidet med å utarbeide dokumentasjon og prosedyrer

Ved starten av oppgaven innså vi fort at dokumentasjonen til dronen var mangelfull. Ikke bare hadde vi problemer med tanke på å legge inn sikkerhetskopi av softwaren, da dette var i feil format, men til og med de minste ting som det å koble seg opp til dronen å kjøre programmer for første gang viste seg være tidkrevende grunnet de manglende prosedyrene og dokumentasjon. Foruten å måtte opparbeide oss bedre kjennskap til både Python-programmering og ROS, måtte vi også bruke mye tid på å sette oss inn i sjødronens virkemåte og oppbygging. Vi kom kjapt til en konklusjon om at dersom det hadde foreligget gode prosedyrer fra starten av, hadde dette spart oss for mye tid. Derfor har vi helt i fra begynnelsen jobbet systematisk med å lage prosedyrer, «formel-hefter», instruksjonsvideoer og dokumentasjon underveis.

Måten vi utførte dette på er at det i starten som regel er en fase der man må lese seg opp på mulige løsninger, og gjøre seg kjent med arbeidsoppgaven. Deretter tester man seg frem, og finner en løsning som fungerer. Samtidig som man tester seg frem noterer vi ned punkter

som er viktig å få med. Når man så har oppnådd ønsket utfall, dokumentere vi de ulike stegene vi tok for å få det til. Under dokumenteringen er det viktig å bruke presist språk, gjerne ha med utklipp/bilde/figurer og å være kort og konsis. Slik sørger man for at man får en prosedyre som er enkel og forståelig, dette er tross alt det viktigste, da en av hovedformålene med prosedyrene er at den skal være enkel og oversiktlig.



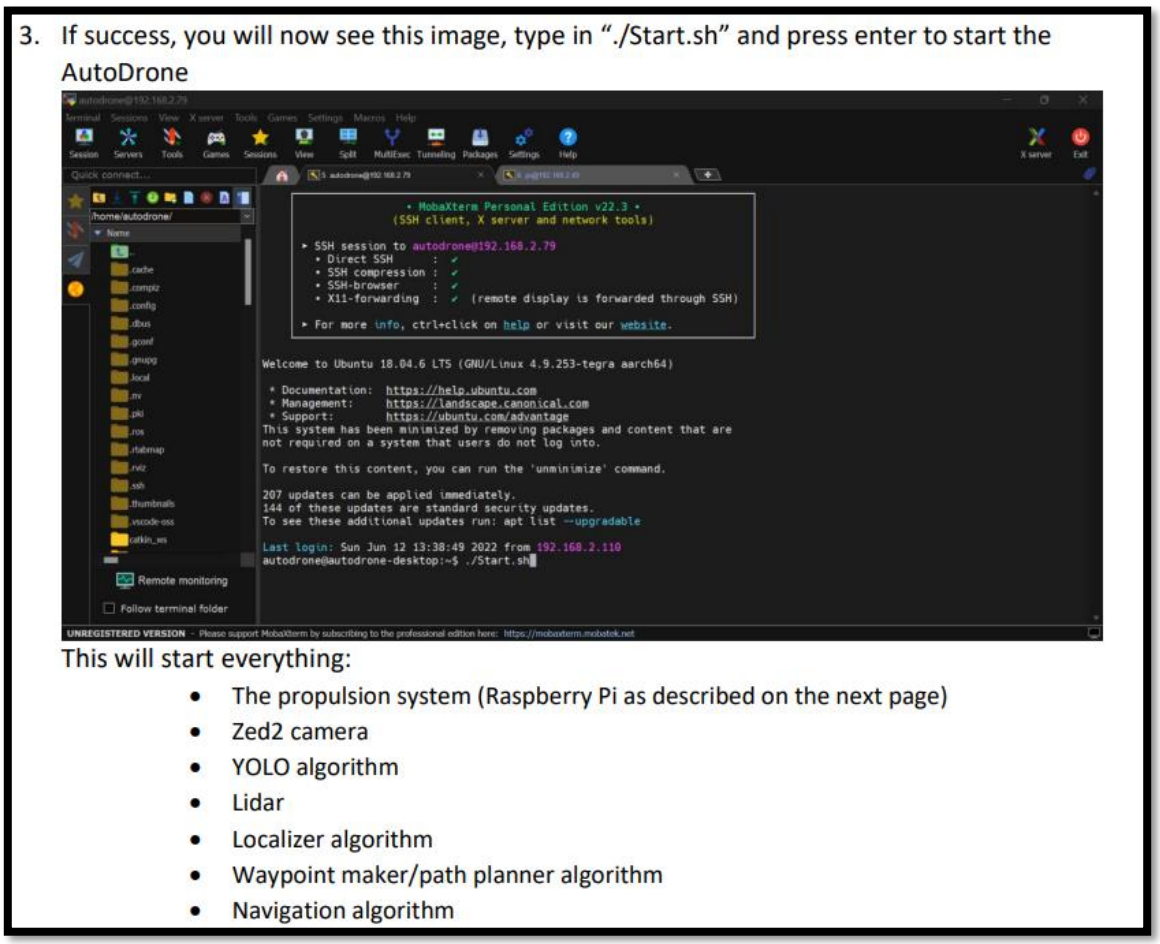
Figur 36: Skjermdump av Oversikt over fjernkontroll med funksjoner (Privat)

Dersom en prosedyre er omfattende, så får vi personen som ikke skrev prosedyren til å gjennomføre arbeidsoppgaven ved å følge dokumentet. Dette gjør vi for å kvalitetssikre at prosedyren er lett å følge, oppnår ønsket resultat og dobbelt sjekker at ingen steg er glemt. Slik får vi også tilbakemeldinger på prosedyren om ting som kan være forvirrende, som bør forenkles og ting som bør presiseres nærmere.

Når prosedyren er utviklet og kvalitetssikret, så har vi dessuten tatt i bruk prosedyrene når vi skal utføre arbeidsoppgaven senere. Det som er så bra med et godt grunnlag og utgangspunkt er at man ikke bare sparer tid, men dersom man finner en bedre løsning er det veldig lett å gå tilbake å gjøre endringer i dokumentet. På denne måten legger vi også opp til å kontinuerlig forbedre dokumentasjonen vår.

Vi har laget ulike former for instruksjoner, manualer og prosedyrer. De fleste er i form av et dokument som stegvis tar for seg hvordan man skal gjøre en viss arbeidsoppgave. Dessuten har vi blant annet også laget et «jukse-ark» (Cheat sheet) med nyttige kommandoer man

bruker ofte mens man opererer dronen, slik at man kan kopiere kommandoer inn i terminalen, og som inneholder kort informasjon om hva kommandoen gjør. I tillegg har vi også benyttet videoformatet, der vi har lagd en instruksjonsvideo på hvordan man implementerer den ferdigtrente vekt-filen fra Yolo treningen i ROS. Vi har også laget en manual som tar en gjennomgang av dronens oppbygning og virkemåte, samt viser hvordan man skal gå frem for å koble seg til dronen via SSH, slik at man kan kommunisere med dronen via PC. Her går vi også gjennom hvordan man skal starte opp dronen slik at man kan styre den via PC eller fjernkontroll, og hvordan man kan sette opp nettverket.



Figur 37: Skjermdump fra Oppstartsprosedyre (Privat)

10.3 Resultat

Arbeidet vårt har resultert i seks dokumenter som beskriver både virkemåten til autodronen, og fremgangsmetoder for viktige oppgaver. I tillegg har vi laget et *Cheat Sheet* som inneholder viktige kommandoer man bruker under operasjon av dronen. Disse finner du som vedlegg til bachelorrapporten. Vi har også laget en instruksjonsvideo som viser hvordan man implementerer en egendefinert objektdeteksjon i ROS, samt modifisert en Google Colab notatbok slik at den fungerer med Yolo-V3-Tiny.

Vi har selv brukt vår egen dokumentasjon flittig gjennom hele arbeidet, noe som har resultert i færre feil og økt effektivitet. Dessuten vil dokumentasjonen være til stor hjelp for studenter som på et senere tidspunkt eventuelt skal fortsette arbeidet på autodronen.

Link til instruksjonsvideo: https://youtu.be/0oViMZ1UVak

Link til nedlastbar Colab Notatbok: https://github.com/Njalmenning/Yolo_filer

11 Konklusjon

Denne bacheloroppgaven har vært en spennende, lærerik og krevende prosess for gruppen. Selve målet med oppgaven har vært å forbedre sjødronen slik at den har gode sjanser for å prestere bra under AutoDrone 2023 konkurransen. Vi ble tidlig overveldet av tyngden på oppgaven, spesielt med tanke på oppstartsproblemene og den manglende dokumentasjon. Fra starten av brukte vi mye tid på å gjøre oss kjent med alle de forskjellige komponentene i autodronen, og ikke minst alle de forskjellige systemene den kjører.
Tidlig i prosessen konkluderte vi med at vi ville fokusere på å forbedre objektdeteksjonen og lokaliseringen. I tillegg ville vi lage gode prosedyrer og dokumentasjon som beskriver hvordan man opererer autodronen, og hvordan man benytter seg av de ulike programvarene.
Vi ble tidlig møtt av defekte komponenter på autodronen som følge av fuktskader, som vi mistet et par uker på å reparere. En positiv konsekvens av dette er at vi ble enda bedre kjent med dronens virkemåte. Dessuten fikk vi forbedret vanntettheten i skroget, samt 3D-printet kapslinger til utsatte komponenter.

Under arbeidet med objektdeteksjonen fant vi flere mangler i det tidligere datasettet, og vi satte tidlig i gang arbeidet med å lage nytt datasett for trening av objektdeteksjons algoritmen. Resultatet av dette viste høy grad av suksess ved deteksjon av bøyer på opptil 20 meter.

Lokaliseringsalgoritmen fra forrige bachelor gruppe var basert på direkte avlesning av koordinater fra *GPS*, dette viste seg å fungere suboptimalt. Derfor modifiserte vi algoritmen slik at den benytter seg av fusjonering av *GPS* og *IMU-data*.

Samtidig som vi satte oss inn i ukjente komponenter og virkemåte på autodronen, så tok vi oss tid underveis til å dokumentere prosessen og lage prosedyrer for håndtering av dronen. På denne måten har vi gjort det enklere for eventuelt senere bachelorgrupper som skal fortsette arbeidet på autodronen.

Når det gjelder kildekode for algoritmene så har vi brukt en del kommentarer i koden, slik at

koden skal være mer leselig og oversiktlig. Vi ser på autodronen sin autonomitet som en del begrensende på grunnlag av de komponenter den har, men det skal være mulig å gjennomføre konkurransen med et relativt godt resultat.

Bachelorgruppen vil fortsette å utvikle navigasjons algoritme frem mot konkurransen.

12 Referanser

[1] AutoDrone, «www.AutoDrone.no,» [Internett]. Available: https://www.autodrone.no/competition.

[2] RaspberryPi. [Internett]. Available: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/.

[3] Emlid. [Internett]. Available: https://docs.emlid.com/navio2/.

[4] N. Developer. [Internett]. Available: https://developer.nvidia.com/embedded/learn/get-started-jetson-xavier-nx-devkit.

[5] StereoLabs. [Internett]. Available: https://www.stereolabs.com/zed-2/.

[6] «Google Colab,» [Internett]. Available: https://colab.research.google.com/. [Funnet 16 05 2023].

[7] [Internett]. Available: https://pjreddie.com/darknet/yolo/. [Funnet 10 05 2023].

[8] «ROS,» [Internett]. Available: https://www.ros.org/. [Funnet 10 05 2023].

[9] [Internett]. Available: https://www.mygreatlearning.com/blog/yolo-object-detection-using-opencv/. [Funnet 14 05 2023].

[10] 26 April 2023. [Internett]. Available: https://github.com/leggedrobotics/darknet_ros.

[11] J. D. S. G. R. &. F. A. Redmon, «You Only Look Once: Unified, Real-Time Object Detection.,» https://doi.org/10.48550/arxiv.1506.02640, 2015.

[12] J. &. S. W. Kaur, «Tools, techniques, datasets and application areas for object detection in an image: a review.,» Multimedia Tools and Applications, 81(27) , 38297–

38351., 2022.

[13] K. P. A. Ł. J. O. B. F. P. &. R. K. Chachuła, «Combating noisy labels in object detection datasets.,» 2022.

[14] L. Z. J. H. A. J. A.-D. A. D. Y. A.-S. O. S. J. F. M. A. A.-A. M. &. F. L. Alzubaidi, «Review of deep learning: concepts, CNN architectures, challenges, applications, future directions.,» Journal of Big Data, 8(1), 53-53., 2021.

[15] C. B. H. M. F. B. N. D. &. G. Z. W. Murthy, «Investigations of Object Detection in Images/Videos Using Various Deep Learning Techniques and Embedded Platforms—A Comprehensive Review.,» Applied Sciences, 10(9), 3280, 2020.

[16] L. J. Myhre og M. Eide, «BO22EH-02 Autonom Sjødrone,» Haugesund, 2022.

[17] A. C. M. &. O.-E. S. Dominguez-Sanchez, «A New Dataset and Performance Evaluation of a Region-Based CNN for Urban Object Detection.,» Electronics (Basel), 7(11), 301., 2018.

[18] [Internett]. Available: https://public.roboflow.com/object-detection/self-driving-car. [Funnet 10 05 2023].

[19] S. Y. Z. Z. M. L. H. S. S. M. M. &. Z. L. Zhou, «A Method to Automatic Create Dataset for Training Object Detection Neural Networks.,» IEEE Access, 10, 1–1., 2022.

[20] [Internett]. Available: https://www.stereolabs.com/developers/release/. [Funnet 25 04 2023].

[21] «Roboflow,» [Internett]. Available: https://roboflow.com/annotate. [Funnet 01 05 2023].

[22] [Internett]. Available: https://www.youtube.com/watch?v=10joRJt39Ns. [Funnet 21 05 2023].

[23] P. Corke, «Robotics, Vision and Control: Fundamental algorithms in Matlab (2nd ed.),» Springer International Publishing, 2017.

[24] ublox. [Internett]. Available: https://www.u-blox.com/en/blogs/tech/gps-accuracy-four-gnss-constellations.

[25] J. &. S. S. K. Bhamu, « Lean manufacturing: literature review and research issues.,» *International Journal of Operations & Production Management,,* Vol. %1 av %234(7),, p. 876–940., 2014.

[26] B. K. S. N. B. H. P. M. C. M. &. H. L. Stecher, «Evidence for Effects of TPS/Lean Manufacturing on Production and Workers.,» *Organizational Improvement and Accountability.,* 2004.

[27] J. Nicholas, « Lean Production for Competitive Advantage (2nd ed.).,» Productivity Press., 2018.

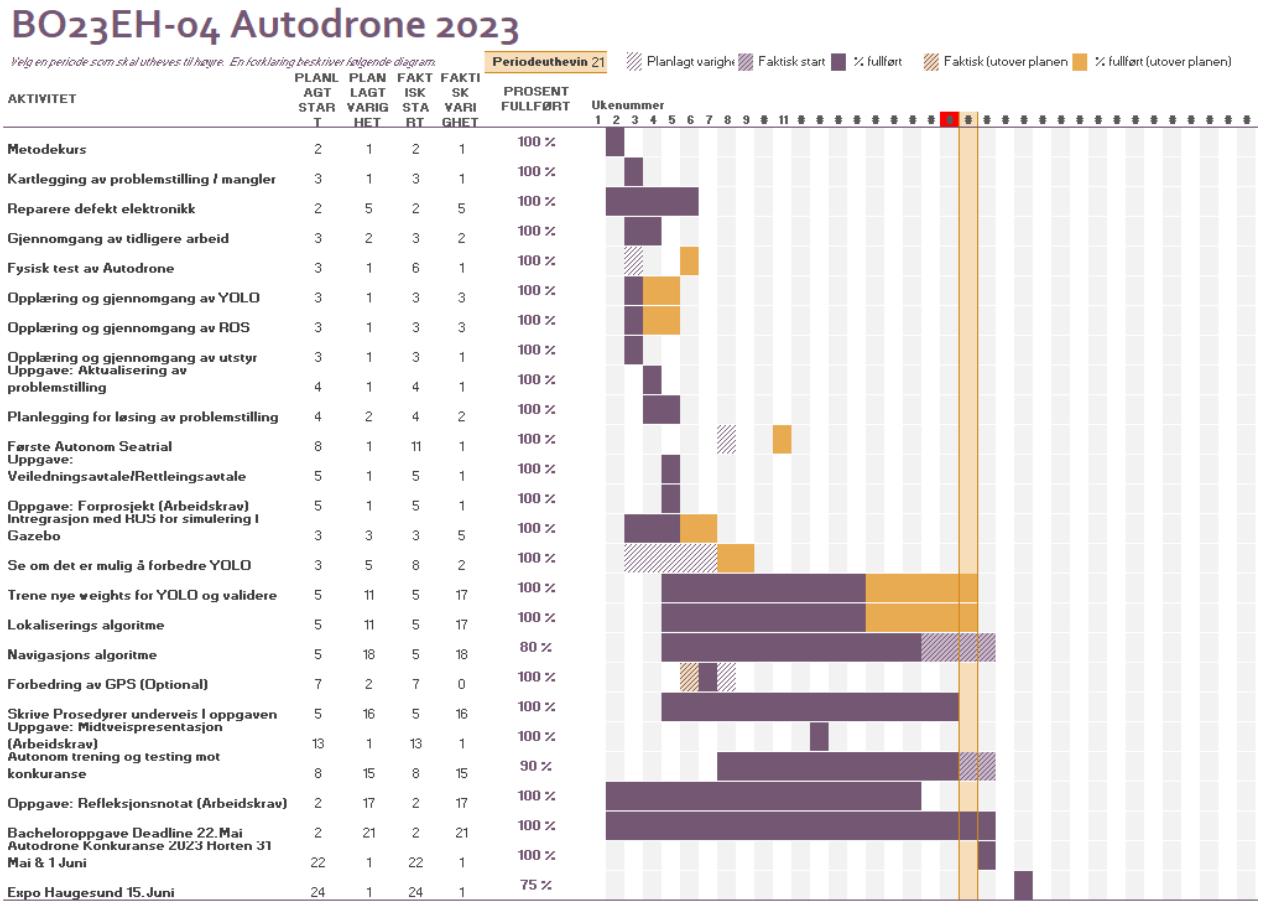
[28] B. Brandall, 30 Mai 2018. [Internett]. Available: https://www.process.st/process-standardization/. [Funnet 4 Mai 2023].

[29] J. C. V. G. C. I. &. M. V. d. C. Fin, «Improvement based on standardized work: an implementation case study.,» *Brazilian Journal of Operations & Production Management,* vol. 14(3), p. 388–395., 2017.

Figurliste

13 Vedlegg

13.1 Fremdriftsplan



Figur 38: Fremdriftsplanen til Autodrone 2023 (Privat)

13.2 Timeliste Jari Inermo

Timeliste Jari Inermo		
Dato	**Timer**	**Beskrivelse**
11.01.2023	4	Metodekurs
12.01.2023	4	Metodekurs
13.01.2023	10	Metodekurs + feilsøking, bli kjent med drone + overføre filer fra Magnus
14.01.2023		
15.01.2023	8	Levere Drone til vårens lokasjon, gå gjennom elektriske komponenter, prøve lage tegning.
16.01.2023	8	Veileder møte + Teste om Raspberry Pi & Navio2 er defekt
17.01.2023	10	Bytte Raspberry Pi + Fact.Reset RUT, pluss sette opp på ny + Teste aktivisering av drone
18.01.2023	10	1st SeaTrials(med fjernkontroll) + Backup
19.01.2023	5	ROS kurs
20.01.2023	8	Skrive "Aktualisering av problemsstilling" + ROS kurs
21.01.2023		
22.01.2023		
23.01.2023	8	ROS kurs
24.01.2023	8	Yolo Kurs Udemy
25.01.2023	8	Yolo Kurs Udemy
26.01.2023	8	Bygge krets for Arduino Nano for laterne lys, reverse engineere gammel krets
27.01.2023	8	Programmere Arduino Nano for laterne lys, med reverese engineering av kode
28.01.2023		
29.01.2023		
30.01.2023	8	Programmere ferdig krets for laterne lys
31.01.2023	8	Montere og feilsøke Arduino Nano til styring av lanterne lys
Januar Total:	123	
01.02.2023	8	Forsudierapport
02.02.2023	12	Yolo Kurs Udemy
03.02.2023	8	Feilsøke og teste ny Arduino Nano, og modusvelger for lanterne lys
04.02.2023		
05.02.2023		
06.02.2023	12	Forstudierapport innlevering + veiledningavtale innlevering
07.02.2023	0	Jobb hele dagen
08.02.2023	8	Ta Backup av dronen, prøve å fil overføre via Moxba
09.02.2023	10	Skrive Sponsor avtale + Presentasjon for veiledningsmøte + Veiledningsmøte
10.02.2023	8	Finne ut hvordan en trener opp weights for Yolov3-Tiny
11.02.2023		
12.02.2023		
13.02.2023	0	Bortreist på jobb i Ænes

14.02.2023	8	Ta Backup av dronen, med Image generator av SD kort
15.02.2023	8	modellere kapsling for Arduino og Raspberry pi I akter koblingsboks + starte 3d print
16.02.2023	8	3d printing fortsetter
17.02.2023	8	Montere kapsling på Raspberry Pi og Arduino
18.02.2023		
19.02.2023		
20.02.2023	8	Feilsøke Raspberry Pi, da det ser ut som SD kort er blitt korrupt
21.02.2023	0	Syk
22.02.2023	0	Syk
23.02.2023	0	Syk
24.02.2023	8	Prøve å opprette ny SD boot disk for Raspberry Pi, da det ser ut som Image filer ikke fungerer
25.02.2023		
26.02.2023		
27.02.2023	8	Prøve å lage ny SD boot disk for raspberry Pi fra start
28.02.2023	8	Fortsettelse med å prøve å lage ny boot disk med ROS og Navio
Februar Total:	130	
01.03.2023	0	Bortreist på jobb Bømlo
02.03.2023	0	Bortreist på jobb Bømlo
03.03.2023	0	Bortreist på jobb Bømlo
04.03.2023		
05.03.2023		
06.03.2023	8	Få dronen oppe å gå igjen etter korrupsjon av navigasjons boot disk
07.03.2023	8	Verifisering av dronen går som den skal igjen etter oppretting av ny boot disk for navigasjon
08.03.2023	8	Sea Trial med fjernkontroll, gjøre klar blåser, ta bilder av blåser + annotering av bilder
09.03.2023	8	Annotering av bilder, plus trening av weights mot rød og gul bøye, pluss test.
10.03.2023	8	Gå gjennom algoritme for navigasjon, og klargjøre dette for autonomi test neste uke.
11.03.2023		
12.03.2023		
13.03.2023	8	Skrive Bachelor rapport
14.03.2023	8	Skrive ny startbeskrivelse for dronen "HVL AUTODRONE STARTBESKRIVELSE 2023"
15.03.2023	8	Lage opptak av nye bøyer, få de konvertert til png
16.03.2023	8	Laste opp png filer til Roboflow og starte annotering
17.03.2023	8	annotere datasett
18.03.2023		
19.03.2023		
20.03.2023	8	ferdigstille annotering for datasett
21.03.2023	8	Klargjøre midtveispresentasjon

22.03.2023	8	Midtveispresentasjon
23.03.2023	0	Pitch Innovasjon og Systemtenking ING303
24.03.2023	0	Jobb hele dagen
25.03.2023		
26.03.2023		
27.03.2023	8	Sette opp PC med Ubuntu 20.04 LTS med ROS og kildekode, og SSH terminal med X11-Forwarding
28.03.2023	8	Jobbe med lokalisering algoritme
29.03.2023	8	Jobbe med lokalisering algoritme
30.03.2023	8	Jobbe med lokalisering algoritme
31.03.2023	8	Jobbe med lokalisering algoritme
Mars Total:	144	
01.04.2023		
02.04.2023		
03.04.2023	0	Påskeferie/Sykdom
04.04.2023	0	Påskeferie/Sykdom
05.04.2023	0	Påskeferie/Sykdom
06.04.2023	0	Påskeferie/Sykdom
07.04.2023	0	Påskeferie/Sykdom
08.04.2023		
09.04.2023		
10.04.2023	0	Påskeferie/Sykdom
11.04.2023	0	Påskeferie/Sykdom
12.04.2023	0	Påskeferie/Sykdom
13.04.2023	8	Skrive I bachelorrapporten
14.04.2023	8	Skrive I bachelorrapporten
15.04.2023		
16.04.2023		
17.04.2023	8	Feilsøking av dårlig WIFI signal på dronen
18.04.2023	8	Undersøkelse av hvorfor GPS er så unøyaktig
19.04.2023	8	Skrive prosedyrer
20.04.2023	8	Skrive prosedyrer
21.04.2023	8	Feilsøking autodronen, da elektronikken er ustabil
22.04.2023		
23.04.2023		
24.04.2023	8	Skrive I bachelorrapporten
25.04.2023	8	Skrive I bachelorrapporten
26.04.2023	8	Lese I dokumentasjon på ZED2 kamera + ROS
27.04.2023	8	Lese I dokumentasjon på ZED2 kamera + ROS
28.04.2023	8	Skrive I bachelorrapporten
29.04.2023		
30.04.2023		
April Total:	96	
01.05.2023	8	Ta nye bilder med autodronen, for å genrere nytt datasett
02.05.2023	8	Annotere bilder av bøyer

03.05.2023	0	Syk
04.05.2023	8	Trene nye weights til YOLO
05.05.2023	8	Validere nye weights på dronen
06.05.2023		
07.05.2023		
08.05.2023	8	Logging av GPS drift
09.05.2023	8	Testing av forskjellige lokasjons algortimer med bruk av IMU
10.05.2023	8	Testing av forskjellige lokasjons algortimer med bruk av IMU
11.05.2023	8	Logging av magnetometer
12.05.2023	8	Testing av Sensor Fusion av GPS + IMU
13.05.2023		
14.05.2023		
15.05.2023	0	Begravelsen til min farmor
16.05.2023	0	Begravelsen til min farmor
17.05.2023	8	Teste lokalisering algoritme
18.05.2023	8	Jobbe mot innlevering av bachelorrapporten
19.05.2023	8	Jobbe mot innlevering av bachelorrapporten
20.05.2023	8	Jobbe mot innlevering av bachelorrapporten
21.05.2023	8	Jobbe mot innlevering av bachelorrapporten
22.05.2023	8	Jobbe mot innlevering av bachelorrapporten
23.05.2023	0	Sette autodronen klar for Autodrone 2023 konkurransen
24.05.2023	0	Sette autodronen klar for Autodrone 2023 konkurransen
25.05.2023	0	Sette autodronen klar for Autodrone 2023 konkurransen
26.05.2023	0	Sette autodronen klar for Autodrone 2023 konkurransen
27.05.2023	0	Sette autodronen klar for Autodrone 2023 konkurransen
28.05.2023	0	Sette autodronen klar for Autodrone 2023 konkurransen
29.05.2023	0	Sette autodronen klar for Autodrone 2023 konkurransen
30.05.2023	0	Testdag Autodrone 2023
31.05.2023	0	Autodrone 2023
01.06.2023	0	Autodrone 2023
02.06.2023	0	Autodrone 2023
Mai/Juni Total:	120	
Sum timer Total:	613	Timer Totalt

13.3 Timeliste Njål Almenning

Timer Njål Almenning		
Dato	**Timer**	**Beskrivelse**
11.01.2023	4	Metodekurs
12.01.2023	4	Metodekurs
13.01.2023	8	Metodekurs og ta ut dronen/fikse mygla koblingsboks
14.01.2023	4	Opplæring Python
15.01.2023	2	Opplæring ROS
16.01.2023	8	Veiledningssamtale. Oppkobling av dronen, skifte Raspi, hovedstrømsbryter
17.01.2023	8	Sette opp rut. Opprette kommunikasjon av systemet
18.01.2023	6	Videre testing. Første "Seatrail".
19.01.2023	8	Utarbeide fremdriftsplan, aktualisering av problemstilling
20.01.2023	8	Opplæring ROS
21.01.2023		
22.01.2023		
23.01.2023	8	Opplæring Yolo
24.01.2023	8	Opplæring Yolo
25.01.2023	8	Testing av Yolo i ROS og Gazebo
26.01.2023		
27.01.2023	8	Lage datasett av simulerings bøyer i Roboflow. Annotere bilder
28.01.2023		
29.01.2023		
30.01.2023	8	Opplæring i Google Colab, og trenig av datasett i Google Colab med lånt GPU
31.01.2023	8	Montere og feilsøke Arduino Nano til styring av lanterne lys
Januar Total:	108	
01.02.2023	4	Forstudierapport
02.02.2023		
03.02.2023	8	Feilsøke og teste ny Arduino Nano, og modusvelger for lanterne lys
04.02.2023		
05.02.2023		
06.02.2023	8	Forstudierapport
07.02.2023		
08.02.2023		
09.02.2023	8	Veiledningssamtale. Oppkobling av dronen, skifte Raspi, hovedstrømsbryter
10.02.2023	8	Finne ut korleis ein genererer yolo weights for Yolov3-tiny
11.02.2023		
12.02.2023		
13.02.2023		
14.02.2023	8	Lage prsedyre og google colab notebook for trening av Yolo
15.02.2023	8	Trene Yolo i googe colab

16.02.2023	8	Finne ut korleis ein kan skalere ned bilete som blir sendt til skjerm fra dronen
17.02.2023	8	Monter kapsling på raspi og nano.
18.02.2023		
19.02.2023		
20.02.2023	8	Implimentere custom weights i ROS.
21.02.2023	8	Implimentere custom weights i ROS.
22.02.2023	8	Implimentere custom weights i ROS.
23.02.2023	8	Implimentere custom weights i ROS.
24.02.2023	8	Implimentere custom weights i ROS.
25.02.2023		
26.02.2023		
27.02.2023		Syk/Sykt barn
28.02.2023		Syk/Sykt barn
Februar Total:	108	
01.03.2023		Syk/Sykt barn
02.03.2023		Syk/Sykt barn
03.03.2023		Syk/Sykt barn
04.03.2023		
05.03.2023		
06.03.2023	8	Feilsøking på dronens boot disk
07.03.2023	8	Feilsøking på dronens boot disk
08.03.2023	8	Sea Trial med fjernkontroll, gjøre klar blåser, ta bilder av blåser + annotering av bilder
09.03.2023	8	Annotering av bilder, plus trening av weights mot rød og gul bøye, pluss test.
10.03.2023	8	Gå gjennom algoritme for navigasjon, og klargjøre dette for autonomi test neste uke.
11.03.2023		
12.03.2023		
13.03.2023	8	Skrive prosedyre for trening av weights til YOLO
14.03.2023	8	Skrive prosedyre for trening av weights til YOLO
15.03.2023	8	Lage opptak av nye bøyer, få de konvertert til png
16.03.2023	8	annotere datasett
17.03.2023	8	annotere datasett
18.03.2023		
19.03.2023		
20.03.2023	8	Annotere datasett med tre farge
21.03.2023	8	Lage midtveispresentasjon
22.03.2023	8	Midtveispresentasjon feilsøke i navigasjon.py
23.03.2023	0	Systemtenking og innovasjon
24.03.2023	8	Trening av weights med tre farger
25.03.2023		
26.03.2023		
27.03.2023	8	Lagd prosedyre for trening av Yolo

28.03.2023	8	Jobbe med lokalisering algoritme
29.03.2023	8	Jobbe med lokalisering algoritme
30.03.2023	8	Jobbe med lokalisering algoritme
31.03.2023	8	Jobbe med lokalisering algoritme
Mars Total:	152	
01.04.2023		
02.04.2023		
03.04.2023	8	Logging av GPS og IMU
04.04.2023	8	Logging av GPS og IMU
05.04.2023	8	Logging av GPS og IMU
06.04.2023	0	Påskeferie
07.04.2023	0	Påskeferie
08.04.2023		
09.04.2023		
10.04.2023	0	Påskeferie
11.04.2023	8	Skrive I bachelorrapporten
12.04.2023	8	Skrive I bachelorrapporten
13.04.2023	8	Skrive I bachelorrapporten
14.04.2023	8	Skrive I bachelorrapporten
15.04.2023		
16.04.2023		
17.04.2023	8	Feilsøking av dårlig WIFI signal på dronen
18.04.2023	8	Validering av YOLO weights
19.04.2023	8	Validering av YOLO weights
20.04.2023	8	Validering av YOLO weights
21.04.2023	8	Skrive prosedyre for YOLO
22.04.2023		
23.04.2023		
24.04.2023	8	Skrive I bachelorrapporten
25.04.2023	8	Skrive I bachelorrapporten
26.04.2023	8	Lese I dokumentasjon på ZED2 kamera + ROS
27.04.2023	8	Lese I dokumentasjon på ZED2 kamera + ROS
28.04.2023	8	Skrive I bachelorrapporten
29.04.2023		
30.04.2023		
April Total:	136	
01.05.2023	8	Ta nye bilder med autodronen, for å genrere nytt datasett
02.05.2023	8	Annotere bilder av bøyer
03.05.2023	8	Veiledningsmøte
04.05.2023	8	Trene nye weights til YOLO
05.05.2023	8	Validere nye weights på dronen
06.05.2023		
07.05.2023		
08.05.2023	8	Logging av GPS drift
09.05.2023	8	Testing av forskjellige lokasjons algortimer med bruk av IMU

10.05.2023	8	Testing av forskjellige lokasjons algortimer med bruk av IMU
11.05.2023	8	Logging av magnetometer
12.05.2023	8	Testing av Sensor Fusion av GPS + IMU
13.05.2023		Familietur Kroatia
14.05.2023		Familietur Kroatia
15.05.2023	0	Familietur Kroatia
16.05.2023	0	Familietur Kroatia
17.05.2023	0	Familietur Kroatia
18.05.2023	0	Familietur Kroatia
19.05.2023	0	Familietur Kroatia
20.05.2023		Familietur Kroatia
21.05.2023	8	Jobbe mot innlevering av bachelorrapporten
22.05.2023	8	Jobbe mot innlevering av bachelorrapporten
23.05.2023	0	Sette autodronen klar for Autodrone 2023 konkurransen
24.05.2023	0	Sette autodronen klar for Autodrone 2023 konkurransen
25.05.2023	0	Sette autodronen klar for Autodrone 2023 konkurransen
26.05.2023	0	Sette autodronen klar for Autodrone 2023 konkurransen
27.05.2023	0	Sette autodronen klar for Autodrone 2023 konkurransen
28.05.2023	0	Sette autodronen klar for Autodrone 2023 konkurransen
29.05.2023	0	Sette autodronen klar for Autodrone 2023 konkurransen
30.05.2023	0	Testdag Autodrone 2023
31.05.2023	0	Autodrone 2023
01.06.2023	0	Autodrone 2023
02.06.2023	0	Autodrone 2023
Mai/Juni Total:	96	
Sum timer Total:	600	Timer Totalt

13.4 Vedlegg lagt ved som egne filer

- Vedlegg 1 Autodrone Startbeskrivelse 2023 HVL.pdf
- Vedlegg 2 From SVO to AVI to PNG HVL.pdf
- Vedlegg 3 RUT950 Setup description HVL.pdf
- Vedlegg 4 Prosedyre for YOLO trening HVL.pdf
- Vedlegg 5 Sponsor Avtale Autodrone 2023 Haugesund HVL.pdf
- Vedlegg 6 Lokalisering algoritme Autodrone HVL.pdf
- Vedlegg 7 Arduino Modus Led kode.pdf