Universitetet i Sørøst-Norge
Fakultet for teknologi og maritime fag

Bacheloroppgave
2022

Gard Johan Tollefsen, Mustafa Abdi Hassan, Jafar Alami, Daniel Andreas Tokle Poverud

Fisketelling

Apparat for å telle og identifisere fisker i elver og i fisketrapper

Bachelor Prosjekt 2022

KONGSBERG JEGER- OG FISKERFORENING

Apparat for å telle og identifisere fisker i elver og i fisketrapper

Veileder Henning Gundersen
Gard J. Tollefsen, Daniel A.T. Poverud,
Jafar Alami, Mustafa Hassan
Gruppe 13, rom i-102

Abstrakt

Dette er bachelor prosjektet utført av Fisketrapp Observasjon, oppgaven er utgitt av Kongsberg Jeger- og Fiskerforening. Prosjektgruppen vår har fått i oppgave å lage et system som skal detektere, gjenkjenne og telle fisk som passerer fisketrapper på en rimelig måte. Oppgaven vår sammenkoblet maskinlæring, objekt gjenkjenning og mye maskinvare for å lage ett komplett system som følger kravene utgitt av Kongsberg Jeger- og Fiskerforening.

Innholdsfortegnelse

1 Introduksjon

Denne rapporten handler om prosjektoppgaven "Apparat for telling av fisker i elver og i fisketrapper" som vår prosjektgruppe har fått av Kongsberg Jeger- og Fiskeforening. Målet for oppgaven er å lage et system som kan konkurrere med systemene som allerede er ute på markedet. Meningen med denne rapporten er å få en systematisk forståelse av oppgaven og framgangsmetoden våres. Gruppen våres består av to elektroingeniør studenter og to dataingeniør studenter.

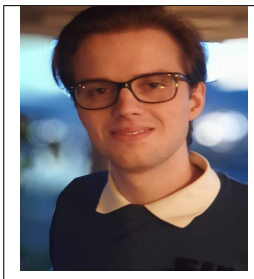
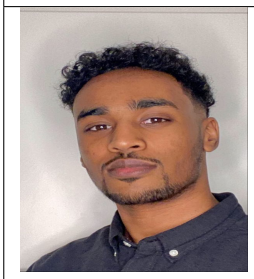
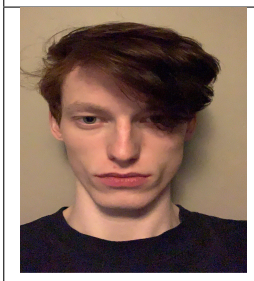
1.1 Problemstilling

Kongsberg Jeger- og Fiskeforening trenger et system som kan telle og identifisere de forskjellige fiskene i fisketrappene sine. Grunnen til at de trenger dette er siden det er stor usikkerhet over hvor mye fisk som faktisk bruker fisketrappene og de andre alternativene som allerede er på markedet er for kostbare, da blir vårt oppdrag å lage et billigere og effektivt alternativ til resten av markedet.

1.2 Oppgavebeskrivelse

Oppdraget går ut på å lage ett system som kan ta automatiske bilder og telle passerende fisker i fisketrappene. Her må vi innhente ett kamera som kan se under vann og i mørket. Det er essensielt at systemet er billig fordi dette systemet skal være mulig å reprodusere og selge videre til andre fiskeforeninger. Videre vil oppdragsgiver at vi skal bruke en kompakt datamaskin, denne må være koblet til ett batteri. Vi må også lage eller finne en programvare som kan identifisere fisken og være i stand til å telle fiskene for å se om fisketrappene er overhode effektive. Mengden av minne er opp til oss å bestemme, og vil avhenge av systemets kvalifikasjoner. Til sist vil oppdragsgiver at vi skal abonnere til en internettleverandør for å få trådløs internett til systemet. Dersom tiden tillater det utvikler vi en programvare som skille mellom abbor, gjedde og ørret.

1.3 Gruppeoversikt

	**Daniel Andreas Tokle Poverud** *Studieretning:* Elektroingeniør *Ansvarsområde:* Gruppeleder og Maskinvare ansvarlig *Tlf:* 979 17 874 *E-post:* Daniel.Poverud@gmail.com
	**Jafar Alami** *Studieretning:* Dataingeniør *Ansvarsområde:* Test og Programvare ansvarlig *Tlf:* 967 28 072 *E-post:* a.jafar0892@gmail.com
	**Mustafa Hassan** *Studieretning:* Dataingeniør *Ansvarsområde:* Programvare ansvarlig *Tlf:* 929 89 488 *E-post:* sa114@hotmail.co.uk
	**Gard Johan Tollefsen** *Studieretning:* Elektroingeniør *Ansvarsområde:* Dokument- og Maskinvare ansvarlig *Tlf:* 978 92 670 *E-post:* Gard.Johan.Tollefsen@gmail.com

1.4 Sensorer, veiledere og observatør

Gjennom prosjektperioden i dette avsluttende faget skal disse personene hjelpe oss med bachelor oppgaven, observere hva vi gjør for å hjelpe sensorene og dømme karakteren til hvert individ i gruppen.

Ekstern veileder og sensor
Navn: Åge Johan Skullestad
Stilling: Formann i Kongsberg Jeger- og Fiskerforening
Åge Johan Skullestad er oppdragsgiveren samt som han er vår eksterne veileder og sensor. Ekstern veileder skal gi oss kritiske tilbakemeldinger slik at Kongsberg Jeger- og Fiskerforening får produktet de ønsker. Dersom prosjektgruppen jobber i feil retning skal oppdragsgiver informere oss og veilede oss. Som ekstern sensor bedømmer han en stor del av karakteren vår.

Intern veileder og sensor:
Navn: Henning Gundersen
Stilling: Førsteamanuensis for USN
Henning Gundersen er vår interne veileder og sensor. Intern veileder kan komme med ideer om hvordan forbedre produktet, og tilbakemelding på hva som kan forbedres, men har ingen rådighet over gruppen. Som intern sensor bedømmer han en stor del av karakteren vår.

Observatør:
Navn: Karoline Moholth Mcclenaghan
Stilling: Universitetslektor for USN
Karoline Moholth er observatøren til prosjektet som skal hjelpe sensorsene vurdere gruppearbeidet vårt. Vi får ikke nødvendigvis direkte tilbakemeldinger av observatøren, men om vi har noen ekstra spørsmål angående prosjektet er mulig å oppsøke henne.

1.5 Aktiviteter

Denne seksjonen inneholder hvordan vi har planlagt og diktert utgivelse av ansvarsområder til hvert gruppemedlem, møter med både intern og ekstern veileder, oppsett av møtereferater og oppfølgingsdokumenter, og fri til øving i eksamens perioden.

1.5.1 Ansvarsområder

Alle gruppemedlemmer har blitt tildelt ansvarsområder, vi har gitt alle to ansvarsområder hver, ett unikt ansvarsområde og ett felles ansvarsområde. Dette er bare for å ha en generell oversikt over hvem som gjør hva, de har hoved kompetansen for sitt område men alle kommer til å jobbe med alt sammen dersom nødvendig. Vi mener at å jobbe på denne måten er en av de mest effektive og strukturerte arbeidsmetodene.

1.5.2 Møter

Møte med Intern veileder Henning Gundersen har vi blitt enig om at forekommer hver fredag kl 12:00 over Zoom eller i grupperommet våres på Innovasjonsloftet rom i-102.
Møte med Ekstern veileder og sensor Åge Skullestad har vi blitt enig om er hver onsdag kl 12:00 over Zoom.
Vi holder to korte uformelle gruppemøter hver uke, mandag og fredag kl 10:00, for å gjennomgå hva vi har gjort, hva vi skal gjøre videre, om noen trenger hjelp og med timeskriving.

1.5.3 Møtereferat

Vi har blitt enig om at møtereferater skal bli utfylt etter hvert enkelt møte som har forekommet med intern og ekstern veileder, og skal sendes til vedkommende innen 24 timer.

1.5.4 Oppfølgingsdokumenter

Vi skal lage oppfølgingsdokumenter hver fredag, etter morgen møte. Dokumentet skal inneholde en oppsummering av forrige uke, hva skal bli gjort neste uke, kort oppsummering av hvordan prosjektet går, kort oppsummering av kritiske aktiviteter og timeskriving.

1.5.5 Eksamens øving

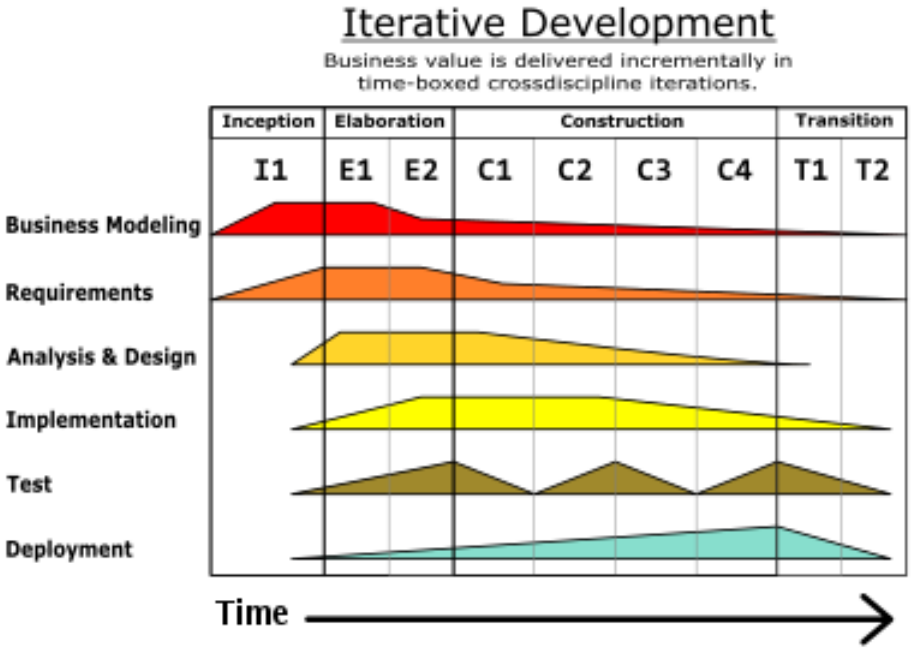
Innenfor Construction fasen av Unified Process har vi valgt to uker med halvveis prosjekt arbeid og halvveis eksamens øving, fra 11.03.2022 til 25.03.2022.

2 Prosjektplan

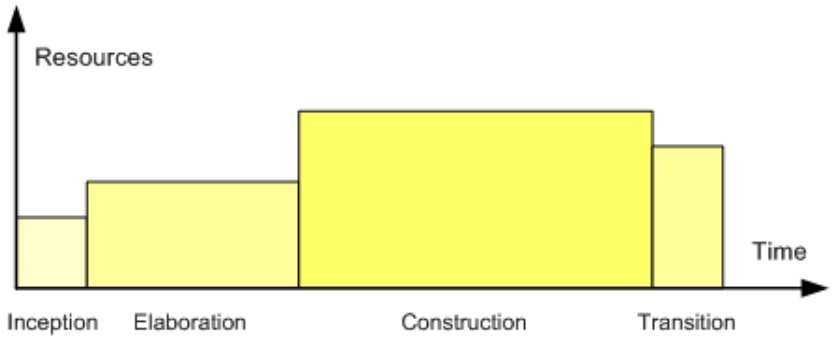
I denne seksjonen av rapporten skal vi forklare valget av prosjektmodell, fasene til prosjektmodellen og hvordan vi har planlagt å gå framover gjennom denne prosjekt perioden. Modellen er en massiv del av planen siden den definerer hvordan vi eventuelt skal få ett ferdig produkt. Disse valgene representerer gruppen våres arbeidsmetodikk og hvordan vi planlegger at prosjektet skal se ut.

2.1 Unified Process

Modellen vi har planlagt å bruke for hvordan vi arbeider gjennom prosjekt perioden er Unified Process. Når man velger modeller velger mann mellom evolusjonære modeller eller iterative modeller, Unified Process er en iterativ modell. Modellen består av fire faser hvor vi har to til tre iterasjoner per fase. Modellen er basert på å forstørre og fremme systemer, gjennom å arbeide parallelt med modellen sine disipliner, vist i Figur 1, gjennom flere iterasjoner, i flere faser av modellen. Disiplinene er business modeling, requirements, analysis and design, implementation, test og deployment.



Figur 1: Unified Process fase betydning. (1)



Figur 2: Unified Process fase betydning. (2)

I Unified Process legger mann minst vekt på Inception fasen ressurs vis og tids vis, i Elaboration bruker mann oftere en lengre periode for å finpusse slik at alt er klart til å bli gjort i Construction fasen som oftest tar lengst tid og mest ressurser, så blir systemet klargjort for utgivelse til interessenter i Transition fasen som bruker mye ressurser i en liten tidsramme som vist i Figur 2.

2.1.1 Inception fasen

Dette er den første fasen til Unified Process, her begynner vi å definere hvordan systemet skal bli realisert. I denne fasen starter vi å etablere de første versjonene av risiko identifisering, krav spesifikasjon, omfang definisjon, design valg, milepæler, tidsplan og kost estimater. Vi har bestemt at denne fasen skal gjøres i to iterasjoner som ender med første presentasjonen.

2.1.2 Elaboration fasen

I denne fasen er vi ment til å fange opp meste parten av systemets krav gjennom use cases for å lange en detaljert risiko analyse, risiko håndtering slik at vi kan redusere problemene som kan påvirke produktet. Inkludert i denne fasen er å bli ferdig med detaljerte versjoner av alt som ble gjort i Inception fasen. Vi har settet av to iterasjons perioder til Elaboration fasen.

2.1.3 Construction fasen

Construction er den mest detaljerte og omfattende fasen av Unified Process, som vi har delt inn i fire iterasjons perioder. I denne fasen skal vi hovedsakelig bli ferdig med analysis and design, implementation og test, for å lage et komplett system som er klart til å bli utlevert i Transition fasen. Systemet skal del og hel testes mye hver iterasjon innenfor denne fasen.

2.1.4 Transition fasen

Den endelige fasen av Unified Process, når vi har kommet til denne fasen burde meste parten av de seks disiplinene være hovedsakelig ferdig. Fordi her skal systemet endelig bli levert til brukerne, sammen med bruker opplæring og system konversjon hvis nødvendig. (3)

2.2 Mål

Målene til gruppen våres med prosjektet er å lage et system som tilfredsstiller læringskravene våre og gjør oss til bedre ingeniører, samtidig som det gir oss bedre karakter på bachelor oppgaven. Målet til oppdragsgiver er å få et system som kan brukes til å telle og identifisere fisk i feltet, og videre produksjon av systemet i etterkant av bachelor oppgaven våres.

2.2.1 Milepæler

I Tabell 1, ser vi milepælene til systemet vårt, altså de kritiske punktene som må være på plass for å få prosjektet gjort ferdig, samt å få god karakter. Vi har definert disse milepælene som fasene i Unified Process og alle tre presentasjonene gjennom prosjekt perioden.

Tabell 1: Prosjekt milepæler.

Dato	Milepæler
10.01.2022	Inception 1 start
07.02.2022	Inception 2 slutt og Første Presentasjon
07.02.2022	Elaboration 1 start
07.03.2022	Elaboration 2 slutt
07.03.2022	Construction 1 start
30.03.2022	Andre Presentasjon
09.05.2022	Construction 4 slutt
09.05.2022	Transition 1 start
20.05.2022	Transition 2 slutt
23.05.2022	Innlevering
25.05.2022	EXPO
03.06.2022	Tredje Presentasjon

2.2.2 Tidsplan

I Tabell 2, har vi ett skriftlig Gantt diagram som illustrerer hvor mange iterasjoner av hver fase som vi har valgt å gjøre og at hver iterasjon varer i 14 dager, unntatt den siste fasen har bare 7 dager per iterasjon.

Tabell 2: Overordnet fremdriftsplan

Fase	**Start**	**Ferdig**	**Viktige gjøremål**
Inception 1	**10.01.2022**	**24.01.2022**	Prosjekt plan Interresent analyse Risiko analyse Test plan Krav spesifikasjon Budsjett
Inception 2	**24.01.2022**	**07.02.2022**	Iterasjon Første Presentasjon
Elaboration 1	**07.02.2022**	**21.02.2022**	Use case Bestilling System arkitektur
Elaboration 2	**21.02.2022**	**07.03.2022**	Bilde merking Prototype Initial installasjons-manual Start av testing Sekvens diagram
Construction 1	**07.03.2022**	**25.03.2022**	Software prototyper Eksamens øving
Construction 2	**25.03.2022**	**08.04.2022**	Andre Presentasjon
Construction 3	**08.04.2022**	**22.04.2022**	Integrasjon av del systemer Del tester
Construction 4	**22.04.2022**	**09.05.2022**	Klargjøres for produksjon Hel test
Transition 1	**06.05.2022**	**13.05.2022**	Fullstendig brukermanual
Transition 2	**13.05.2022**	**20.05.2022**	Kvalitetskontroll Test
Innlevering	**20.05.2022**	**23.05.2022**	Finpuss av dokumentasjon

2.3 Kravspesifikasjon

Her skal vi vise de ulike kravene som har blitt definert i begynnelsen og i løpet av prosjektet. Det er prioritets forskjeller imellom krav, noen krav har høyere prioritet og er viktigere enn andre krav, og det er derfor vi satt en prioriterings grad definert som klasser for vise å forskjellen.

Tabell 3: Prioritering av krav.

Klasse A	Kravet er nødvendig for at prosjektet skal fullføres.
Klasse B	Kravet er viktig, men systemet fungerer uten det.
Klasse C	Kravet er ønsket, men har lavest prioritet.

Krav prioritering er første del av kravene definert i i Tabell 3. Hvor vi har tre klasser, klasse A, klasse B og klasse C, rangert etter hvor viktig kravet er i prosessen av å lage systemet.

Tabell 4: Klassifisering av krav.

Funksjonelle krav - FK	Krav som definerer hvordan systemet oppfører seg eller fungerer.
Ikke-funksjonelle krav - IFK	Krav som definerer bruk og drift av systemet.

Den andre delen av krav definering er klassifisering av typene krav, dermed er kravene delt inni to klassifikasjoner, funksjonelle krav og ikke-funksjonelle krav, inni Tabell 4 er krav klassifikasjonene definert. Derfor bruker vi de forskjellige typene krav vi har fått eller funnet til dette prosjektet, videre i de andre tabellene og i den framtidige test dokumentasjonen.

I Tabell 5 har vi fått mange funksjonelle krav fra oppdragsgiver og et par forslag fra intern veileder. Vi har delt inn krav tabellen inni fire deler som er nummer, klasse, krav og hvem som foreslo kravet til å begynne med. Dette er gjort slik at fremtidig testing skal være lettere å sette opp og gjennomføre.

Tabell 5: Funksjonelle krav.

Nr.	**Klasse**	**Krav**	**Presentert av**
FK01	A	System skal gjenkjenne laks og ørret.	Kongsberg Jeger- og Fiskerforening
FK02	A	Sytemet skal telle fisk som passerer.	Kongsberg Jeger- og Fiskerforening
FK03	A	Systemet skal detektere fisk på en meters avstand.	Kongsberg Jeger- og Fiskerforening
FK04	A	Systemet skal være gjennoppladbart.	Kongsberg Jeger- og Fiskerforening
FK05	B	Systemet burde ha sitt eget oppladning system.	Henning Gundersen
FK06	B	Systemet skal sende live bilder til UI/database.	Kongsberg Jeger- og Fiskerforening
FK07	C	Systemet skal kunne gjenkjenne tre fiskearter.	Kongsberg Jeger- og Fiskerforening

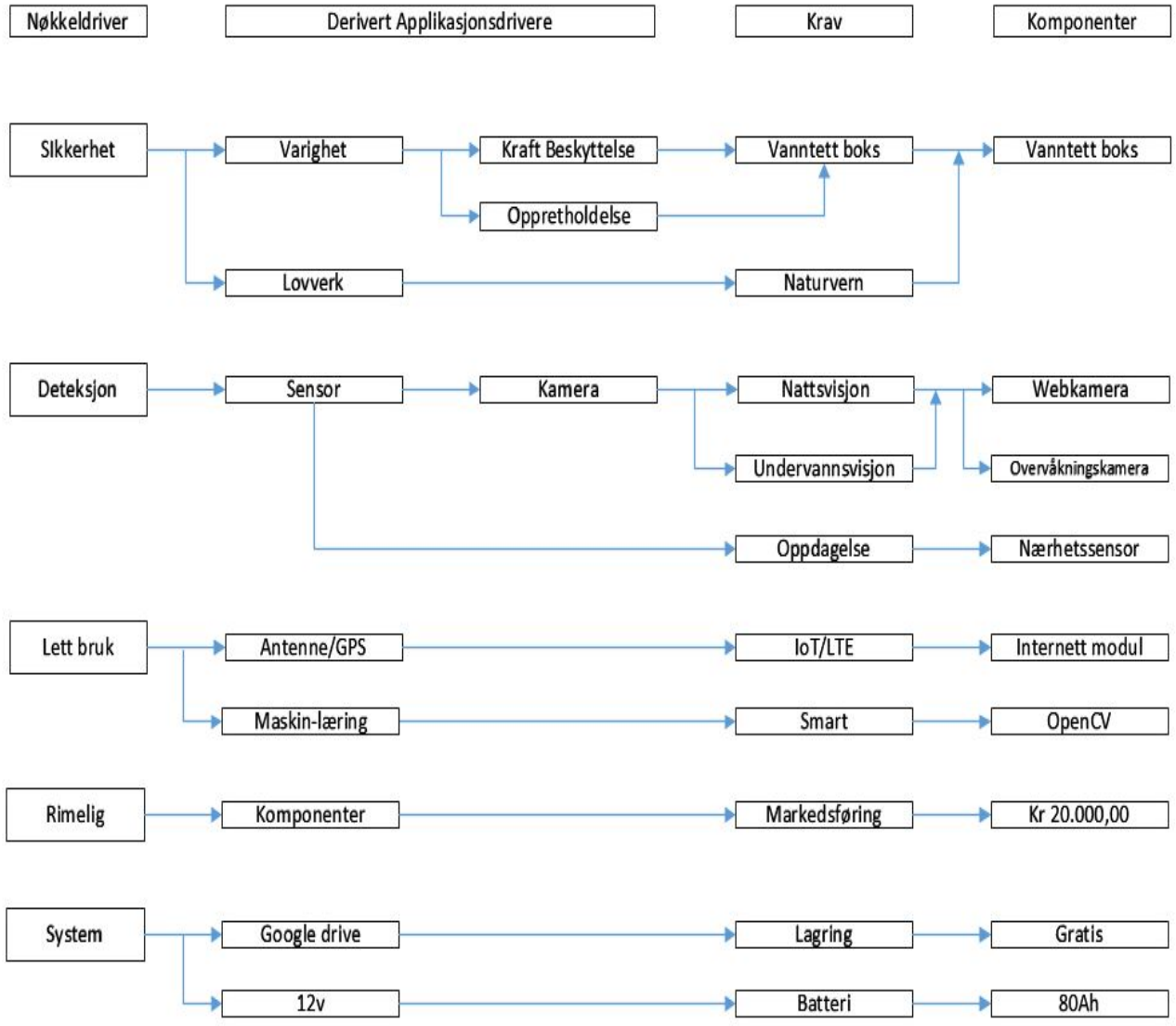
I Tabell 6 har vi fått to ikke-funksjonelle krav av oppdragsgiver og ett av oss slik at vi kan teste alle komponentene sine spesifikasjoner. Disse kravene er hovedsakelig bare krav vi kan bekrefte via inspeksjon.

Tabell 6: Ikke-funksjonelle krav.

Nr.	**Klasse**	**Krav**	**Presentert av**
IFK01	A	Kamera huset skal være vanntett.	Kongsberg Jeger- og Fiskerforening
IFK02	A	Hovedsystem huset skal være regn tett.	Kongsberg Jeger- og Fiskerforening
IFK03	B	Avlastning på ledningene som går inn i systemhusene.	Kongsberg Jeger- og Fiskerforening
IFK04	B	Komponentene skal fungere som behovet.	Gruppe 13

2.3.1 Nøkkel drivere

Nøkkel drivere vist i Figur 3, kan affektere prosjektet på mange mulige måter. Nøkkel drivere er de største faktorene som omhandler prosjektet. Forskjellige organisasjoner trenger forskjellige nøkkel drivere og kommer an på hva det endelige målet med prosjektet er.



Figur 3: Nøkkel drivere.

2.4 Budsjett

Ved å lage et estimat av hvor mye penger vi kommer til å bruke, blir det lettere å holde prosjektet innenfor budsjettet. Oppdragsgiver ønsker at sluttproduktet skal kunne bli produsert for under 50,000.00 kr. Oppdragiver ønsker ett system som kan fortsettes å bli produsert etter vi er ferdige med det, og solgt til andre fiskeforeninger og kunder. Dermed har vi laget et prisanslag på hva de forskjellige komponentene koster, sett i Tabell 7.

Tabell 7: Stykkliste for hele prosjektet sammenlagt.

Produkt	**Mengde**	**Pris**	**Frakt**	**Total pris**
Raspberry Pi 4 Model B	1	kr 2051	kr 332	kr 2383
GoPro Hero 9	1	kr 4395	x	kr 4395
Logitech C925e HD Webcam	1	kr 803	kr 99	kr 902
Kamera hus	1	kr 976	x	kr 976
Akrylplater	4	kr 389	x	kr 1556
Fugemasse	4	kr 249	x	kr 996
Fotoelektrisk sensor	4	kr 717.5	x	kr 2870
SIM7600-H 4G HAT (B)	1	kr 882	kr 556	kr 1438
Intel Neural Compute Stick 2	1	kr 745	kr 99	kr 869
Fritids batteri	1	kr 1499	x	kr 1499
Batteri lader	1	kr 649	x	kr 649
DC-DC Converter	1	kr 475	kr 145	kr 620
USB A Female to USB 4 Pin	3	kr 35	kr 110	kr 215
Strips	1	kr 40	x	kr 40
Koblingsboks	1	kr 80	x	kr 80
Rekkeklemme	1	kr 40	x	kr 40
Testkabel med krokodilleklemme	2	kr 33	x	kr 66
Aquablock Rørmannsjett	4	kr 50	x	kr 200
Dør hengsler	2	kr 25	x	kr 50
3.3kΩ Resistor	8	kr 19.3	x	kr 154.4
1.2kΩ Resistor	4	kr 2.3	x	kr 9.2
Sum				kr 20007.6

Siden dette er den totale prisen for produksjonen av produktet ville ikke det vært den endelige prisen. I Tabell 8 har vi budsjettliste over alt vi endte med å bruke og hva produktet koster å produsere, hvis det skulle ønskes å lage vanntette bokser også.

Tabell 8: Stykkliste for produksjon av systemet.

Produkt	**Mengde**	**Pris**	**Frakt**	**Total pris**
Raspberry Pi 4 Model B	1	kr 2051	kr 332	kr 2383
Logitech C925e HD Webcam	1	kr 803	kr 99	kr 902
Kamera hus	1	kr 976	x	kr 976
Akrylplater	4	kr 389	x	kr 1556
Fugemasse	4	kr 249	x	kr 996
Fotoelektrisk sensor	4	kr 717.5	x	kr 2870
SIM7600-H 4G HAT (B)	1	kr 882	kr 556	kr 1438
Fritids batteri	1	kr 1499	x	kr 1499
Batteri lader	1	kr 649	x	kr 649
DC-DC Converter	1	kr 475	kr 145	kr 620
USB A Female to USB 4 Pin	3	kr 35	kr 110	kr 215
Strips	1	kr 40	x	kr 40
Koblingsboks	1	kr 80	x	kr 80
Rekkeklemme	1	kr 40	x	kr 40
Testkabel med krokodilleklemme	2	kr 33	x	kr 66
Aquablock Rørmannsjett	4	kr 50	x	kr 200
Dør hengsler	2	kr 25	x	kr 50
3.3kΩ Resistor	8	kr 19.3	x	kr 154.4
1.2kΩ Resistor	4	kr 2.3	x	kr 9.2
Sum				kr 14311.6

I Tabell 9 har vi satt opp hvem som har produsert komponenten, hvem som er produsenten, hvilke land det er produsert i og nettsiden eller butikken vi har kjøpt komponenten fra. Eneste usikkerheten er Wifi abonnomentet fra Telia, siden vi fikk aldri vite hvor mye det kostet av oppdragsgiver.

Tabell 9: Leverandører.

Produkt	**Produsent**	**Land**	**Nettside**
Raspberry Pi 4 Model B	CanaKit	USA	Amazon.com
GoPro Hero 9	GoPro	USA	Power.no
Logitech C925e HD Webcam	Logitech	USA	Proshop.no
Kamera hus	Bud Industries	USA	no.Mouser.com
Akrylplater	RIAS	Norge	Bauhaus.no
Fugemasse	Tec7	Norge	Jernia.no
Fotoelektrisk sensor	Carlo Gavazzi	Litauen	Elfadistrelec.no
SIM7600-H 4G HAT (B)	Waveshare	Kina	Aliexpress.com
Intel Neural Compute Stick 2	Intel	USA	Dustin.no
Fritidsbatteri	Biltema	Norge	Biltema.no
Batteri lader	Biltema	Norge	Biltema.no
DC-DC Converter	XP Power	USA	no.Mouser.com
USB A Female to USB 4 Pin	StarTech	USA	no.rs-online.com
Strips	Jernia	Norge	Jernia.no
Koblingsboks	Jernia	Norge	Jernia.no
Rekkeklemme	Jernia	Norge	Jernia.no
Dør hengsel	Biltema	Norge	Biltema.no
3.3KΩ Resistor	Vishay	USA	no.Mouser.com
1.2KΩ Resistor	Vishay	USA	no.Mouser.com
WiFi abonnent	Telia	Norge	Telia.no

I Tabell 10 har vi laget en liten ekstra tabell for produkt koder for nettsidene som bruker koder slik at det er lette å finne tilbake til spesifikke komponenter.

Tabell 10: Produkt koder.

Produkt	**Produkt kode**
Fotoelektrisk sensor	Elfa: 137-59-004
Kamera hus	Mouser: 563-DPS-28707-C
DC-DC Converter	Mouser: 209-DTJ1524S05
3.3KΩ Resistor	Mouser: 71-CPF23K3000FKE14
1.2KΩ Resistor	Mouser: 71-MBA02040C1201FCT0
USB A Female to USB 4 Pin	RS: 186-2833

2.5 Helse, miljø og sikkerhet

HMS er viktig for ethvert prosjekt som inneholder elektronikk, programvare og verktøy. Poenget med HMS er at vi får sikret både personell og sluttproduktet, det er ett par prosesser vi må gjennom for at alt blir sikret innenfor prosjektet.

2.5.1 Elektroniske komponenter

Elektrostatiske utladninger kan enkelt skade elektroniske komponenter som vi har i systemet vårt. Det er mulig at elektrostatisk utladning bygger seg opp i mennesker og objekter og deretter utlades i elektronikken våres. Derfor er det viktig at vi utlader oss før vi handler med elektronikk. Metoden får å utlade seg for statisk elektrisitet er å berøre ett metall objekt som er jordet. Når du pakker ut en komponent som er følsom for statisk elektrisitet, må du ikke ta den ut av den antistatiske emballasjen før du er klar til å installere den. Før du åpner den antistatiske emballasjen, må du sørge for å fjerne statisk elektrisitet fra kroppen. Dersom du ikke skal bruke den lenger må du pakke den tilbake i en antistatisk beholder.

3 Verktøy

Inni denne delen av prosjektrapporten skal vi vise fram alle verktøyene vi har brukt i løpet av prosjektperioden, sammen med en forklaring på hva disse verktøyene er og hvordan vi har brukt de diverse verktøyene.

3.1 Kommunikasjonsverktøy

- **Discord**
  Discord er et gratis kommunikasjons program hvor vi laget en egen kanal til gruppen vår for å dele informasjon og snakke sammen utenfor universitetet. Når vi startet prosjektet var det en del Koronavirus restriksjoner, vi startet å bruke Zoom, så fant vi ut at alle i gruppen er mer komfortabel med Discord for intern kommunikasjonen.

- **Zoom**
  Zoom er ett program for å holde online møter med forskjellige personer. I begynnelsen av prosjektet brukte vi Zoom til møter med gruppen, intern og ekstern veileder. Men vi skiftet intern kommunikasjon til Discord, og vi startet med personlige møter med intern veileder på grupperommet vårt. Vi endte opp å bruke Zoom bare til møter med ekstern veileder.

- **E-post**
  Vi bruker e-post for å kommunisere oppfølging, møtereferater og diverse med intern og ekstern veilder. Samtidig som vi får informasjon av universitet om frister og ting som EXPO.

3.2 Arbeidsverktøy

- **Fritzing**
  Fritzing er et åpen kildekode-initiativ for å utvikle amatør- eller hobby-CAD-programvare for design av elektronisk maskinvare. Vi har brukt det til å illustrere små kretser innenfor prosjektet.

- **Google drive**
  Google drive er skylagring for alle filene vi har laget gjennom prosjektperioden. Gruppe

medlemmene kan lett se, legge til eller endre om det er noe feil i dokumentene på driven. Den blir også brukt til å lagre data fra raspberry pi-en

- **Overleaf/LaTeX**
  Overleaf er en online LaTeX editor som flere kan redigere og jobbe i samtidig. Siden vi har valgt å gjøre hovedrapporten i LaTeX var Overleaf ett klart valg å bruke til hovedrapport skrivning.

- **Microsoft Visio**
  Microsoft Visio er et program som er enkelt å bruke for å lage illustrasjoner og arkitektur til rapporten vår. Dermed har vi brukt Visio til å lage et par figurer til rapporten vår.

- **Draw.io**
  Draw.io er et gratis tegningsverktøy som gir muligheten til å tegne og dele på en enkel måte. Vi har brukt Draw.io for å tegne use case, sekvens diagrammer og klasse diagrammer.

- **OpenCv**
  OpenCVen er et stort biblotek for Computer vision som brukes til bildebehandling. OpenCv kan også støtte Java, C++, Python og Matlab. I tillegg er den gratis for bruk under BSD lisensen med åpen kildekode.

- **Tensorflow**
  Tensorflow er en programvareramme som er brukt for Maskinlæring og kunstig intelligens. Hovedapplikasjonene er klassifisering, persepsjon, forståelse, oppdagelse, prediksjon og skapelse. Tensorflow er skrevet i Python, C++ og Cuda, og støtter C++, Java, Javascript og Python. Vi har brukt Tensorflow for vårt prototype der den er brukt for å gjenkjenne fisk.

- **Google Colab**
  Google Colab er et program som er brukt for trening av modellen.

- **LTspice**
  LTspice er brukt til å simulere elektriske kretser, så vi kan bruke den til å kontrollere at vi ikke har gjort feil kretsdesign med å simulere kretsen på forhånd.

- **Remoteit:** Remoteit er et program som brukes til å jobbe på raspberry pi over internett.

- **VNC**
  VNC er et program som vi har brukt til å koble til raspberry pi. For å få tilgang til vår Raspberry Pi som er viktigste komponent av systemet vårt via internett. Vi var avhengig av et program slik at vi kunne få tilgang til raspberry pi uten at vi måtte være på stedet. VNC og remoteit er en perfekt program for slik prosjekt for at det gir muliheten at du kan jobbe med prosjektet når som helst. Nå, viser vi oppsett av VNC.

- **LabelImg**
  Vi brukte labelimg for å annotere bildene.(4)

- **Roboflow**
  Roboflow er et program som vi har brukt for å annotere datasettet vårt, roboflow gjøres det enkelt å eksportere datasettet til forskjellige format. Som er bra for som brukes forskjellige modeller som tar forskjellige input.

- **Makesene.ai**
  Makesense.ai er et program som er brukt til å annotere data innen maskin læring. Det har vi brukt for å annotere datasett vårt for å trene modellen videre på det. Vi opplevde at makesense.ai er mye enklere og raskere å annotere med.(5)

4 Risikoanalyse

I denne seksjonen av rapporten skal vi se på hvorfor vi gjør en risikoanalyse, hva er risiko, vår risiko matrise, forskjellige risiko nivå-er og hvordan vi ville håndtert alt av risiko.

4.1 Meningen med risikoanalyse

Gjennom utviklingen og planleggingen av systemet har vi notert oss flere problemer som kan oppstå med systemet. Ved å lage en komplett risikoanalyse kan vi muligens hindre framtidige dilemma-er i videre utvikling av systemet. Dermed er hensikten å utdype relevante risiko-er i prosjektet. I risikoanalysen er det viktig å prioritere de største problemene som kan oppstå, slik at vi kan minimalisere de og gjøre det lettere for personer å ta hensyn til dette.

4.2 Om risiko

En risiko er en måte å måle sannsynlighet mot konsekvensene etter eller av en hendelse. Grunnen til å gjøre rede for risikoene er for å forhåpentligvis unngå katastrofale scenario-er. Ved å utføre en risikoanalyse kan vi få oversikt på risikoene på en veldig oversiktlig metode.

$$Risikonivå = Sannsynlighet \cdot Konsekvens \tag{1}$$

I videre tabeller kommer vi til å bruke ordene risikonivå, sannsynlighet og konsekvens. I Formel 1 har vi satt opp hvordan disse tingene korrelerer.

4.3 Risiko matrise

I dette avsnittet har vi rangert de ulike risikoene fra 1 til 5 sannsynlighet, og 1 til 5 konsekvens. Disse rangerer fra ubetydelige risiko-er til katastrofer vi ikke kan gjøre noe med. Med Risiko-matrisen sett i Tabell 11, er det lett å se hvor farlige for systemet vårt noen risiko-er er i forhold til andre.

Tabell 11: Risiko-matrise.

Konsekvens		Sannsynlighet				
		1	2	3	4	5
		Ubetydelig	Marginal	Kritisk	Alvorlig	Katastrofe
1	Ubetydelig	1	2	3	4	5
2	Marginal	2	4	6	8	10
3	Kritisk	3	6	9	12	15
4	Alvorlig	4	8	12	16	20
5	Katastrofe	5	10	15	20	25

Fra Tabell 12, har vi en sannsynlighets-matrise som enkelt sett bare viser hvilken sjanse sannsynlighets-graden har.

Tabell 12: Sannsynlighets matrise.

Sannsynlighetgrad	Sjanse
1	10%
2	30%
3	50%
4	70%
5	90%

I tilegg til sannsynlighets matrisen har vi også en konsekvens-matrise, sett i Tabell 13.

Tabell 13: Konsekvens matrise.

Konsekvens	Sjanse
1	Vil ikke nødvendigvis ha noe å gjøre med systemet
2	Vil sannsynligvis ha mildere effekt på systemet, men ikke prioritet.
3	Vil ikke nødvendigvis gjøre systemet ustabilt men forsinkelser er forventet.
4	Kan gjøre at systemet blir ekstremt forsinket og ustabilt.
5	Vil gjøre systemet ustabilt og umulig å bruke.

4.4 Risiko nivå-er

Risikonivåene kan bli sett i Tabell 14, her har vi prioritetsliste for hvilken risiko-er som trengs å ta hånd om først. Risiko-er som er rangert fra 20-25 er klassifisert katastrofal, det betyr at systemet ikke kommer til å fungere eller i verstefall harmfull hvis ikke tatt hånd om.
Alvorlig risk, rangert fra 15-16 er klassifisert alvorlig, som betyr at systemet igjen ikke kommer til å fungere dersom vi ikke tar tiltak.
Betydelig risiko, rangert fra 10-12 er klassifisert betydelig, det vil si at handlinger behøves ellers vil det vesentlige med systemet ikke fungere og prosjektet vil være hensiktsløs.
Medium risiko, rangert fra 8-9 er klassifisert moderat betyr at et tiltak ville generelt sett gjort systemet vårt bedre, dette er nødvendig å adressere får å få et utmerket system.
Lav risiko, rangert 4-6 er klassifisert marginal, dette vil si at tiltak vil mest sannsynlig øke livskvaliteten til systemet.
Minimal risk, rangert 1-3 er klassifisert ubetydelig, grunnen til at denne er ubetydelig er siden det nesten ikke har noe effekt på systemet og hvordan vi ville eventuelt utført prosjektet.

Tabell 14: Risiko nivåer.

Risk nivå	Prioritet
Katastrofal Risk 20-25	Katastrofalt. Tiltak må påføres.
Alvorlig Risk 15-16	Alvorlig. Tiltak skal påføres.
Betydelig Risk 10-12	Betydelig. Tiltak trengs snarest å påføres.
Medium Risk 8-9	Moderat. Tiltak burde påføres.
Lav Risk 4-6	Marginal. Tiltak nødvendig, men haster ikke.
Minimal Risk 1-3	Ubetydelig. Tiltak unødvendig.

4.5 Håndtering av risikoene

De risikoene vi foreløpig har kan bli sett i Tabell 15. Vi har en gjennomsnittlig poengsum for sannsynlighet, konsekvens og risikonivå for risikoene våre. Gjennomsnittlig får vi sannsynlighet på 2,5 og en konsekvens på 3,4 med en risikonivå på 7,9.

Tabell 15: Forskjellige risiko-er.

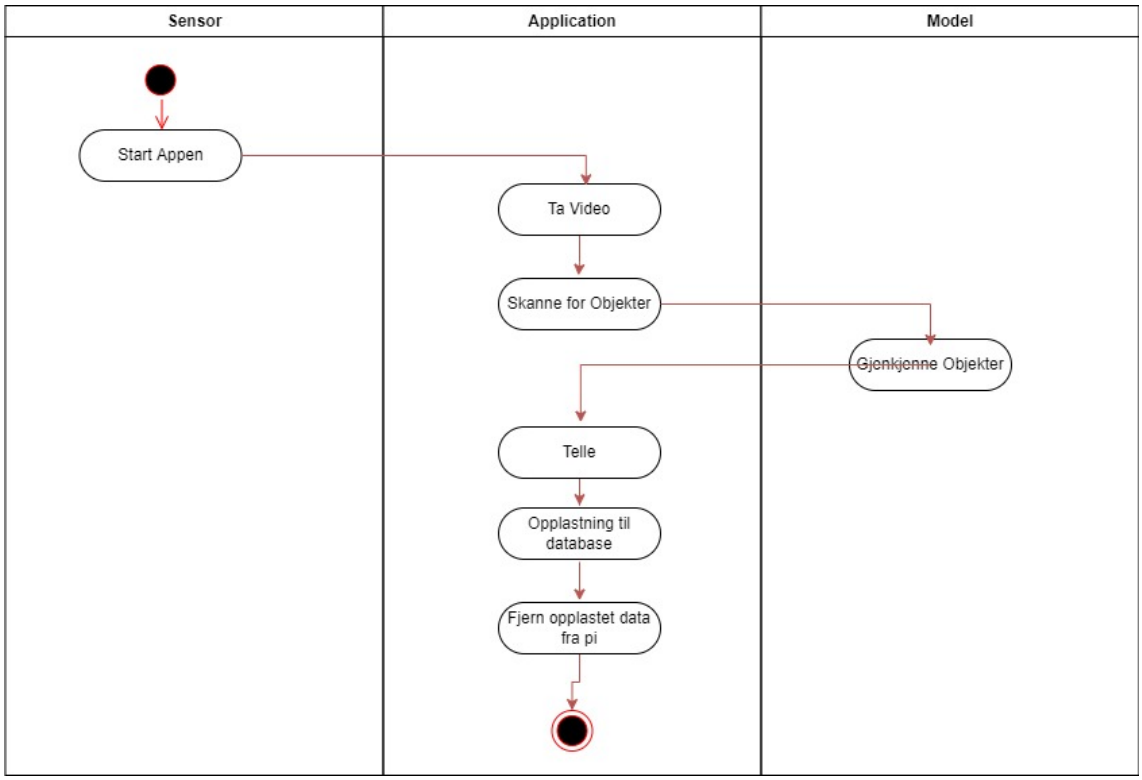
Risikoanalyse for telling av fisk i elver og i fisketrapper				**Risikovurdering**		**Risikonivå**
				S	K	
			Gj. Snitt risiko	2.47	3.2	**7.89**
Risiko nr.	**Hendelse**	**Konsekvens av feil**	**Tiltak**	**S**	**K**	
R1	Feil med Maskinsyn programmer.	Beregningsprogram vil ikke fungere.	Iterere på koden og så vi kan få deteksjon og telling til å fungere.	1	2	2
R2	Identifisering.	Identifisere fisk feil.	Justere måten vi teller fisk på.	2	2	4
R3	Feil telling.	Teller feil eller ikke i det hele tatt.	Kjøre flere korreksjonstester, så vi kan få høyere nøyaktighet.	2	2	4
R4	Dårlig miljø.	Installasjon/testing må utsettes.	Gjøre oss klare på værmeldinger før eventuelle tester/installasjoner.	2	3	6
R5	Implementasjon.	Feil/dårlig sammenkobling av systemet.	Kjøre en del deltester av komponenter i systemet.	3	4	12
R6	Budsjettering.	Bruker mer penger enn det vi har lov til.	Lage ett godt estimat for prosjektet.	1	5	6
R7	Menneskelig feil.	Sykdom, uhell, familie problemer, osv.	Lage ett godt arbeidsmiljø, profesjonell og vennlig.	5	1	5
R9	Defekt i komponenter.	Komponenter er defekte.	Bestille varer godt i forkant og kanskje ekstra komponenter.	4	3	12
R10	Sen sending av komponenter.	Lite tid til å sette sammen systemet.	Lage komponent liste slik at vi kan bestille varer i god forvei for prosjektet.	3	2	6
R11	Dårlig arbeid/Kommunikasjon.	Svakt gruppearbeid.	Lage en god prosjektmal som vi alle kan følge opp.	4	4	16
R12	Feil med OpenCV program.	Beregningsprogram vil ikke fungere.	Vi kan enten prøve å iterere koden eller se om vi kan bruke andre programmer.	1	3	3
R13	Installasjons feil.	Systemet vil muligens ikke fungere.	Vi lager en brukermanual for å sette opp systemet.	2	4	8
R14	Lovverk.	Systemet følger ikke lovverket.	Gjøre oss klare på Norske lover om fisketrapper og naturvern.	2	5	10
R15	Miste koden.	Kildekoden blir slettet/borte/korrupt.	Ta regelmessige sikkerhetskopier av koden.	2	5	10

5 Modellering av produktet

I dette kapitellet har vi brukt use case og diagrammer for å innkapsle deler av prosjektet vårt. Følgende snakker vi om operasjonelle scenario, use case, sekvens diagram og software arkitektur.

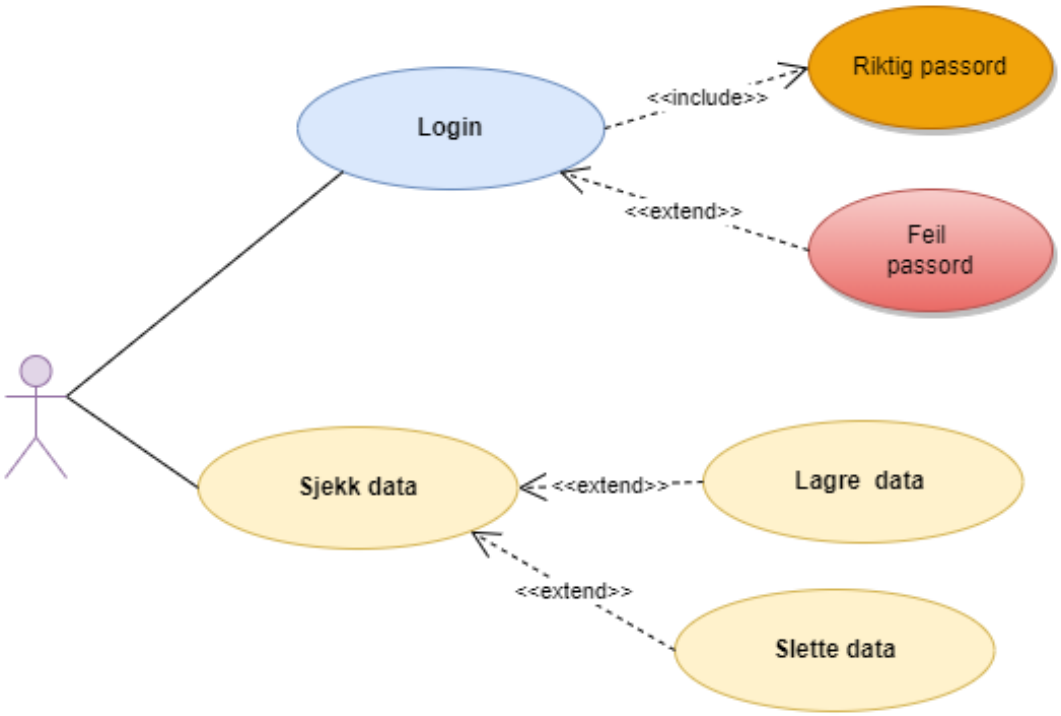
5.1 Operasjonelle Scenario

I dette operasjons scenario-et beskriver vi hvordan vårt system fungerer. Når sensoren oppdager noe i fisketrapp, så sender signal til raspberry pi og deretter raspberry pi sender signalen til Kamera for å ta en video opptak. Deretter skal raspberry pi-en prosessere video-en for å gjenkjenne fisk. Hvis det ligger fisk i video-en skal system telle antallet og sende både tellingen og video-en til en lagringsplass. Som bruker har tilgang til og kontroll på.



Figur 4: Aktivitetsdiagram.

5.2 Funksjonelle (Use Case)



Figur 5: Use Case Model

Login

Hensikten å ha login er å begrense adgangen til alle dataene, slik at kun de som er registrert som bruker skal ha adgang dataene.6

Bekreftet

Brukeren må angi riktig passord for å få tilgang.7

Feil passord

Når brukeren angir feil passord, vil brukeren bli varslet.vist i8

Sjekk status

For å ha oversikt på system om systemet fungerer. I våre tilfelle vi sjekker om systemet sender inn dataene fra raspberry pi-en til google driven som er hoved lagringplassen vår. 9

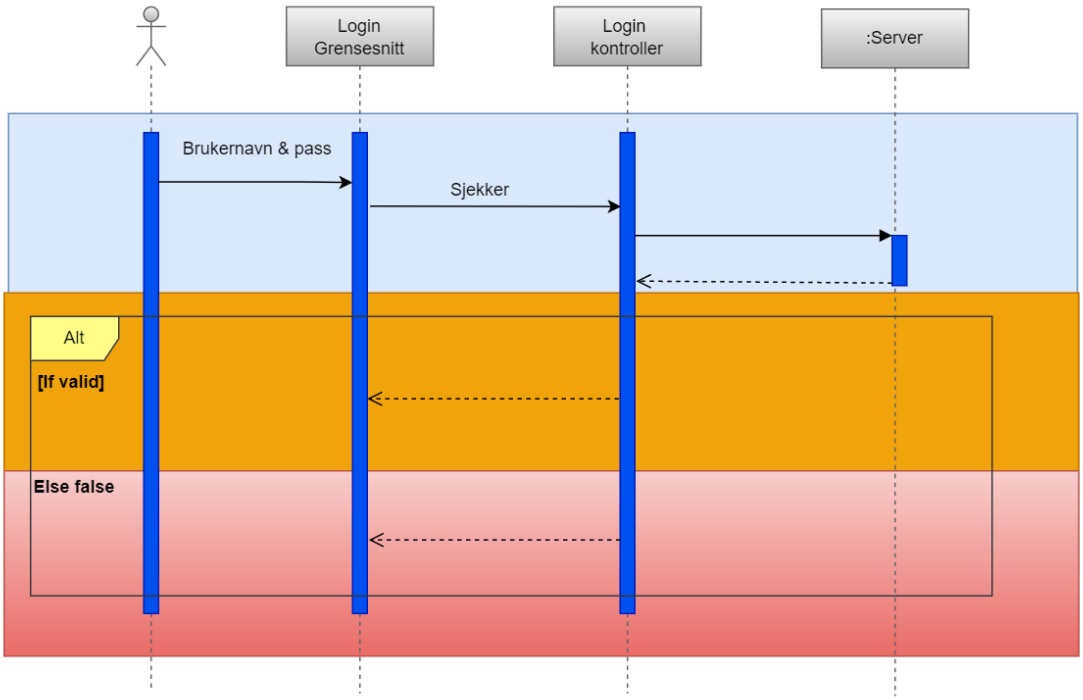
Sjekk data

Muligheten til å se på bilder, fisktall og slette data om det er unødvendig.

5.3 Sekvens diagram

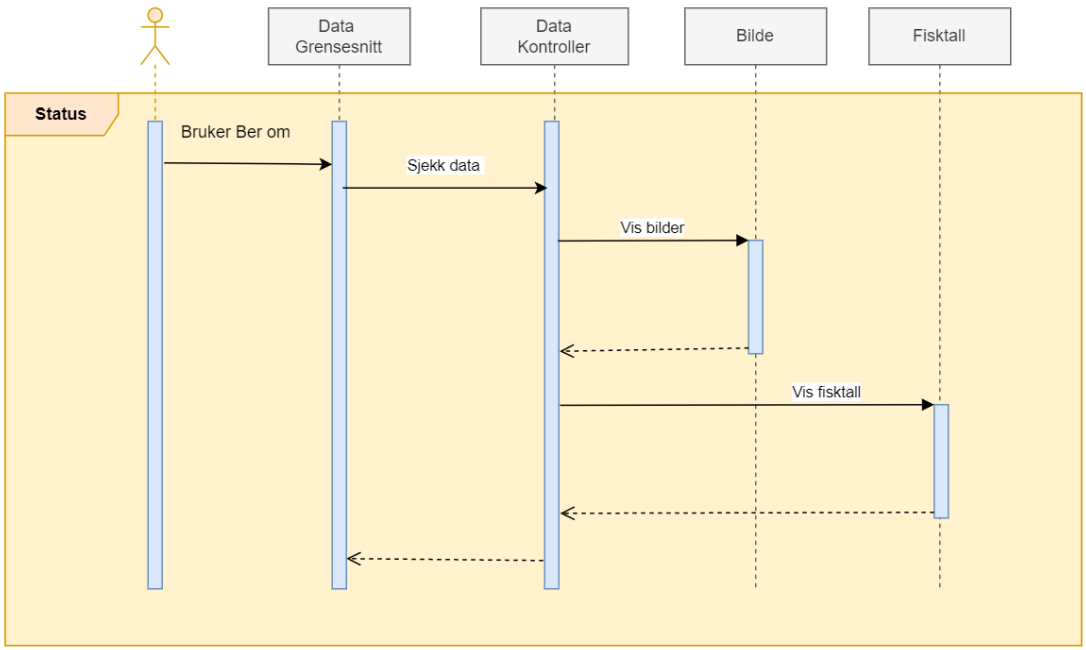
Hensikten med sekvensdiagram er å vise hendelsesekvenser som fører ønsket result. I sekvens diagram vises enkelt og veldig detaljert samhandlinger mellom objekter og aktører.

5.3.1 Login



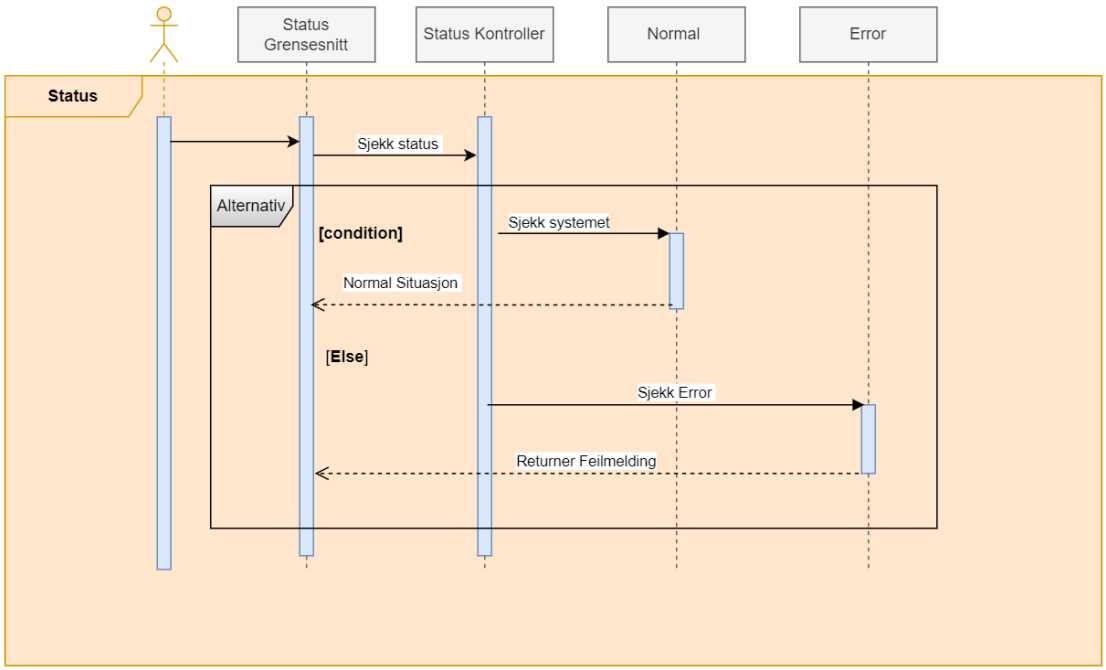
Figur 6: Login

5.3.2 Sjekk Data



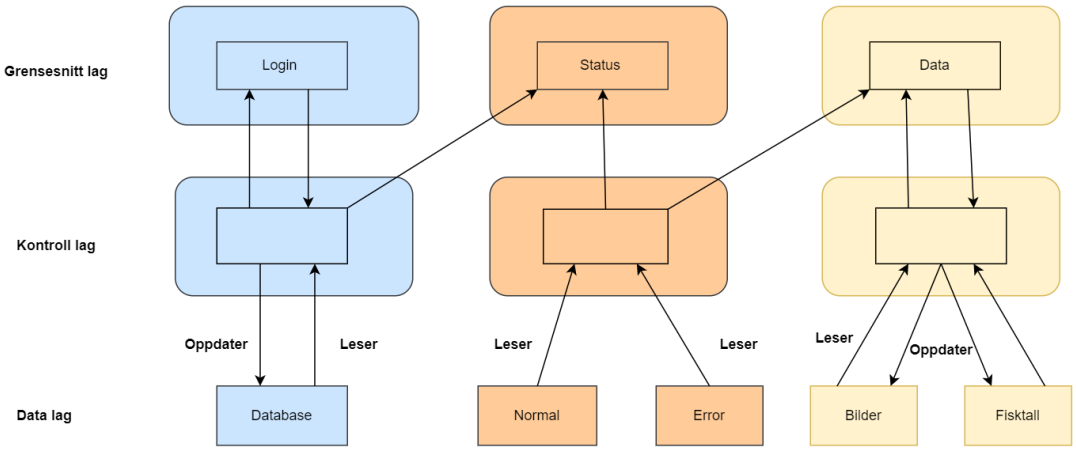
Figur 7: Data sjekk

5.3.3 System status



Figur 8: System sjekk

5.4 Programvare arkitektur



Figur 9: Programvare Arkitektur

6 Maskinvare

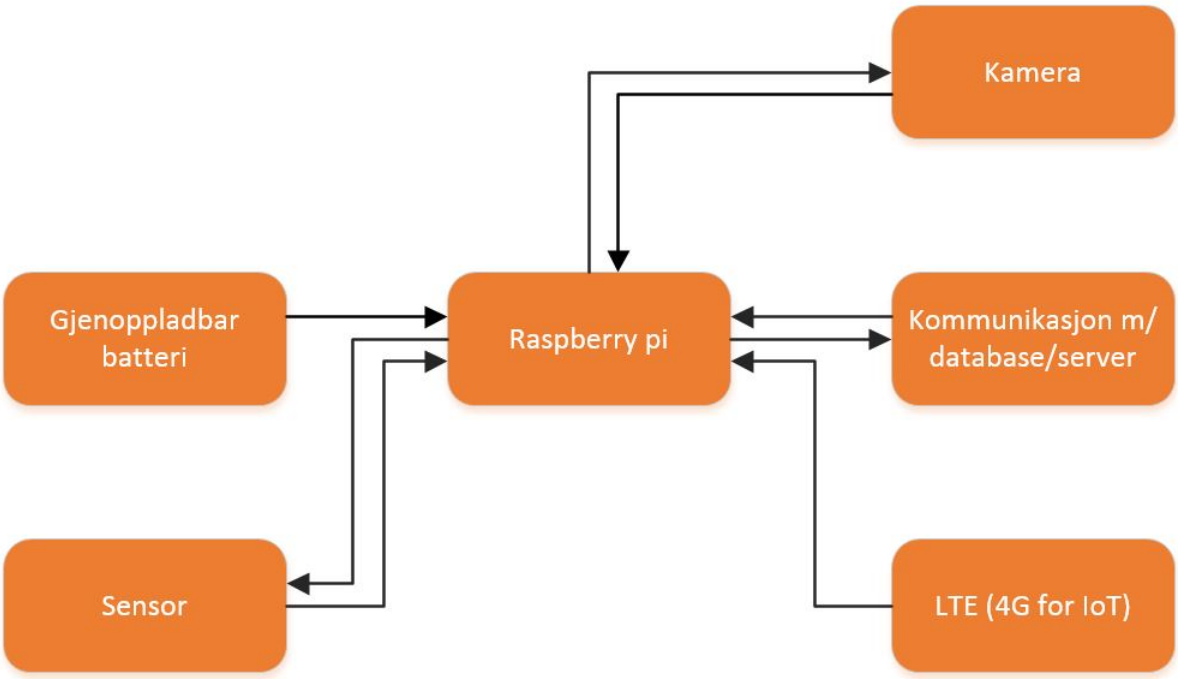
I denne delen av rapporten skal vi forklare utvalg av komponentene brukt i systemet, sammen med utregningene av komponentene og andre komponenter.

6.1 Komponent utvalg

Her har vi oversikten til maskinvaren sammen med alle komponentene til systemet og hvorfor vi valgte disse spesifikke komponentene over andre med utdypning.

6.1.1 Maskinvare oversikt

I Figur 10, har vi laget ett blokkdiagram for å visualisere komponentene til systemet og hvordan de kommuniserer med hverandre som vist av pilene.



Figur 10: Oversikt av systemets maskinvare.

6.1.2 Kompakt pc

Kompakt pc-en er den viktigste komponenten i system designet vårt siden oppdragsgiveren har ønsket at vi bruker en kompakt pc til å gjøre jobben, og det er bare en slik som kan gjøre jobben. Men vi ønsket å se om det var mulig å gjøre det samme med en mikroprosessor Arduino i stedet, hovedsakelig fordi pris forskjellen er ganske stor. Spesifikasjonene til Arduino Uno R3 og Raspberry Pi 4 Model B er nedenfor i Tabell 16.

Tabell 16: Kompakt pc og mikrokontroller spesifikasjoner.

Spesifikasjoner	Arduino Uno R3	Raspberry Pi 4 Model B
CPU	16MHz	1.5GHz
GPU	x	VideoCore VI 3D Graphics
RAM	2KB	8GB
USB	x	4x
HDMI	x	2x 4Kp60
Pris	250 kr	2200 kr

Gjennom spesifikasjonene i Tabell 16 og kunnskapen vi allerede har om hva systemet vårt trenger er valget klart. Raspberry Pi har et innebygd grafikk kort og en inngang for et eksternt grafikk kort hvis det innebygde ikke er nok, mer prosesserings kraft og lagringsplass. I forhold til Arduino som for å kunne gjøre samme jobben trenger mange eksterne komponenter for å gjøre samme jobb. Dermed for oss som skal holde på med bilde prosessering er det mest realistiske valget ut ifra spesifikasjonene til både Arduino og Raspberry Pi er å bruke Raspberry Pi for å ikke overkomplisere prosjektet. En oppgradering fra Raspberry Pi ville vært å bruke en Jetson Nano, men desverre var det ingen igjen på markedet i starten av prosjektet.

6.1.3 Kamera

Systemet vårt trenger ett kamera som kan bli brukt til å ta gode bilder i dårlige lysforhold og miljøer. Siden systemet er ment til å kunne ta bilder undervann og om natta. Dermed er det krav å kunne ta undervanns bilder i dårlige lysforhold, så det er dette vi ser etter i neste del.

Tabell 17: Kamera spesifikasjoner.

Spesifikasjoner	GoPro Hero 9	GoPro Hero 10	Olympys TOUGH TG-6
Produkttype	Actionkamera - 5K	Actionkamera - 5K	Digitalkamera - 4K - kompakt
Oppløsning	20 MP	23 MP	12 MP
Maks oppløsning	5120x2880	5120x2880	3840x2160
Vanntett	10m	10m	15m
FPS	30	60	30
Bildestabilisator	Hypersmooth 3.0	Hypersmooth 4.0	Optisk
Pris	4400 kr	5790 kr	3986 kr

Her har vi stilt fram tre mulige valg for kamera med mesteparten av spesifikasjonene, Olympus TOUGH TG-6, GoPro Hero 9 og 10 som sett i Tabell 17. Oppløsning og maks oppløsning er ikke viktig for kamera vi skal kjøpe siden vi skal nedgradere grafikken på bildene slik at bildene tar mindre plass på minnekortet uansett, dermed er Olympus den beste for dette deretter Hero 9. Kamera må være vanntett siden det skal filme fisk undervann, og alle tre er vanntette, men for ekstra beskyttelse av kamera har vi valgt å kjøpe hus til kamera som sett i seksjon 5.3.2 av rapporten. For FPS eller bilderuter per second vinner Hero 10 og uavgjort mellom Hero 9 og Olympus. Bildestabilisator er bare for å stabilisere bilde, men her vinner GoPro'ene siden de har sin egen programvare dedikert til bildestabilisasjon sammen med lette innstilling oppsett for bruk om kvelden og natta.

I konklusjon har vi valgt GoPro Hero 9, siden den har best pris for spesifikasjonene vi får, Olympus er også ganske bra men for bruk undervann og om natta er GoPro'ene mye bedre generelt. Hero 10 er bare en dyrere versjon av Hero 9 i forhold, dermed er det bedre med Hero 9.

6.1.4 Sensor teori

Når vi ser på deteksjon av fisker er det tre hovedtyper av sensorer som brukes for fisketelling. Disse typene sensorer for fiske telling er resistiv, optisk og hydroakustisk tellere. Her skal vi se på fordelene og ulempene sammen med hvilken av disse sensorene vi har valgt å jobbe med.

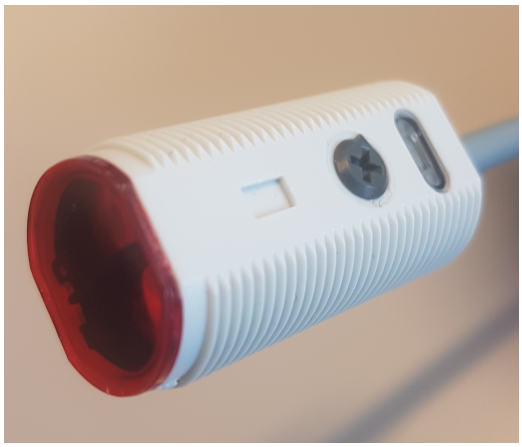
Resistiv teller er den mest uvanlige innenfor fisketelling imellom disse tre tellerne. Sensoren måler den elektriske motstanden til vannet og fisken med elektroder og en wheatstone-bro, siden vann har større elektrisk motstand enn fisken ser sensoren endringen i elektrisk motstand og sier ifra at "noe" er der.

Optisk teller bruker lysstråler som fiksen passerer for å finne ut størrelsen og farten på fisken. Fisken blir så opptelt av brukeren og validert med kamera. Optiske tellere kan bare bli brukt i små elver eller fiske trapper.

Hydroakustisk teller er veldig tilpassings dyktig, altså de kan bli brukt i store innsjøer og elver uten bruk av strukturer som fiske trapper. Men i gjengjeld har hydroakustiske tellere dårlig ytelse og er mye dyrere enn de andre tellerne i forhold. (6)

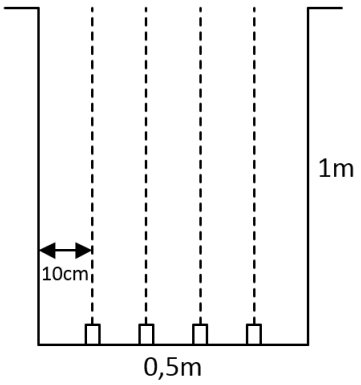
Mellom de tre sensor typene, resistiv teller bruker for mye plass siden du må ha store mellomrom mellom de tre sensorene for å kunne bruke denne typen. Hydroakustisk er den generelt beste sensor typen innenfor fisketelling men den er for dyr, vanskeligst å installere og mest unøyaktig. Optiske tellere er best innenfor fisketrapper men lyset kan være plagsomt for fisken, som ville gått imot norsk lovverk. Dermed må mann bruke optisk sensor med infrarødt lys for å kunne bruke denne typen sensor.

For vårt system har vi valgt å bruke fotoelektrisk sensorer som er under optiske tellere. En annen grunn til å velge optiske sensorer er fordi det er allerede fungerene eksempler ute i feltet, for eksempel Riverwatcher laget av Vaki Aquaculture Systems LTD fra Island. Denne typen sensor bruker rødt lys og infrarødt lys til å telle når strålen den sender ut blir reflektert tilbake til sensoren av fisken som bryter strålen. Vi bruker typen som har infrarødt lys for å ikke forstyrre fisken. Siden vi skal detektere en fisk på ca. en halvmeters avstand siden det er den generelle bredden på en fisketrapp sin inngang/utgang må vi lage ett array av sensorer for å kunne detektere all fisken på halvmeters avstanden.



Figur 11: Fotoelektrisk sensor.

For oppgaven om å telle fisk bruker vi PA18CAD10PASA - Light scanner fotoelektrisk sensor som vi kjøpte fra Elfa, som vist i Figur 11. Dette er en diffuse-reflective optisk sensor, men denne spesifikt bruker infrarødt lys og kommer med kabel for lettere installasjon. Det at sensoren er diffuse-reflective betyr at sensoren har både sender og mottaker vedsiden av hverandre inni sensoren. Sensoren har IP67 så vi kan bare bruke den i vann opptil en meter dybde. Sensorene har en skrue som sett i Figur 11, denne skruen justerer lengden på strålen til sensoren, men vi har bare settet den på maks siden oppdragsgiveren ønsker deteksjon på 1 meters avstand, som er maks rekkevidden til strålen.



Figur 12: Sensor rekke oppsett.

Vi har kjøpt inn fire sensorer for å sette opp en liten sensor rekke som vist i Figur 12. Basert på informasjonen fra Ekstern veileder så har fisken bredde på ca. 7-10cm dermed kan vi sette opp en sensor rekke med 10cm mellomrom fra veggen til første sensor og 10cm mellom

hver sensor også. Sensorene sin stråle har også radius på ca. 1cm så det er mindre plass for fisken å passere gjennom uoppdaget.

6.2 Hvorfor raspberry pi 4 B

Vi hadde ett par valg å velge mellom når det gjelder kompakt kort. For enkelhetens skyld velger vi mellom de to hovedtypene som er raspberry pi og arduino. Vi endte med å velge raspberry pi-en siden den har mye bedre bilde prosesserings kraft, som vil si at den kan sende flere bilder til databasen på kortere tid. raspberry pi-en er ikke en mikroprosessor, det vil si at den har en innebygd GPU.

6.3 Sette opp raspberry pi

(7) Det som gjelder nå er å sette opp raspberry pi-en. Fra brukermanualen har vi noen steg vi kan simpelt følge. Refererer til brukermanualen som følger med raspberry pi-en **??** side 3 til 5:D

1. Installer raspberry pi-en i boksen sin (for å gi ekstra beskytelse).

2. Neste er å installere kjøleribben hvis det er behov (altså hvis raspberry-en overheter).

3. Ett alternativ er å bruke en CanaKit vifte for høy ytelse applikasjoner. Dette kan bli gjort ved å koble rød og svart kabel til 5V og GPIO pinne 04, henholdsvist. Pinnene er vist i 13.

4. Sett inn MicroSD kortet som allerede har lastet inn NOOBS versjon 3.1.0 inn i raspberry pi-en.

5. Koble til tastatur og pc mus til raspberry pi-en.

6. Koble til HDMI monitor eller pc skjerm.

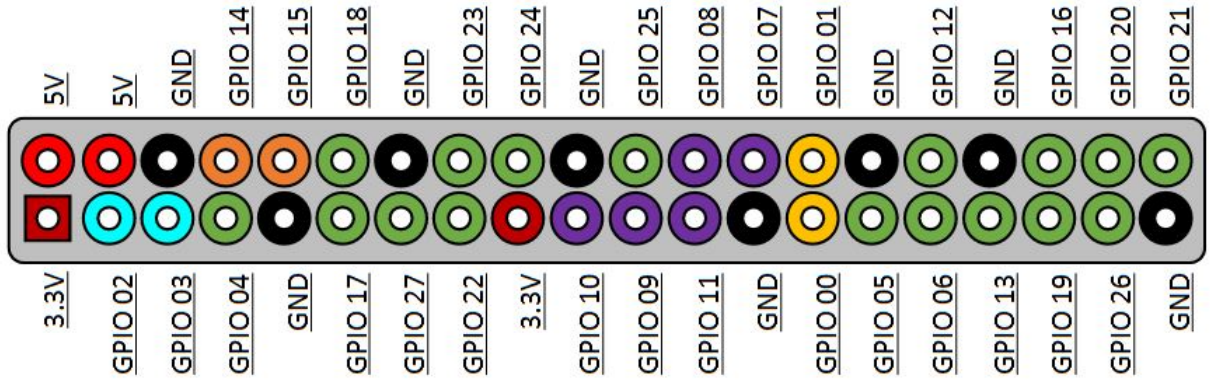
7. Koble raspberry-en til strømmen og kjør "Raspian.

8. Nå er det bare å vente på installeringen.

9. Systemet burde restarte og oppsettet burde være i orden.

6.4 Pinner og oversikt

Når vi kjøper og kobler systemet vårt er det viktig å benytte seg av hvordan portene fungerer. Fra Figur 13, har vi pinnene til raspberry pi-en lagd i Microsoft Visio. 5V og 3.3V portene er spenningsinnganger. GPIO innganger er enkelt sagt bare programmerbare porter. GND er jording. Programmet vi bruker for å programmere portene er i språket Python 2.



Figur 13: Raspberry pi pinout.

6.5 SIM system og abonnement

Vi har mange måter å koble SIM kort til rapsberry pi. Det som er lettest og mest effektivt å gjøre er å koble en HAT utvidelse til raspberry-en som vil gi oss en SIM kort luke. Da gjelder det De største konkurrentene på markedet er waveshare sin SIM7600X 4G HAT, og sixfabs 4G/LTE Cellular Modem Kit. Fra Tabell 18 har vi laget en spesifkasjonsliste for de forskjellige typene. Forskjellene mellom disse er funnet i (8)
Utifra kunde anmeldelser sier de at de kun har oppnådd hastigheter på kun 30 Mbps opp og ned, og derfor er vi kritiske til sixfabs LTE kit. Det anbefales å bruke sixfabs LTE kit siden SIM7600 har vanskelig antenna å montere, men hvis man gjør god undersøkelse burde

Spesifikasjoner	SIM7600X m/ antenne	4G/LTE Cellular Modem Kit
Region	Global	Europa, Afrika og Midtøsten
Datarate	50 Mbps Opp, 150 Mbps Ned	150 Mbps Opp, 50 Mbps Ned

Tabell 18: SIM spesifikasjoner.

det gå bra. Derfor velger vi SIM7600 komponenten. Når vi bestiller SIM hatten kan vi velge mellom flere typer,og vi kommer oss fram til av vi velger global serien som er betegnet SIM7600G-H. Abonnement vi bruker er fra Telia siden de har sin egen avdeling for IoT enheter, og vi trenger bare 1 GB for prototypen av systemet våres. Prisen fra de forskjellige 4G/5G leverandørene varierer men systemet vi lager er hovedsaklig i Kongsberg og Telia sitt nett er velfungerende i byen.

6.6 Batteri beregning

Systemet skal ligge ute i elver og fisketrapper rundt om kring i Norge, da er det viktig at vi har en god batteriløsning for systemet. I denne delen skal vi ta for oss de forskjellige batteri beregninger og kontrollering.

6.6.1 Finne energi forbruk

Ett av de viktige tingene med systemet har med batteriet og levetid. For at systemet skal fungere ute når vi ikke nødvendigvis har stikkontakter eller andre strømkilder må vi finne en løsning for dette. Da velger vi å finne ett batteri som kan forsyne systemet vårt. Det første vi kan gjøre for å finne riktig type batteri er å lage en energi margin for systemet. Her er det laget ett par tabeller for de forskjellige strømforbrukene fra høyt til lavt. Tabell 19 viser hvor mye strøm systemet bruker når den bruker aller mest strøm. Tabell 20 viser hvor mye strøm systemet bruker når den bruker aller minst strøm. Det vil si at raspberry pi-en går på absolutt minste mengden strøm mulig. Til slutt viser Tabell 21 hvor mye strøm systemet bruker når den trekker gjennomsnittlig strøm. Energiforbruket er bestemt av hva slags apparat det er, strømforbruk, spenning, mengden apparater og hvor mange timer systemet er på. Strømforbruket for raspberry pi-en er funnet i (9). Strømforbruket for SIM hatten er funnet i (10). Strømforbruket for USB webkamera-et er funnet i (11).

Apparat	Strømforbruk	Spenning	Watt per time	Mengde	Timer	Total watt
Raspberry pi	1300mA	5,1V	6,5W	1	24	156,0W
Kamera	500mA	5,0V	2,5W	1	4	16,0W
Sensor	100mA	12,0V	1,2W	4	24	115,2W
Sim hat	2,3mA	5,0V	0,0115W	1	24	0,3W
Total						287,5W

Tabell 19: Strømforbruket til systemet (Høyest).

Apparat	Strømforbruk	Spenning	Watt per time	Mengde	Timer	Total watt
Raspberry pi	600mAh	5,1V	3,1W	1	24	74,4W
Kamera	500mA	5,0V	2,5W	1	4	16,0W
Sensor	10mA	12,0V	0,12W	4	24	11,52W
Sim hat	2,3mA	5,0V	0,0115W	1	24	0,3W
Total						102,22W

Tabell 20: Strømforbruket til systemet (Lavest).

Apparat	Strømforbruk	Spenning	Watt per time	Mengde	Timer	Total watt
Raspberry pi	1080mAh	5.1V	5,4W	1	24	129,6W
Kamera	500mA	5,0V	2,5W	1	4	16,0W
Sensor	40mA	12,0V	0,48W	4	24	46,08W
Sim hat	2,3mA	5V	0,0115W	1	24	0,3W
Total						191,98W

Tabell 21: Strømforbruket til systemet (Gjennomsnittlig).

Altså det er viktig å beregne belastningen i watt for å kunne bestemme batteri størrelsen. Effekt formelen for å finne watt bruk er gitt ved Formel 2.

$$P = U \cdot I \tag{2}$$

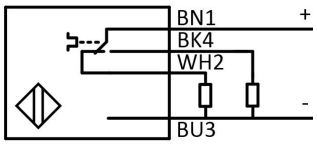
Hvor P er effekt oppgitt i watt, U er spenning i volt og I er strøm i ampere. Dersom vi går ut ifra gjennomsnittlige forbruket av systemet kan vi se at vi bruker 200Wh. Ett eksempel av beregningene for gjennomsnittlig forbruk av strøm så ser vi at raspberry pi bruker 1010mA gjennomsnittlig og trekker 5 volt. Dermed blir utregningen som sett i Utregning 3:

$$P = 5V \cdot 1010mA \tag{3}$$

Dette gir oss en effekt på 5,1 Watt, og slik har alle effektverdiene blitt regnet. Det totale energi forbruket vårt kommer på under 200 watt og kan bli sett i Tabell 21.

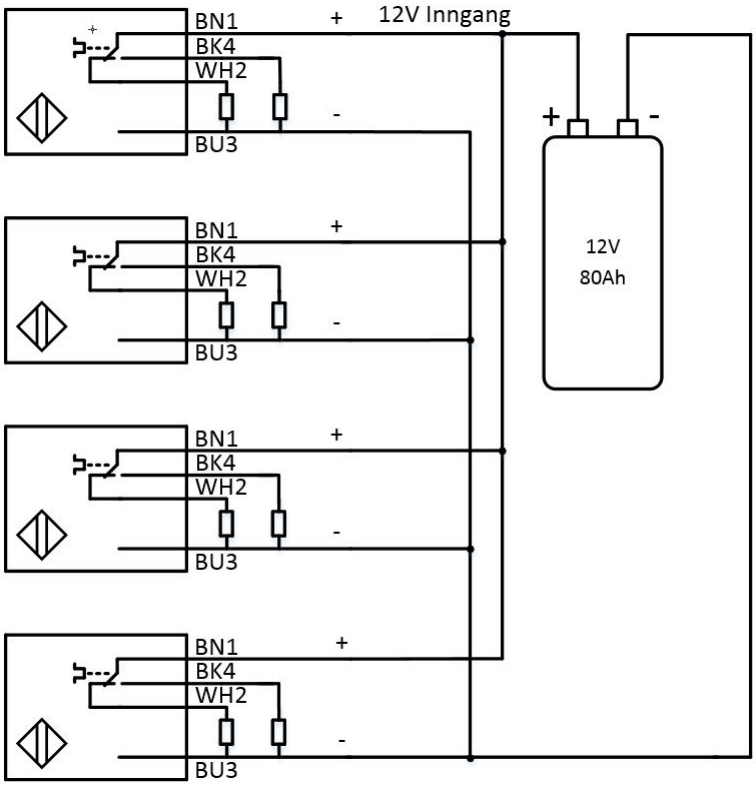
6.7 Sensor

Fra installasjonsmanualen gitt av sensorene vi kjøpte fra Elfadistrelec sier at vi skal koble sensorene som vist i Figur 14, her ser vi at BN1 er brun ledning, BK4 er svart ledning, WH2 er hvit ledning og til slutt BU3 er blå ledning.



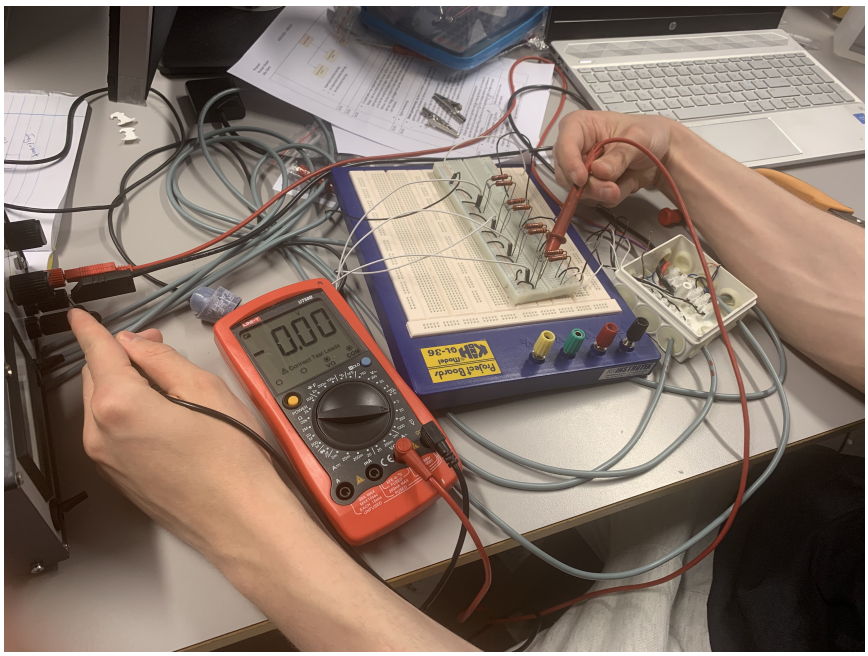
Figur 14: Sensor koblings-diagram.

Det er fullt mulig å koble sensorene i parallell for å kunne lade alle sensorene samtidig, uten å måtte øke spenningen fra 12 volt. Fra Figur 15 har vi visuell representasjon av sensorene koblet i parallell. Dette gjøres ved å koble alle negative kablene (blå) fra sensorene sammen og positive kablene (brun) fra sensoren sammen, henholdsvist. Dette gjelder også for polene på batteriet.



Figur 15: Sensorene koblet i parallell.

Denne sensoren tar i bruk en bryter som automatisk lukker seg når strålen er brutt. Dersom vi måler på svart ledning ser vi at ingen spenning går her når strålen ikke er brutt, som vist i Figur 16.



Figur 16: Måling uten brutt stråle.

Vi kan nå regne generelt hva resistorene i kretsen blir ved bruk av Formel 4.

$$U = IR \tag{4}$$

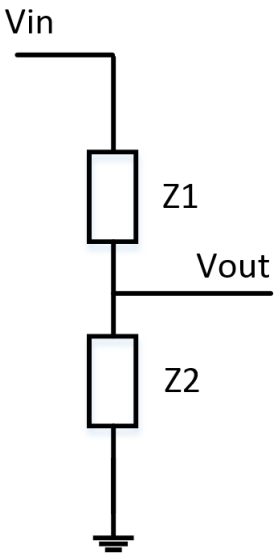
$$R = U/I$$

$$R = \frac{12V}{0,1A}$$

$$R = 120\Omega$$

Fra utregningene ser vi at motstanderen vil bli 120 ohm. Når vi testet dette systemet i praksis la vi merke til at resistorene blir veldig varme så dette var ikke aktuelt. Siden raspberry pi-en skal ha 1,8 volt til 5 volt tenkte vi å lage en spenningsdeler for å få så nærmest 3,3 volt som overhode mulig. I elektrolabben fikk vi låne noen varme resistante resistorer på størrelsen 3300 ohm og da kan vi beregne herifra gjenstående resistorer.

Kretsen for en spenningsdeler er vist i Figur 17 og går ut på å endre forholdet til resistorene for å endre forholdet til inngangs-spenningen.



Figur 17: Kretsen for en spenningsdeler

Ved å løse formelen for Formel 5 kan vi finne hvor stor resistor Z2 skal være for å kunne gi Vout en utgangsspenning på 3,3 volt.

$$Vout = \frac{Z2}{Z2 + Z1} \cdot Vin \tag{5}$$

$$(Z2 + Z1) \cdot Vout = Z2 \cdot Vin$$

$$Z2 \cdot Vout + Z1 \cdot Vout = Z2 \cdot Vin$$

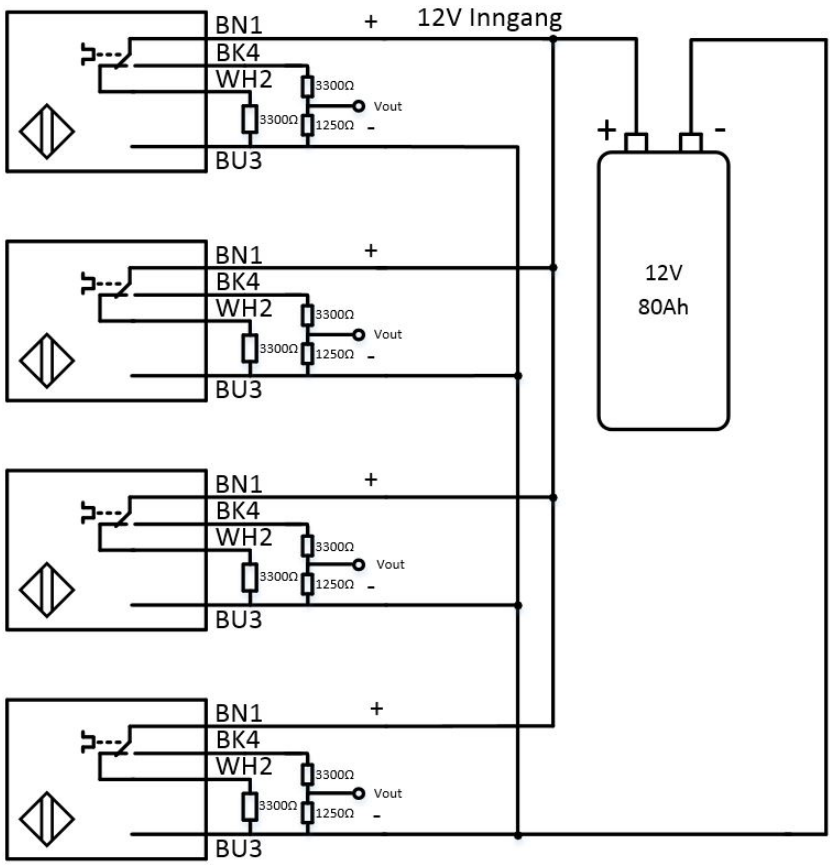
$$Vout \cdot Z2 - Z2 \cdot Vin = -Z1 \cdot Vin$$

$$Z2 \cdot (3,3V - 12V) = -10890\Omega V$$

$$Z2 = \frac{-10890\Omega V}{-8,7V}$$

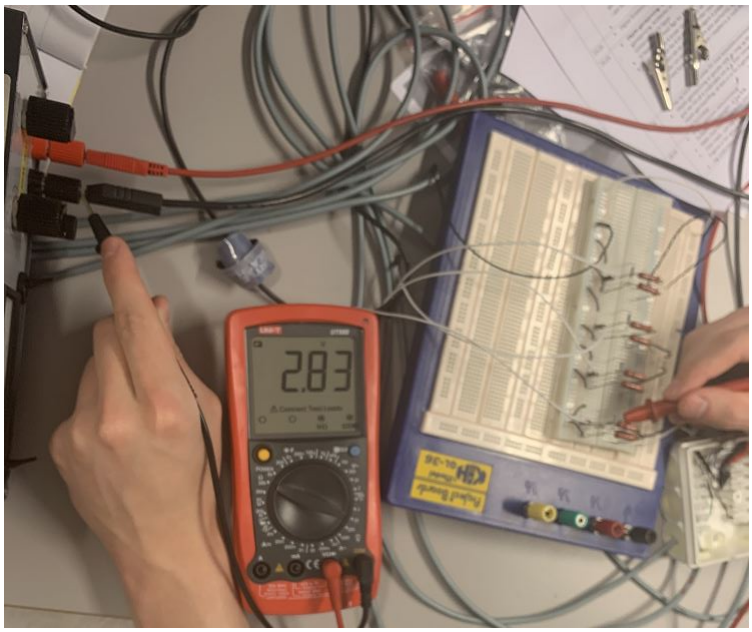
$$Z2 = 1250\Omega$$

Det aller nærmeste vi kommer til 1250 ohms resistorer er enten 1300 eller 1200 hvor vi bestemte oss for å ta 1200 ohms resistorene siden vi allerede hadde de tilgjengelig, dette vil endre utgangsspenningen vår litt men ikke nokk til å gjøre noen negative endringer. Kretsen vår vil deretter se ut som vist i Figur 18.



Figur 18: Sensor diagram med resistor verdier.

Når vi måler med multimeter på Vout får vi spenning 2,83 volt når strålen blir brutt, som sett i Figur 19 hvor vi har kontrollert spenning i praksis.



Figur 19: Måling med brutt ståle.

Vi målte 2,83 volt fra praksis målinger og 3,3 volt fra teoretiske beregninger. Fra Formel 6 får vi avviket mellom praksis og teoretiske målinger.

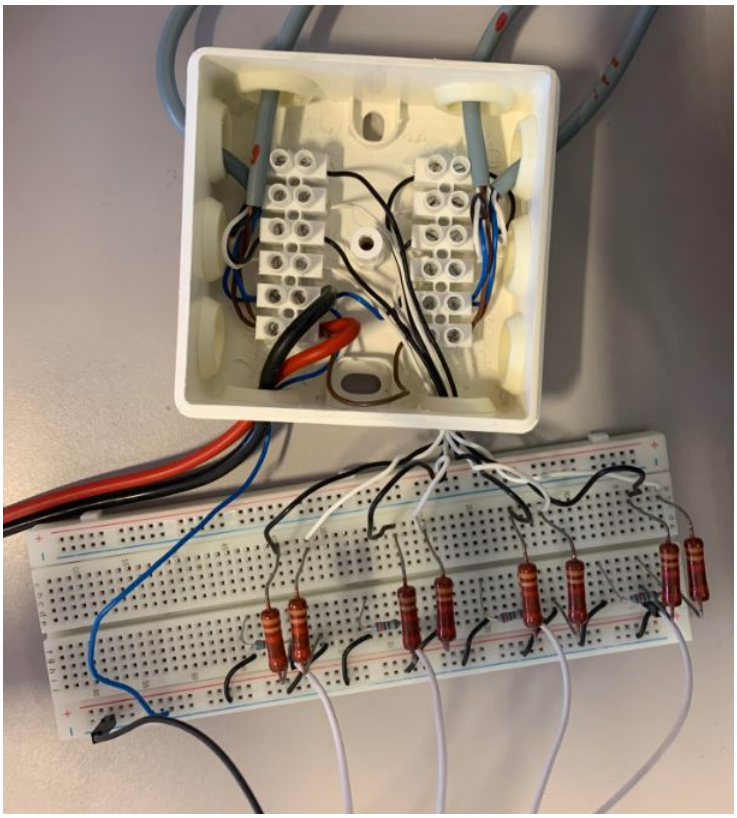
$$Avvik = \frac{Beregnetverdi - Måltverdi}{Måltverdi} \cdot 100 \tag{6}$$

$$Avvik = \frac{3,2V - 2,83V}{2,83V} \cdot 100$$

$$Avvik = 13,07\%$$

Vi ser at avviket er på 13,07% fra teoretisk til praksis målingene, men dette vil ikke endre noe i systemet vårt og er derfor tilstrekkelig.

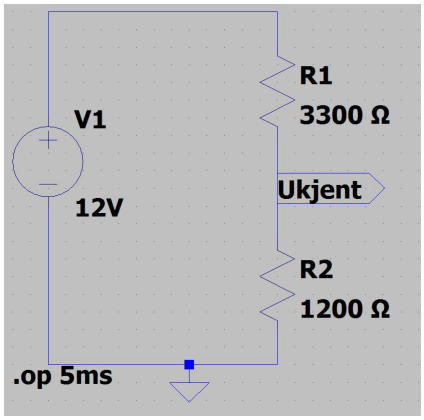
Måten vi valgte å koble kretsen i praksis var å trekke sensorene inn i en koblingsboks og til ett brødbrett som igjen gikk til raspberry pi-en.



Figur 20: Koblingsboks.

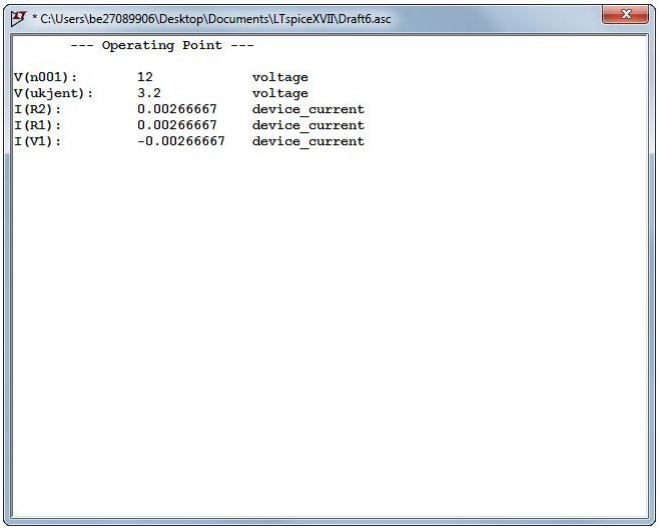
6.7.1 Simuleringer

Rett etter at vi hadde beregnet verdiene for systemet vårt valgte vi å simulere i LT-Spice. Her bruker vi en spenningsforsyning navngitt V1, R1 og R2 er 3300 ohm og 1200 ohm, henholdsvis. Simuleringene er vist i Figur 21.



Figur 21: Simuleringer av spenningsdeling.

Verdiene vi har fått av disse simuleringene er vist i Figur 22. Her ser vi at vi skal få omtrent 3,2 volt på den ukjente spenningsverdien.



Figur 22: Verdier av simulasjoner.

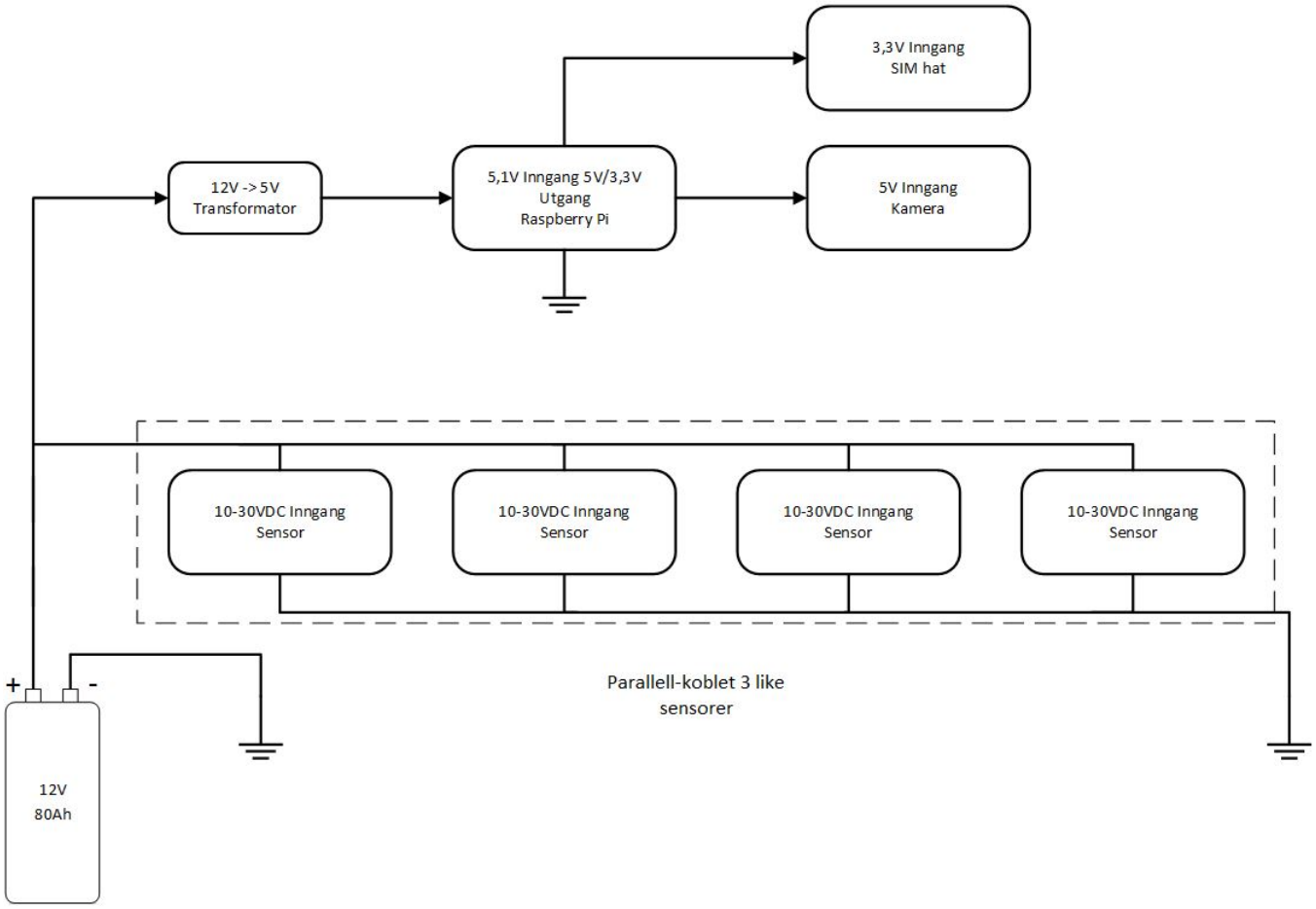
6.8 Batteri

Vi trenger ett batteri siden kravene fra veileder var at vi skulle lage ett system som var batteri drevet og kunne bli ladet opp. Oppladningssytemet vårt går ut på å bytte mellom to batterier. Vi har derfor bestemt oss for å kjøpe ett 12 volt, 480 ampere og 80Ah utladning batteri fra lokal butikken som er egnet for sol eller vindkraft (fritid). Med dette batteriet er det mulig å installere ett solcellepanel etter kundens behov. Det første vi kan gjøre med batteriet er å kontrollere og teste utgangsspenningen ved hjelp av ett multimeter, som vist i Figur 23.

Figur 23: Batteri kontrollering.

Dette gjøres ved å putte multimeter-et på 20 volt DC på pluss polen til batteriet og COM til negative polen til batteriet. Husk å start med å koble positive polen først og deretter negative, for å koble vekk fra batteriet koble så fra negative polen først. Spenning viser 12,6 volt som er tilstrekkelig for systemet.

I Figur 24 er det en illustrasjon over hva batteriet skal drive.



Figur 24: Batteri oversikt.

6.8.1 Lading

For å lade opp systemet vårt gjennom lange perioder har vi kjøpt inn en stort gjennopladbart batteri som er 12V, 480A med 80Ah utladning. Batteriet er laget for å bli brukt som en lagringskilde for sol- og vindkraft, dermed er batteriet designet for å kunne bli utladet og oppladet mye, sånn ca. 500 ladesykluser ved 50% utladningsdybde. Dette er mye bedre enn ett vanlig bil batteri, eller lignende, derfor er dette en av de beste batteri valgene til systemet vårt

Figur 25: Biltema fritidsbatteri.

Det vi mener med 50% utladningsdybde betyr at vi burde begynne å lade batteriet allerede etter at vi har brukt 50% av ladningen, for å ikke skade den. Det som er bra med dette batteriet er at det har en indikator på når batteriet må skiftes eller lades, det vil si at lyset på toppen lyser grønn når den er i stand, lyser oransje når den må lades og lyser hvit når

den må byttes ut sett i Figur 26.

Figur 26: Batteri indikator.

Batteriet lades ved å bruke laderen i Figur 27, og kobles ved å putte positive klypen på positive polen på batteriet og så negative klypen på negative polen. For å ta av klypene tar du bort først den negative.

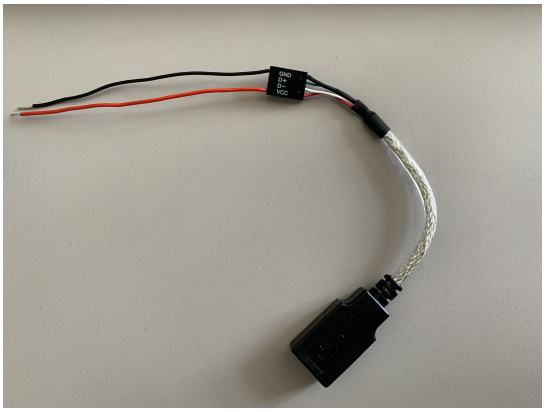
Figur 27: Laderen til batteriet.

6.8.2 DC-DC Converter

Batteriet gir oss en spenning på 12V, som ikke er aktuelt for raspberry pi-en. Siden batteriet også skal drive raspberry pi-en og kamera-et som går på 5 volt forsyning trenger vi en DC til DC omformer Det vi trenger for denne delen er DC til DC omformeren, sett i Figur 28. Vi trenger også en Female port med fire pins for å kunne overføre 5 volt spenningen til USB

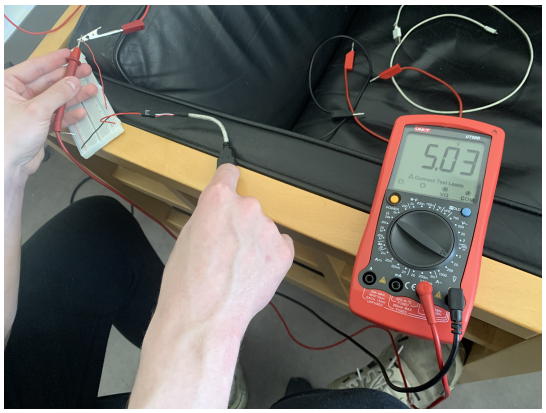
Figur 28: XP Power DC-DC Omformer.

porten, sett i Figur 29 med diverse ledninger.



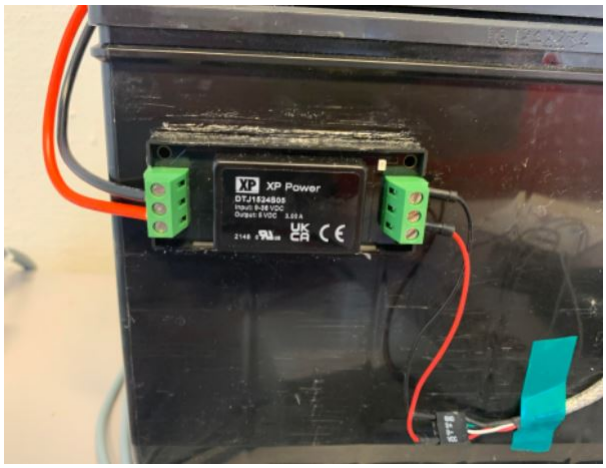
Figur 29: Female 4 pin port.

Det neste vi trenger da er en USB female port som vi også velger å lime til siden av batteriet for nyttighetssykld med fugemasse. DC til DC omformeren kobles til USB female porten, husk jordingen. Nå kan vi gjøre multimeter mål igjen og vi ser fra Figur 30 at vi får 5,03 volt spenning fra USB porten som er hva vi trenger.



Figur 30: Kontrollering av spenning i DC-DC omformer.

DC-DC omformeren er koblet og vist i Figur 31. Ved å gjøre det på denne måten blir det litt lettere å ha kontroll når systmet ligger ute i naturen.



Figur 31: XP Power DC-DC Omformer på batteriet.

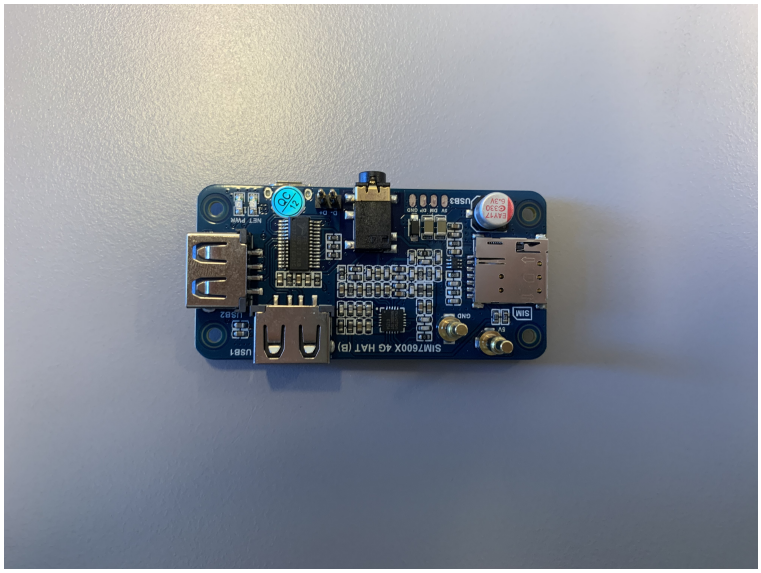
6.8.3 Bildebehandling

Raspberry pi-en hadde problemer med å kjøre maskinsyn programmet, den kjørte alene programmet på rundt 5-10 bilder i sekundet. Dermed tenkte vi at den trengte litt hjelp til bildebehandlingen, så vi kjøpte inn Neural Compute Stick 2 fra Intel. Denne ideen gikk ettersom at vi fant ut det var nesten umulig å kjøre real-time applikasjoner på raspberry pi-en vi bruker.

6.9 SIM7600G-H oppkobling

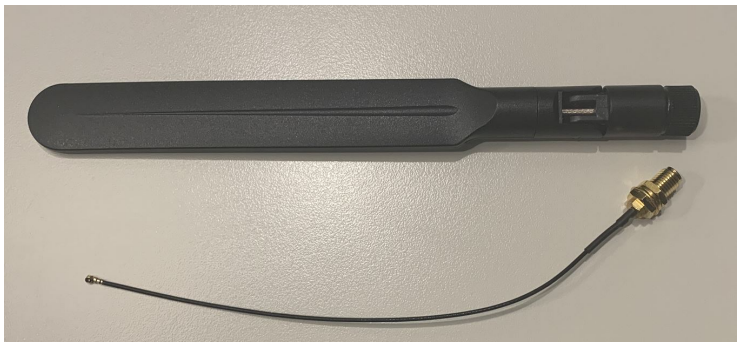
For å koble opp enheten vår er det vitalt og å ha alle komponentene på plass, det vil si at vi har på plass disse komponentene, sett i listen under:

1. SIM7600G-H 4G Hat Versjon (B). Dette er modulen som gir 4G til systemet, vist i Figur 32.



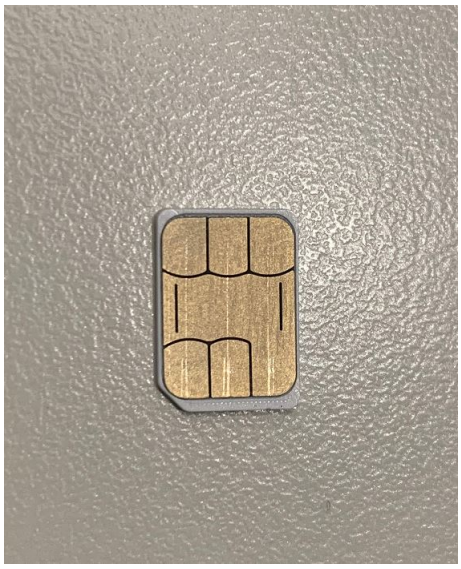
Figur 32: SIM7600G-H (B).

2. Antenne (Følger med i pakken.), vist i Figur 33. Her skrur vi antennepluggen enkelt til antenna. Antenna hjelper med å sende og motta internett når systemet er utendørs.



Figur 33: Antenne til SIM7600G-H.

3. SIM kort er hva som tilatter systemet å få 4G og kan fås hos internett leverandører som for eksempel Telia, vist i Figur 34.



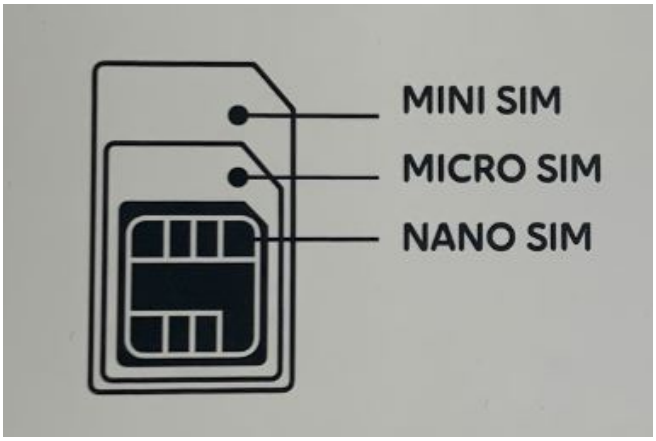
Figur 34: SIM-kort til systemet.

4. USB-A til Micro-B kabel, vist i Figur 35. Dette er bare en kabel som gir oss strømtilførsel til hatten.

Figur 35: USB-C kabel.

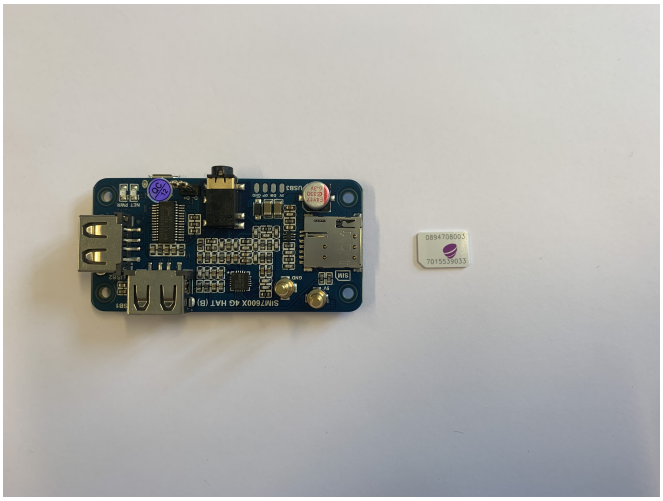
Når vi installerer systemet må vi aller først være sikker på at SIM hatten er av og at det ikke går noe strøm i systemet, følg Forhåndsreglene. For å sammenkoble antennen til SIM hatten må vi koble antenne pluggen til knappen angitt "MAIN". Vi må også koble USB-C kablen til strømtilgangen og USB portene til datamaskinen din. Hvordan man konfigurerer SIM hatten kan bli fulgt i stegene under:

1. Knekk ut det aktiverte SIM kortet ditt til størrelsen "Nano SIM" som vist i Figur 36.



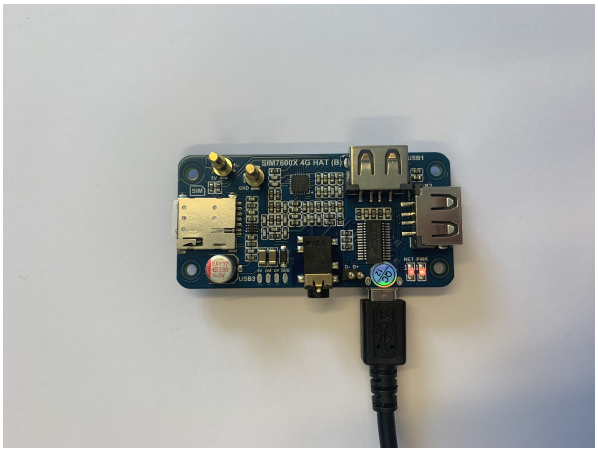
Figur 36: SIM kort størrelser.

2. Nå kan du putte SIM kortet i luken vist i Figur 37.



Figur 37: SIM kort luke.

3. Koble også til strøm med USB-C kablen som fulgte med i settet, som vist i Figur 38.



Figur 38: Strømtilkobling med USB-C kabel.

4. Last ned drivere for hatten ved å trykke på linken:
   Klikk her

5. Ekstrakt filene f.eks. på skrivebordet og kall filen "SIM7600G-H".

6. Åpne enhetsbehandling på PC-en din og se at du har flere udefinerte porter som heter "SimTech, Incorporated".

7. For hver av portene, i våres tilfelle, måtte vi oppdatere driverene individuelt som kan bli gjort ved å å trykke på hver av "SimTech Incorporated" og oppdatere driverene ved hjelp av Filen som allerede ligger på skrivebordet kalt "SIM7600G-H", og ikke nødvendigvis ved hjelp av nettet.

8. Når alle portene er oppdatere burde enhetsbehandlingen se ut som vist i Figur **??**

9. Avhengig av hvilken operativsystem du har, må du velge henholdsvis.

10. Vi kan nå få 4G på hatten ved å legge til nytt nettverk ved å åpne kontrollpanel.

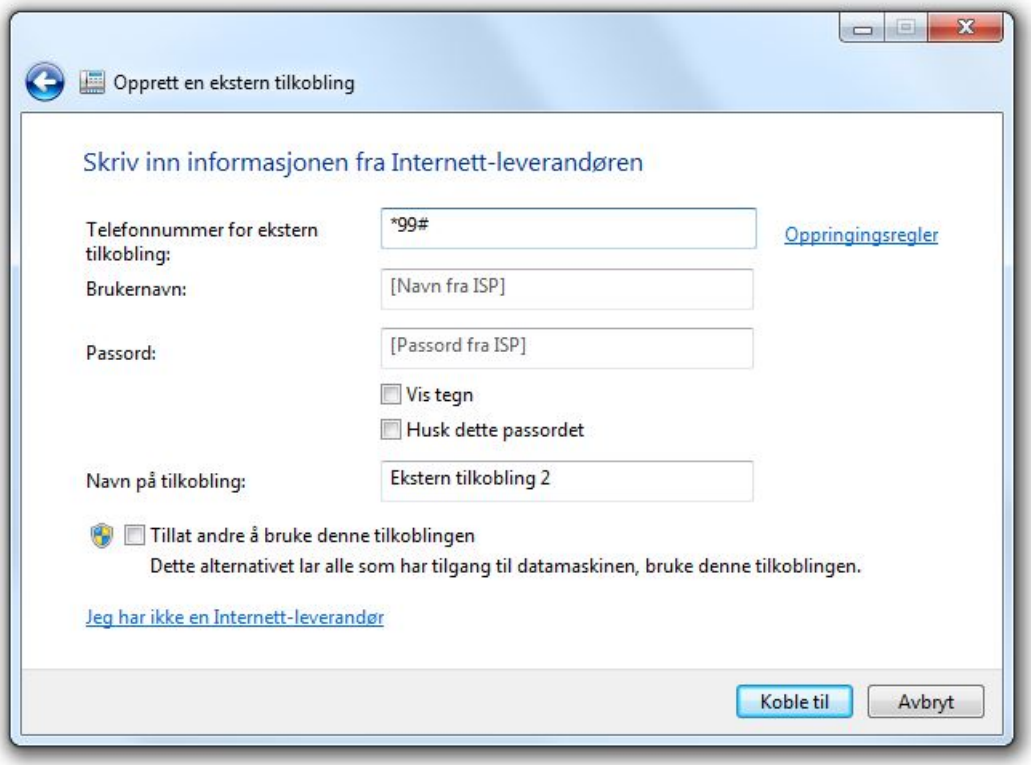
11. Heretter naviger til "Nettverk og Internett".

12. nå "Nettverks og Delingssenter".

13. Konfigurer ny tilkobling eller nytt nettverk.

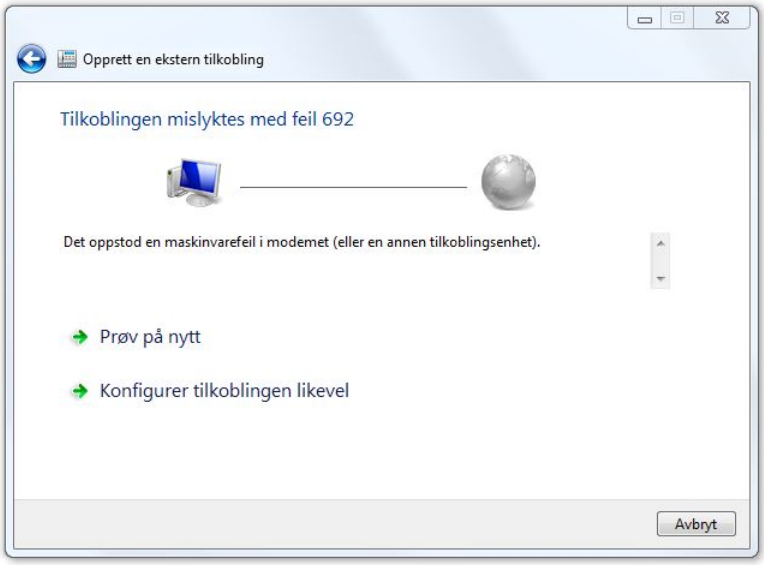
14. Konfigurer en ekstern tilkobling.

15. Skriv in telefonnummeret under "Telefonnummer for ekstern tilkobling:". Husk å skrive "*" før og "#" etter, vist i Figur 39.



Figur 39: Opprett en ekstern tilkobling.

16. Nå vil du få ett vindu som vist i Figur 40.

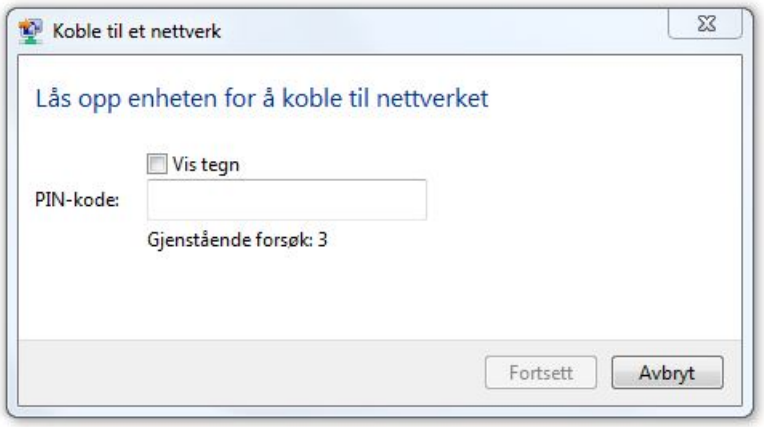


Figur 40: Konfigurer tilkobling.

17. Nå burde hatten være koblet til, men for å aktivere SIM kortet må vi gå til nettverkskoblingene vist i Figur 41.

Figur 41: Lås opp enheten.

18. Klikk på "Nettverk uten navn" og koble til med PIN koden til SIM kortet ditt, vist i Figur 42.



Figur 42: Strømtilkobling med USB-C kabel.

19. Nå skal nettverket fungere med driverene til SIM hatten.

Vi er nå ferdig med å sette opp hatten mot datamaskinen og vi kan nå fortsette med oppsettet til raspberry pi-en, og vi starter med å koble USB-C kabelen fra SIM hatten til raspberry pi-en.

1. Åpne terminalen til rapsberry pi-en og skriv inn "sudo apt-get install minicom".

2. Videre skriver "sudo minicom -D /dev/ttyUSB2.

3. Nå kan vi gjøre noen tester for å se at SIM hatten faktisk fungerer, og kan bli gjort ved å skrive inn noen kommandoer i minicom som: "AT" for å teste kommunikasjonen mellom SIM hatten og raspberry pi-en, "AT+CSQ" for å sjekke signalstyrken, "AT+COPS?" for å sjekke nettverket (denne skal gi feilmelding).

4. Vi kan nå prøve å låse opp SIM kortet ved å skrive "AT+CPIN=2103" hvor 2103 ville vært SIM kort pin koden din.

5. Dersom SIM hatten fungerer og blinker kan du hoppe over de neste stegene.

6. Dersom det ikke fungerer kan vi begynne med å åpne terminalen på nytt.

7. Skriv "ls /dev/ttyUSB2"

8. Skriv "sudo minicom -D /dev/ttyUSB2".

9. Skriv "AT+CUSBPIDSWITCH=9011,1,1".

10. Nå må du vente en en liten stund fordi SIM hatten begynner å jobbe og skal levere noen verdier.

11. Skriv deretter "ifconfig" for å få IP addressen

12. Skriv "sudo dhclient -v usb0"

13. Skriv til slutt "AT+CUSBPIDSWITCH=9001,1,1", guide fra https://www.waveshare.com/wiki/Rasp

14. Nå kan det hende at du må låse opp SIM kortet på nytt ved å skrive "AT+CPIN=2103" igjen i minicom fra terminalen.

15. Dersom dette ikke fungerer kan vi prøve en annen metode som krever at vi går tilbake til terminalen.

16. Skriv "minicom -D /dev/ttyUSB2" igjen.

17. Skriv "AT+CUSBPIDSWITCH=9001,1,1".

18. Skriv "ifconfig -a" for å sjekke at usb0 forsvinner.

19. Skriv "sudo su" for å få administrasjonstilgang.

20. Skriv "rmmod qmi_wwan".

21. Skriv "wget https://www.waveshare.com/w/upload/0/00/SIM7600_NDIS.7z" for å laste ned wwan0 drivere.

22. Skriv "sudo apt-get install p7zip-full -y".

23. Skriv "7z x SIM7600_NDIS.7z -r -o./SIM7600_NDIS".

24. Skriv "cd SIM7600_NDIS".

25. Skriv "sudo apt install raspberrypi-kernel-headers".

26. Skriv "sudo apt-get install –reinstall raspberrypi-bootloader raspberrypi-kernel".

27. Skriv "sudo su".

28. Skriv "make clean.

29. Skriv "make".

30. Skriv "ls".

31. Skriv "insmod simcom_wwan.ko".

32. Skriv "lsmod".

33. Skriv "ifconfig -a".

34. Skriv "sudo ifconfig wwan0 up".

35. Skriv "minicom -D /dev/ttyUSB2".

36. Skriv "AT$QCRMCALL=1,1".

37. Skriv "apt-get install udhcpc".

38. Skriv "udhcpc -i wwan0".

39. Skriv "ifconfig -a".

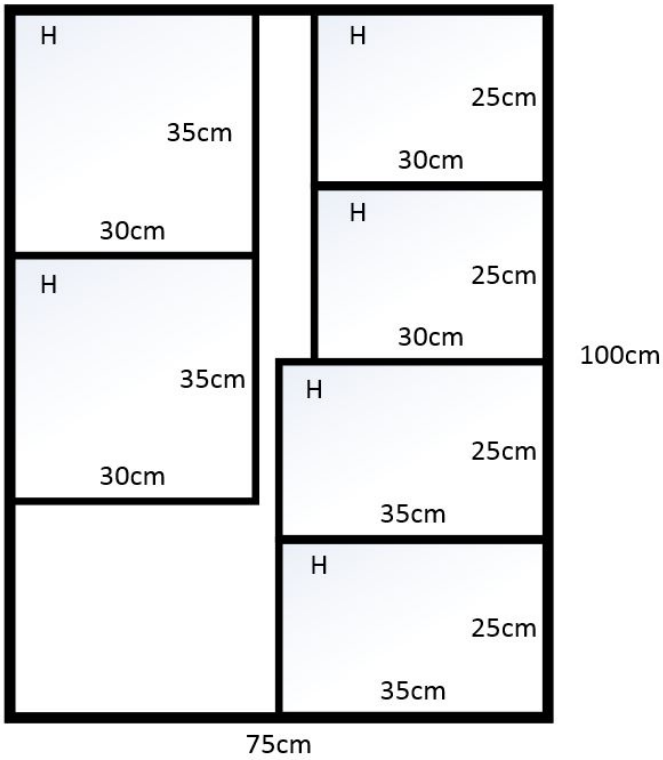
40. Skriv "ping -I wwan0 www.waveshare.com".

41. Skriv "route add -net 0.0.0.0 wwan0", denne guiden følger https://www.waveshare.com/wiki/Raspber

Da er SIM hatten klar for bruk og for enkelhetens skyld så velger vi å ta bort SIM kort koden ved å skrive "AT+CLCK="SC",0,"1234". Informasjon for innstallasjon er funnet i (12), men har vært veldig innviklet så jeg lagde ny.

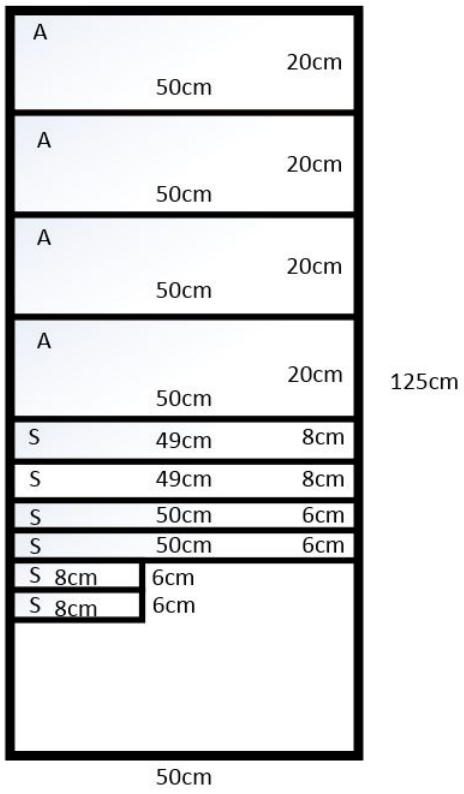
6.10 Vanntette bokser

Siden systemet vårt skal være under vann, tåle regn og andre naturfenomener må vi lage en løsning for dette. Vi benytter oss av akryl-plater til å lage vanntett sensor-boks, akvarium og hoved-boks til systemet. Alle boksene kan bli laget av plexiglass men siden den typen plastglass vi brukte tillater små mengder bøying er det greit når vi fyller boksene med vann. Akrylglasset har blitt kuttet opp med en veggmontert glasskjærer i ett rammeverksted. Vi endte med å bruke en plate på 75cm · 100cm og to plater på 50cm · 125cm. Hovedhuset til systemet vårt skal ha to plater med målene 35cm · 30cm, to plater med målene 30cm · 25cm og til to plater med målene 35cm · 25cm. Hovedhuset bestemte vi oss for å ha ett lukk som vi kunne åpne for å Akvariumet til systemet vårt skal ha fire plater med målene 50cm · 20cm og ett stort underlag med målet 50cm · 50cm som blir brukt til testing og expo. Den siste boksen er til sensorhuset som har to plater med målene 49cm · 8cm, to plater med målene 50cm · 6cm og til slutt to plater med målene 8cm · 6cm. Videre har platene blitt benevnet med forbokstavene H, A og S, for hovedhus, akvarium og sensorhus henholdsvis. Platen som er 100cm · 75cm er kappet til til målene vist i Figur 43.



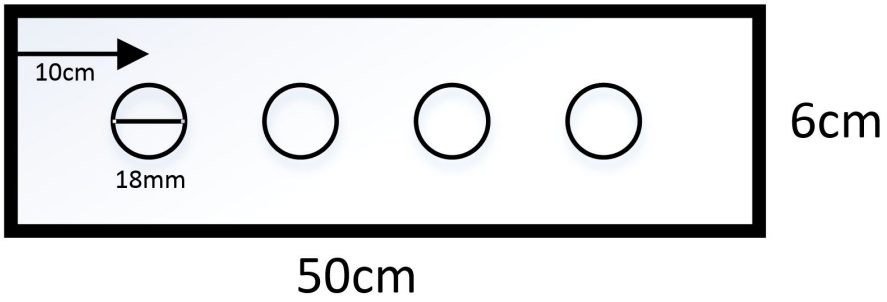
Figur 43: Plate med mål 100cm· 75cm.

Den neste platen som er 125cm · 50cm er kappet opp til målene sett i Figur 44.



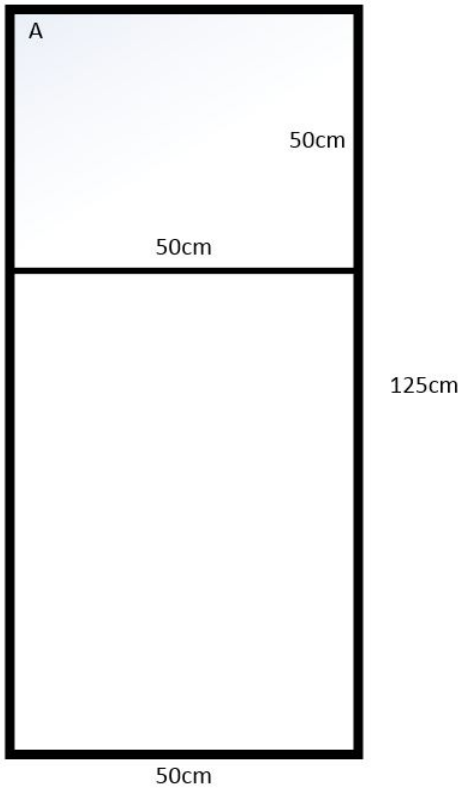
Figur 44: Første plate med mål 125cm· 50cm.

Den ene platen som har målene 50cm · 6cm skal ha hull drillet 10cm fra hverandre med 18mm diameter, sett tydelig i Figur 45.



Figur 45: Sensor hull.

Den siste platen som også er 125cm · 50cm er kappet opp til måle gitt under i Figur 46.

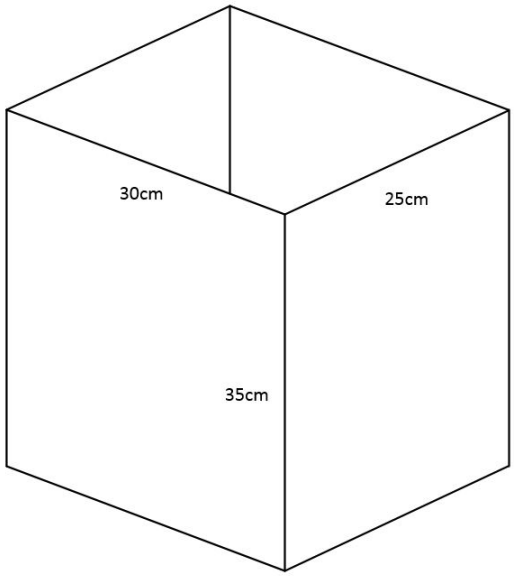


Figur 46: Andre plate med mål 125cm· 50cm.

Hovedhuset til systemet skal kun tåle vind og regn, akvarium-et skal tåle store mengder vann slik at vi kan teste og til slutt skal sensorhuset gjøre det mulig for oss å montere den i fisketrappen og beskyte sensorene. Dette vil naturligvis kreve mye fugemasse siden det skal vanntett. Ett lite tips er at det blir lettere å smøre fugemassen ut hvis vi har ett lag med fett i mellom, så f.eks. bruke litt grønnsåpe på fingerene.

6.10.1 Hovedhuset

Hovedhuset har vi valgt til å være rektangulær med ett lukk vi kan ta av for å inspisere eller endre systemet. Vi starter med å teipe og lage en rektangulær form med de to platene som er 35cm · 30cm og de to platene som er 35 · 25cm som vist i Figur 47



Figur 47: Hovedhus oppsett.

Nå fuger vi fast den ene 30cm · 25cm til bunnen av hovedhuset. Heretter trenger vi en måte å tette lukket som vi gjør ved å fuge innersiden av boksen og legge den på hode over en dag, dette vil skape en slags tette mekanikk som gjør den vind og regntett men ikke fullstendig vanntett når vi legger lukket på boksen. Vi bruker også to hengsler som er fuget fast i siden, sett i Figur 48.

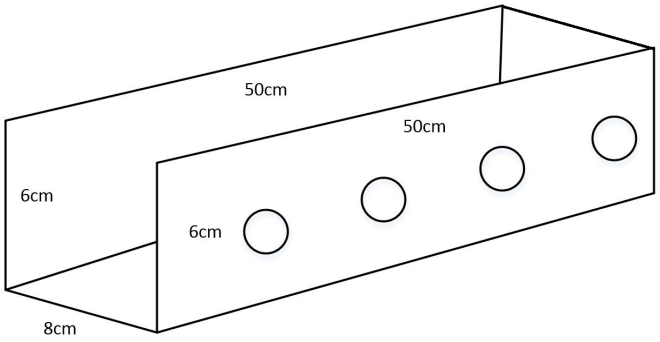
Figur 48: Hengsler til hovedhuset.

For lukket bruker vi bare en treplanke som går tvers over akrylglasset for å gjøre det enklere å gripe og for å ha noe å feste hengslene på. Sluttproduktet for hovedhuset er vist i Figur 49

Figur 49: Ferdig hovedhus.

6.10.2 Sensor

Sensorhuset har også valgt til å lage være rektangulær og en halvmeter lang slik at sensorene kan detektere over denne lengde, denne kan være justerbar til hvor mange sensorer du ønsker å bruke. Vi starter med å teipe sammen platene med målene 49cm · 8cm og de to platene med 50cm · 6cm til en U form og 8cm · 6cm platen i enden, sett i Figur 50.



Figur 50: Sensor oppsett.

Nå fuger vi sammen alle kantene og venter til det tørker. Neste steget er å montere sensor pluggene i platen med hullene, som vist i Figur 51

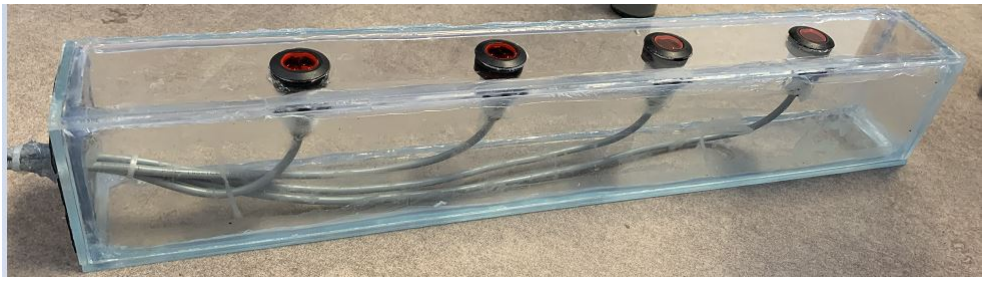
Figur 51: Sensor plugg montering.

Nå skrur vi sensorene inne i rammene vist i Figur 52.

Figur 52: Sensor montering.

Når disse fire hullene har tørket skjærer vi bort resisterende fug. Nå må vi drille ett hull i platen med målene 8cm · 6cm for å kunne trekke kablene igjennom. Dette gjøres ved å drille ett hull med diameter 15mm og deretter limes på. Til slutt har vi den siste platen 49cm · 8cm som limes for å fullføre rektangelet. Nå kan vi også lime på en mansjett som avlaster kablene i hullet til platen med 8cm · 6cm.

Figur 53 viser hvordan sluttproduketet av sensorhuset ser ut.

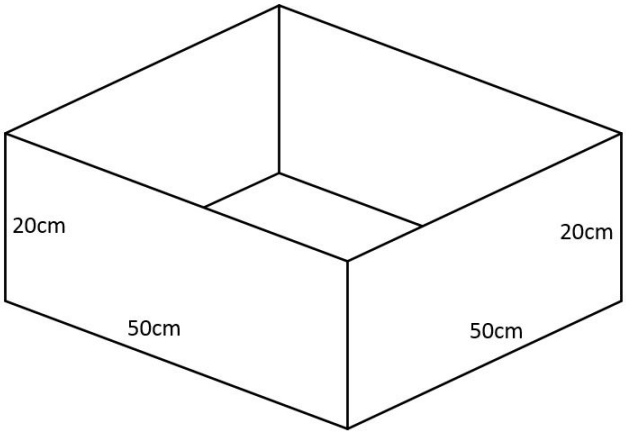


Figur 53: Ferdig sensorhus.

For å gjøre det mulig å montere sensorhuset til en fisketrapp fuger vi bare en planke i lengden av sensor-rammen, det vil si ca 60cm · 10cm i vårt tilfelle. Denne metoden kommer altså an på hvor stor sensorhuset er.

6.10.3 Akvarium

Akvarium-et har vi valgt til å være firkantet og skal tåle store vannmengder, så vi starter Akvarium-et har blitt lagd ved å kombinere alle 50cm · 20cm platene og deretter lage en ramme av de. Det gjøres enkelt ved å først teipe sammen side til side, og deretter påføre fugemasse for å feste rammene, vist i Figur 54.



Figur 54: Akvarium oppsett.

Når denne tørker legger vi rammen på den platen vi kappet til 50cm · 50cm og påfører nødvendig fugemasse i kantene. Sluttproduktet av akvarium-et skal se ut som Figur 57, hvor vi også har lagt til noen bakgrunner for å gjøre det enklere for deteksjonsprogrammet.

Figur 55: Ferdig akvarium.

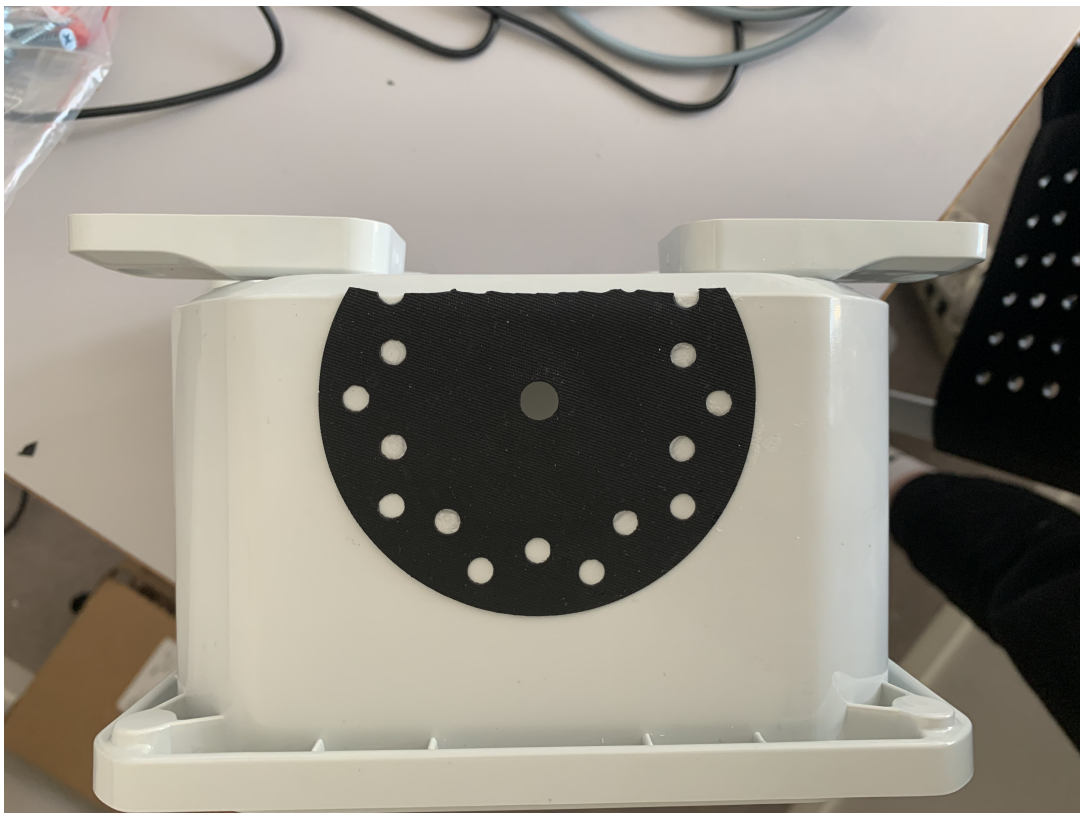
6.10.4 Kamera hus

Kamerahuset har blitt kjøpt fra Mouser og har dimensjonene 17.8cm · 17.8cm · 13.2cm. Dette er den siste komponenten som trenger hus. Vi bestemte oss for å drille hull i kamera

Figur 56: Kamera hus.

huset for å så kunne trekke USB kabelen igjen med ett mansjett for avlastning som sett i

Figur 57. Vi velger også å fuge dette i sammen for å så vanntette boksen.



Figur 57: Kameraboks hull.

7 Programvare

I dette kapitlet forklares vi teorien bak systemet, som trenges for å få til oppgaven. Vi gå grundig gjennom utfordringer og løsninger vi fikk gjennom software delen av oppgaven underveis. Vi forklares viktige begreper og oppgaver som vi lærte på veien mot å løse oppgaven. Vi starter først med å forklarer hva maskinlæring er og hva vi trenges for å kunne gjenkjenne fisk.

7.1 Maskin læring

Maskinlæring er et sett med algoritmer som blir mattet med strukturerte data for å utføre en oppgave som det er ikke programmert til å gjøre.

7.1.1 Maksinlæring algoritmer

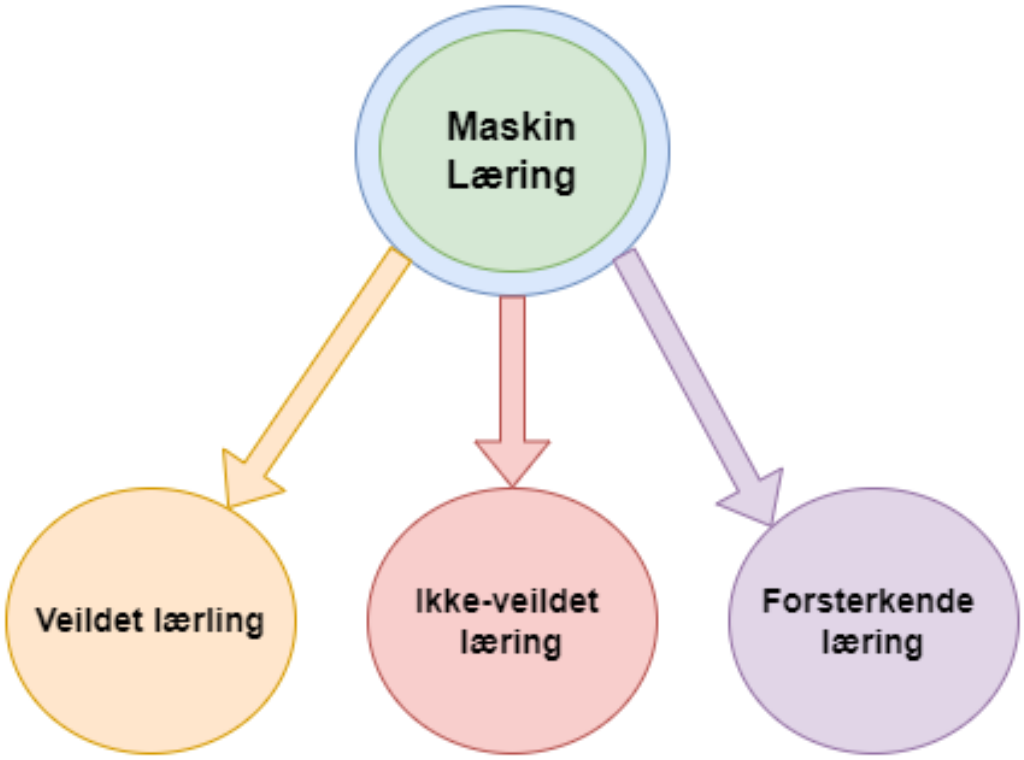
- **Veildet læring:**
  Veiledet læring er en type av maskin læring som lærer fra annotering data som ble annoterte av mennesker. Det er enklere, mer nøyaktig og pålitelig metode enn ikke-veiledet læring.**??**

- **Ikke-veildet læring:**
  I ikke-veiledet læring, maskinen lærer fra ikke annoterte data. Det blir ikke mattet som veiledet læring for å kunne lære. Istedenfor, lar modellen jobbe på egen hånd for å oppdage mønstre og informasjonen som tidligere var uoppdaget. Noen av ikke-veiledet læring algoritmer er clustering, anomali deteksjon. Et eksempel på ikke-veiledet læring er at maskinen sammenligner input data med tidligere dataene. Et annet eksempel er når vi ser en ting, så husker vi egenskapene til det. Neste gang vi ser, så gjenkjenner vi det fra egenskapen vi har sett tidligere.

- **Forsterkende læring:**

7.2 Dyp læring

Dyp læring er subset av maskinlæring som består av tre lag, input lag, output lag og mellom eller skjult lag.

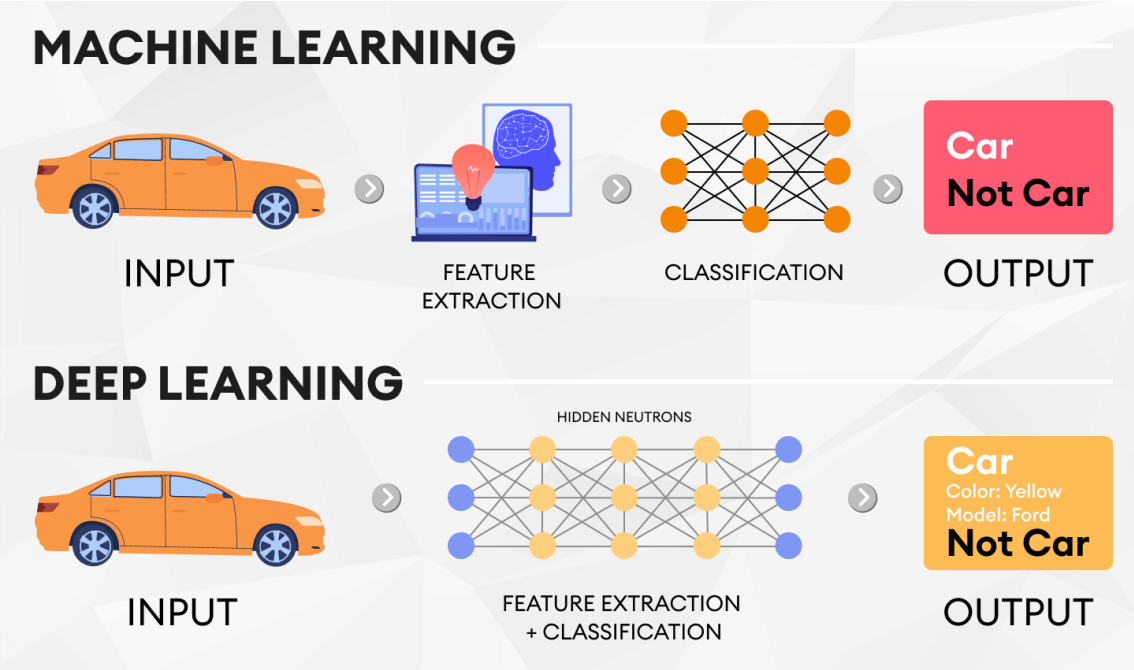


Figur 58: Maskinlæring algoritmer.

7.2.1 Dyplæring algoritmer:

Det er flere typer dyp læring algoritme og den vi bruker er konvulsjon nevrale nettverk(CNN). Det er en av det meste brukte og kjente i dyp læring algoritme.

- Forskjellen mellom maskin læring og dyp læring(13) Bildet **??** forklarer de meste viktige punktet i maskinlæring og dyplæring.



Figur 59: Maskin læring mot dyp læring.

7.3 Objekt Gjenkjenning

Objekt gjenkjenning er en data synteknikk som fokuserer på å identifisere objekter i bilder, videoer og live-opptak. Objekt gjenkjenning modellen er trent med annoterte data for å utføre denne prosessen med nye data. Hoved oppgaven er å identifisere objekter med firkant bokser rundt objektet.

7.3.1 Objekt gjenkjenning algoritme

Det er flere typer, men vi valgte å skrive litt om SSD og YOLO.

- **Single Shot Detector(SSD)**

- **YOLO:**
  YOLO står for **you only look once**, det er et toppmoderne objekt gjenkjenning

algoritme. YOLO algoritme kan gjenkjenne og oppdage ulike objekter i et bilde "In real time". Oppdagelse av objekter i YOLO er gjort ved regresjon problem og gir klassesannsynlighetene til de oppdagede bildene. Hastigheten å finne objektet gjør at det er blitt nesten standard til å oppdage objekter inn data maskinsyn feltet og YOLO gir nøyaktige resultater med minimale bakgrunns feil. Grunnen at YOLO er raskere enn de andre objekt gjenkjenning algoritme er:

  - Endrer størrelse av input data til 448x448. Bildet blir delt inn i ulike rutenett, hvert av rutenett har dimensjon på SxS.
  - Kjører et enkelt konvolusjonelle nevrale nettverk på bildet. Det betyr at den ser på bildet bare en gang.

  YOLO algoritmen består av ulike varianter, Noen av de vanlige er yolov3, yolov4 og yolov5.

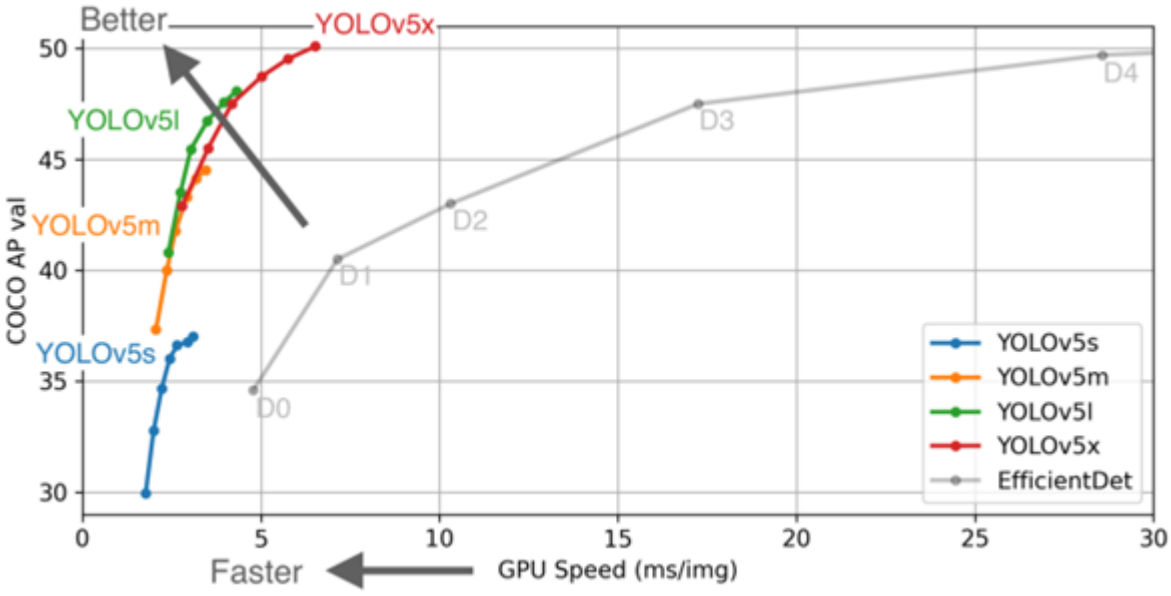
- **Yolov5** (14), Vi valgte å trene modellen vår med YOLOV5. Yolov5 er tilgjengelig i fem varianter, YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l og YOLOv5x. Hver av dem tilbyr forskjellige gjenkjenning nøyaktighet. Grunnen at vi trente modellen vår med YOLOv5 er at den mye mindre enn andre YOLO varianter. YOLOV5 vektfil er 27 megabyte, mens YOLOV4 vektfil er på 244 megabyte. Det betyr 90 prosent større enn YOLOV5.

7.4 Datasett

I sammenheng med objekt gjenkjenning et dataset er et sett av bilder med deres respektive annotering. For å trene opp en modell i Objektgjenkjenning, det er kjempe viktig å ha nok data for å kunne trene opp en god modellen. klargjøring av datasett kan skje i flere måter

- Samle masse bilder som finnes i internett
- Ekstrakter bilder fra videoer
- Ta bilder selv.

I våres tilfelle vi fant masse bilder av laks som vi brukte i å trene opp modellen hos nordfisk(15), og ekstraktet en god del bilder av video-en vi har fått av eksterne veilederen. Etter at vi har skaffet oss nok data, så begynte å annotere.

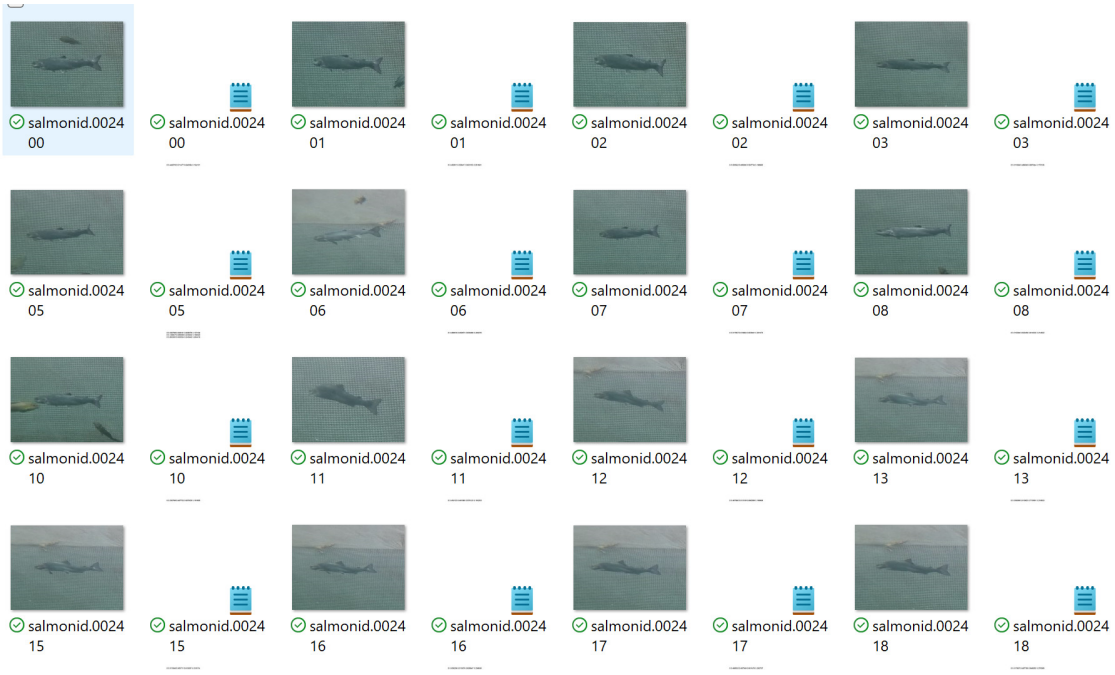


Figur 60: YOLOV5 varianter.

7.5 Annotering

Data annotering er en prosess å annotere bildene som blir brukt til å trene opp modellen. I en veiledet maskinlæringsmodell brukes annoterte data til å gi korrekt svar til modellen. Kvaliteten på annotering påvirker direkte på ytelsen av modellen. Det anbefales å ha minimum 1000 bilder for hver klasse om det er flere objekter du ville gjenkjenne og alle dataene må annoteres. Det fleste annotering program støtter YOLO format og gir en txt file per bildet.

Hver av txt file innholder en annotasjon for hvert objekt i bildet som er vist i Figur 61. Den første Tall 0 i figure62 er objekt klass id. For datasett vårt klass id 0 referer til klassenavn som er "laks". Resten av tallet indikerer xyhw koordinater som er markert rundt objekt i bildet.



Figur 61: Bilder med text file.

```
0 0.214217 0.337176 0.220941 0.394813
```

Figur 62: txt format.

7.6 Prototype

7.6.1 Transfer Learning

Transfer learning er en metode som brukes en forhåndstrent modell og adaptere den til å brukes på en lignene oppgaver, ved å finjustere parametere av den forhåndstrent modell. Transfer learning gjør det mulig å utvikle et kraftig modell som er bygget på arbeidet av experter innenfor maskinlæring(16). Selv om tensorflow og Yolov5 har gjort det enklere å utvikle modeller med transfer learning gjennom tensorflow object detection api og Yolov5, er det forstatt krevende. Det kreves mye forsking, grundige forberedelser, utvikling og testing. Tensorflow object detection api Vi startet å jobbe med TensorFlow object detection API for vårt prototype, fordi API-en har gjort prosessen av å finjustering en forhåndstrent modell enklere. For å kunne bruke API-en må vi endre litt på koden til filene som er allerede lagt ut for oss. For å utvikle et modell må vi følge stegene under.

- Samle bildene dine og annoter dem.
- Lag TF Record.
- Last ned en forhåndstrent modell.
- Lag en config fil.
- Tren modellen.
- Konverter modellen til TFlite.

Samle bildene dine og annoter dem Hvis man er heldig, finnes det allerede en dataset på nettet klar for bruk. Men i vårt tilfelle, det var vanskelig å finne et datasett av laks som vi syns er i riktige miljø. Å samle de riktige bilder og labler dem er en viktig steg innenfor maskinlæring prosjektet. Det finnes flere forskjellige metoder til å label en datasett, vi har valgt å bruke den mest nøyaktige og pålitelige metode som er in-house labeling. Her skal vi annoter hvert bilde manuelt, vi har valgt den metoden i følge KISS prinsippet. Vi måtte først finne en bildesamling av laks, vi fant en som inneholder 10 000 bilder. For hvert bilder vi må label dem, siden den her var vårt prototype valgte vi å annotere kun 1000 bilder gjennom lablimg, det er å merkere hvor i bilde objekte er og lagre det i et xml eller txt fil.

Lag TF Record

Tf records er en fil format som er en komponent av tensorflow. Den er en binært lagrings-format som har et stort innvirkning på ytelse av vårt importert pipeline(17). Det vil si at det tar mindre tid å trene modellen, binær data tar mindre plass på disken, tar mindre tid å kopiere og er mer effektiv til å lese fra disken.I tilegg, det gjør at vi slipper å tenke på å dele datasetet inn til en Train, Test og Val sets. Det er også viktig å blande det for ikke å ha noen partisk datadistribusjon. Det tar tid med å beholde filstrukturen og opprettholde tilfeldige datadistribusjon, men tfrecord ordner alt det der med et fil(18).

Last ned en forhåndstrent modell

Vi må laste ned en forhåndstrent modell, for å trene med vårt tilpasset data. Valg av modell er viktig, og har en stor betydning på ytelse av vårt tilpasset modell. Det finnes mange forskjellige modeller, og vi måtte finne hvilket modell er best for vårt applikasjon. Det er å gjenkjenne fisk på et raspberry pi. Så det er viktig at den er lett og kompatible med raspberry pi. Tensorflow tilbyr mange modeller som er klar for transfer learning innenfor

tensorflow model zoo tf2, her er modellen trent på coco dataset og er scoret på hvor raskt den er, hvor nøyaktig den er og hva slags output vi får(19). Fra utvalget har vi valgt for første prototypen å bruke SSD MobileNet V2 FPNLite 320x320. Ut ifra Figur 63 kan man se at den er en av de raskeste modellene, det er også en av de mest populær og brukt modeller, så vi har valgt å bruke den model til å starte med. Etter hvert vi har endret på modellene basert på yttesle på raspberry pi. For andre prototypen har vi benyttet SSD MobileNet V1 FPN 640x640 og for tredje prototypen har vi valgt og bruke yolov5 som er ikke inkludert med tensorflow model zoo tf2.

Model name	Speed (ms)	COCO mAP	Outputs
CenterNet HourGlass104 512x512	70	41.9	Boxes
CenterNet HourGlass104 Keypoints 512x512	76	40.0/61.4	Boxes/Keypoints
CenterNet HourGlass104 1024x1024	197	44.5	Boxes
CenterNet HourGlass104 Keypoints 1024x1024	211	42.8/64.5	Boxes/Keypoints
CenterNet Resnet50 V1 FPN 512x512	27	31.2	Boxes
CenterNet Resnet50 V1 FPN Keypoints 512x512	30	29.3/50.7	Boxes/Keypoints
CenterNet Resnet101 V1 FPN 512x512	34	34.2	Boxes
CenterNet Resnet50 V2 512x512	27	29.5	Boxes
CenterNet Resnet50 V2 Keypoints 512x512	30	27.6/48.2	Boxes/Keypoints
CenterNet MobileNetV2 FPN 512x512	6	23.4	Boxes
CenterNet MobileNetV2 FPN Keypoints 512x512	6	41.7	Keypoints
EfficientDet D0 512x512	39	33.6	Boxes
EfficientDet D1 640x640	54	38.4	Boxes
EfficientDet D2 768x768	67	41.8	Boxes
EfficientDet D3 896x896	95	45.4	Boxes
EfficientDet D4 1024x1024	133	48.5	Boxes
EfficientDet D5 1280x1280	222	49.7	Boxes
EfficientDet D6 1280x1280	268	50.5	Boxes
EfficientDet D7 1536x1536	325	51.2	Boxes
SSD MobileNet v2 320x320	19	20.2	Boxes
SSD MobileNet V1 FPN 640x640	48	29.1	Boxes
SSD MobileNet V2 FPNLite 320x320	22	22.2	Boxes
SSD MobileNet V2 FPNLite 640x640	39	28.2	Boxes
SSD ResNet50 V1 FPN 640x640 (RetinaNet50)	46	34.3	Boxes
SSD ResNet50 V1 FPN 1024x1024 (RetinaNet50)	87	38.3	Boxes
SSD ResNet101 V1 FPN 640x640 (RetinaNet101)	57	35.6	Boxes
SSD ResNet101 V1 FPN 1024x1024 (RetinaNet101)	104	39.5	Boxes
SSD ResNet152 V1 FPN 640x640 (RetinaNet152)	80	35.4	Boxes
SSD ResNet152 V1 FPN 1024x1024 (RetinaNet152)	111	39.6	Boxes
Faster R-CNN ResNet50 V1 640x640	53	29.3	Boxes
Faster R-CNN ResNet50 V1 1024x1024	65	31.0	Boxes
Faster R-CNN ResNet50 V1 800x1333	65	31.6	Boxes
Faster R-CNN ResNet101 V1 640x640	55	31.8	Boxes
Faster R-CNN ResNet101 V1 1024x1024	72	37.1	Boxes
Faster R-CNN ResNet101 V1 800x1333	77	36.6	Boxes
Faster R-CNN ResNet152 V1 640x640	64	32.4	Boxes

Figur 63: Liste med forhåndstrent modeller.

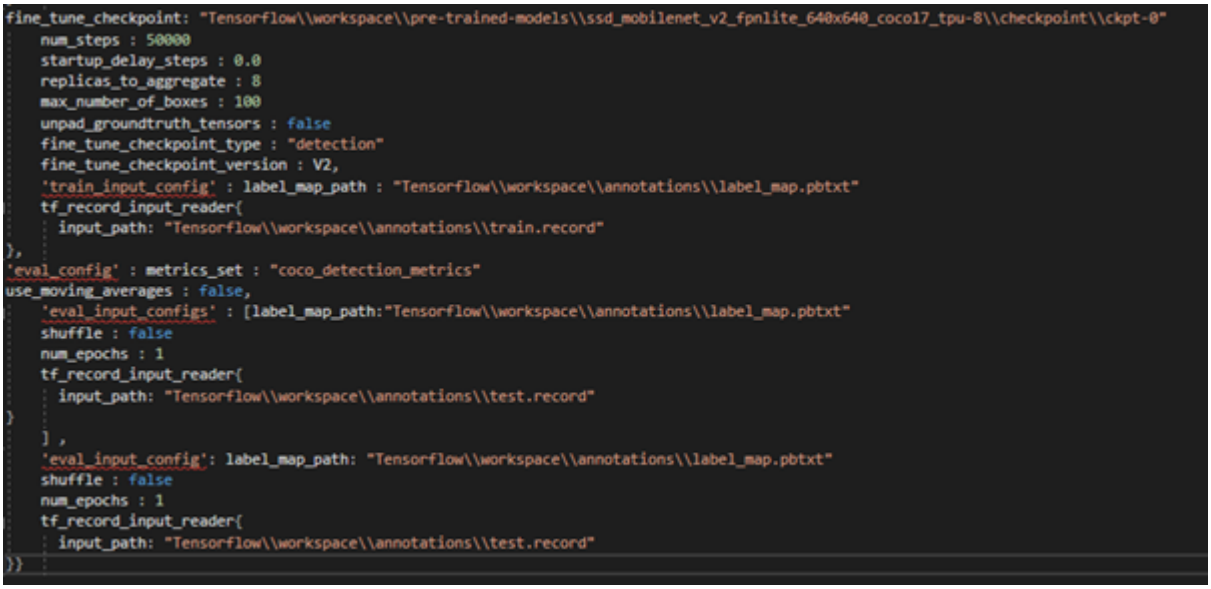
Lag en config fil Innenfor tensorflow object detection API(19), alt informasjon som er nødvendig for trening av modellen og evaluering finnes i en pipeline.config fil. Vi får dette fra API-en og vi må bare endre litt søkestier for å kunne trene og evaluere(20).

```
fine_tune_checkpoint: "PATH_TO_BE_CONFIGURED"

num_steps: 50000
num_classes: 2

fine_tune_checkpoint_type: "classification"

train_input_reader {
  label_map_path: "PATH_TO_BE_CONFIGURED"
  tf_record_input_reader {
    input_path: "PATH_TO_BE_CONFIGURED"
  }
}
eval_input_reader {
  label_map_path: "PATH_TO_BE_CONFIGURED"
  shuffle: false
  num_epochs: 1
  tf_record_input_reader {
    input_path: "PATH_TO_BE_CONFIGURED"
  }
}
```

Figur 64: Tensorflow Config file.

Tren modellen For å trene modellen, må vi først kopiere tf training python script. Treningen skjer gjennom kommandolinje, med to argumenter Model_dir og pipeline_config_path. Model_dir referere til den søkesti der trening prosessen vil lagre checkpoint filer. Pipeline_config_path referere til den søkesti der pipeline.config file er lagret. For å kjøre treningen bruke vi kommandoen:

```
python Tensorflow\models\research\object_detection\model_main_tf2.py
--model_dir=Tensorflow\workspace\models\my_ssd_mobnet_tuned
--pipeline_config_path=Tensorflow\workspace\models\my_ssd_mobnet_tuned
\pipeline.config--num_train_steps=1000
```



Figur 65: Custom Config file.

```
I0929 19:14:03.429471 140008397502336 model_lib_v2.py:652] Step 24300 per-step time 0.355s loss=0.056
INFO:tensorflow:Step 24400 per-step time 0.335s loss=0.047
I0929 19:14:37.807487 140008397502336 model_lib_v2.py:652] Step 24400 per-step time 0.335s loss=0.047
INFO:tensorflow:Step 24500 per-step time 0.337s loss=0.056
I0929 19:15:12.283904 140008397502336 model_lib_v2.py:652] Step 24500 per-step time 0.337s loss=0.056
INFO:tensorflow:Step 24600 per-step time 0.308s loss=0.052
I0929 19:15:46.930425 140008397502336 model_lib_v2.py:652] Step 24600 per-step time 0.308s loss=0.052
INFO:tensorflow:Step 24700 per-step time 0.361s loss=0.046
I0929 19:16:21.292615 140008397502336 model_lib_v2.py:652] Step 24700 per-step time 0.361s loss=0.046
INFO:tensorflow:Step 24800 per-step time 0.338s loss=0.048
I0929 19:16:55.302360 140008397502336 model_lib_v2.py:652] Step 24800 per-step time 0.338s loss=0.048
INFO:tensorflow:Step 24900 per-step time 0.373s loss=0.062
I0929 19:17:29.647791 140008397502336 model_lib_v2.py:652] Step 24900 per-step time 0.373s loss=0.062
INFO:tensorflow:Step 25000 per-step time 0.311s loss=0.054
I0929 19:18:04.469359 140008397502336 model_lib_v2.py:652] Step 25000 per-step time 0.311s loss=0.054
```

Figur 66: Eksempel på trening output.

For evaluering må vi kjøre den evaluering skript for å finne mAP (Mean Average Precision) og loss, den blir brukt for å evaluering yttesle av vårt model. For å kjøre evaluering script bruke vi kommandoen:

```
python Tensorflow\models\research\object_detection\model_main_tf2.py
--model_dir=Tensorflow\workspace\models\my_ssd_mobnet_tuned
--pipeline_config_path=Tensorflow\workspace\models\my_ssd_mobnet_tuned
\pipeline.config --checkpoint_dir=Tensorflow\works pace\models
\my_ssd_mobnet_tuned
```

```
Accumulating evaluation results...
DONE (t=0.02s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.833
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.911
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.859
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.800
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.838
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.567
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.873
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.894
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.800
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.896
```

Figur 67: Eksempel på evaluering output.

Konverter modellen til TFlite

Deretter skal vi frosne grafen, det betyr å lagre det siste lag av modellen. Grunnen for dette er fordi vi skal konvertere en frosne inference graph til TFlite. For å få til dette skal vi bruke exporter_main_v2.py skript (21). For å kjøre dette skal vi bruke kommandoen:

```
python Tensorflow\models\research\object_detection\exporter_main_v2.py
--input_type=image_tensor --pipeline_config_path=Tensorflow\workspace\models
\my_ssd_mobnet_tuned\pipeline.config --trained_checkpoint_dir=Tensorflow
\workspace\models\my_ssd_mobnet_tuned --output_directory=Tensorflow
\workspace\models\my_ssd_mobnet_tuned\export
```

For å få til en tflite version av modellen, som kan kjøres på raspberry pi. Finnes det to stadier, først er det å kjøre export_tflite_graph_tf2.py for å generere en tflite-friendly mellomliggende SavedModel. Deretter skal vi bruke Tensorflow Lite Converter for å konvertere SavedModel til tflite. Her skal vi kjøre to kommandoer den første er å eksportere Tflite inference graph og andre er for å konvertere til Tflite.

```
python Tensorflow\models\research\object_detection\export_tflite_graph_tf2.
py --pipeline_config_path=Tensorflow\workspace\models\my_ssd_mobnet_tuned\
pipeline.config --trained_checkpoint_dir=Tensorflow\workspace\models\my_ssd
_mobnet_tuned --output_directory=Tensorflow\workspace\models\my_ssd_mobnet_
tuned\tfliteexport
```

```
tflite_convert --saved_model_dir=Tensorflow\workspace\models\my_ssd_mobnet_
```

```
tuned\tfliteexport\saved_model --output_file=Tensorflow\workspace\models\my
_ssd_mobnet_tuned\tfliteexport\saved_model\detect.tflite --input_shapes=1,
300,300,3 --input_arrays=normalized_input_image_tensor--output_arrays='TFLi
te_Detection_PostProcess','TFLite_Detection_PostProcess:1','TFLite_Detectio
n_PostProcess:2','TFLite_Detection_PostProcess:3' --inference_type=FLOAT
--allow_custom_ops
```

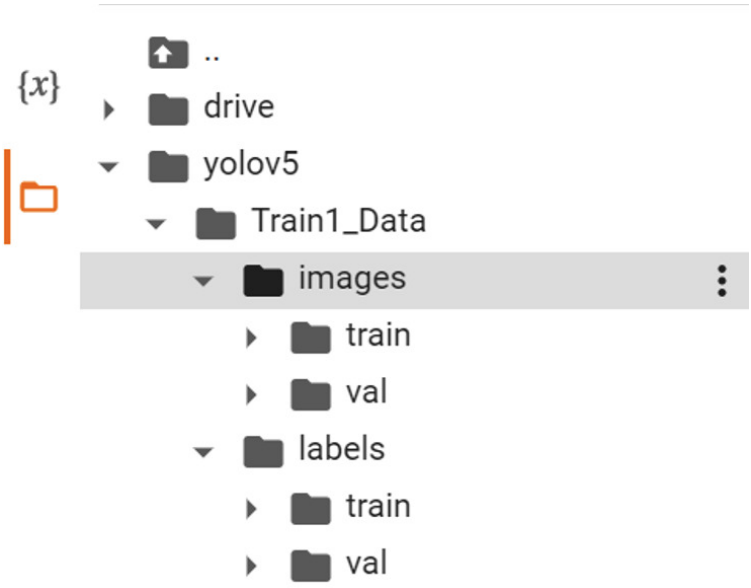
Testing av Tensorflow modellen Etter vi fikk kjørt Tensorflow-SSD modellen på raspberry pi, visste vi ikke hvordan vi kunne evaluering modellen. Vi hadde ikke noe til å sammenligne modellen med. Da valgte vi å trene en YOLOv5 modell med 150 bilder og 100 epoch, som er mye mindre data og weights enn SSD modellen. Vi så når vi testet begge modeller på samme video, at YOLOv5 modellen kunne gjenkjenne fisken mye bedre enn SSD modellen. Så vi valgte å jobbe videre med YOLOv5 modellen.

7.7 Trening av YOLOv5 modell

Før vi starter trening av modellen, vi trenger å lage en strukturen i filsystemet. Filstrukturen skal se ut som bilde i nedenfor. I hoved mappe lager vi to mapper til som hver av dem skal innholde to mapper i seg selv.

7.7.1 filstruktur

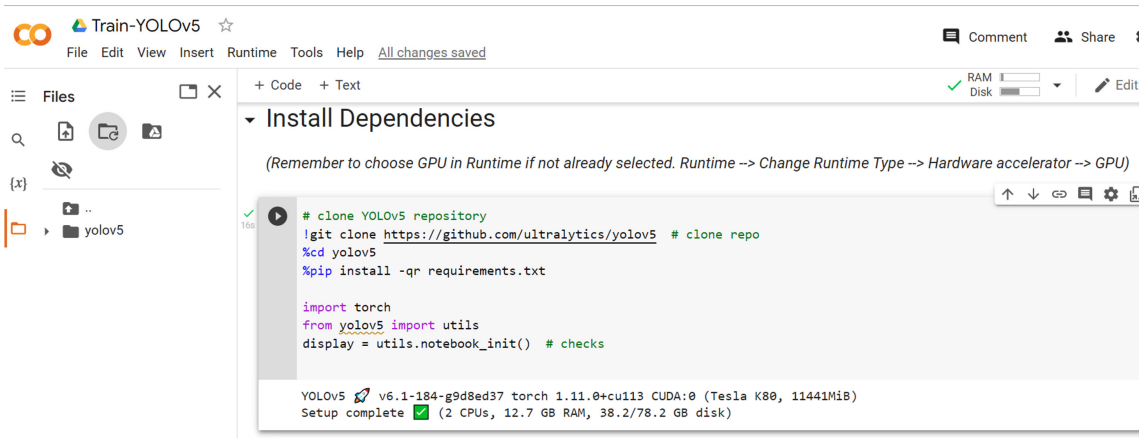
Strukturen av filen er kjempe viktig før trening av YOLOV5 modellen. Vi trenger å lage to mapper i hovedmappe som er vist i figur68, Train1_DAta, "labels" og "images", hver av de mappene skal ha to mapper inne seg som vi har vist det i bilde nedenfor. Alle text filene til våre data skal være i "labels" mappe og bildene skal være i "images" mappe. Det anbefales å ha 80% av dataene i "train" mappe og 20% i val.



Figur 68: Filstruktur

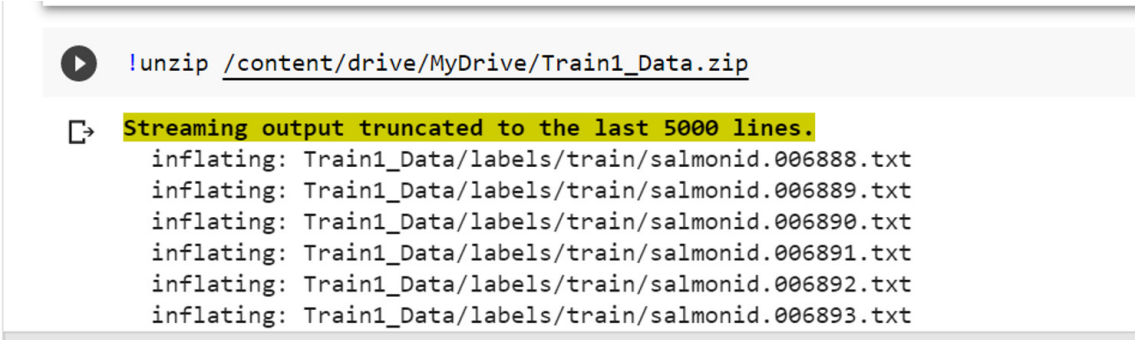
7.7.2 Google Colab

Datavitenskapen er stort og full av uendelige funksjoner, det er nesten umulig å lære hver av dem. For nybegynnere i datavitenskapen er en krevende jobb å velge en riktige verktøy som kan utføre jobben de trenger. Etter forskning på nettet, fant vi ut at google colab er en grei program å trene modellen vår i. Colab er en produkt fra Goolge research, hvor det lar oss trene våre maskinlæring og dyplæringsmodeller på CPU-er, GPU-er og TPU-er. Å bruker google colab alt vi trenger er å ha inertnet og Gmail konto, siden koder lagres i google drive. For å trene yolov5 modellen, vi må klone yolov5 fra github repo og installere kravene, det kan vi finne på bilde i nedenfor. Colab gir oss 12 timers koninuerlig utførelsetid. Etter 12 timer, mister vi alt vi trente og vi må start alt på nytt.



Figur 69: Google colab

Etter installasjon av alle krav, vi må gi google colab tilgang til datasett filen som vi har det i google drive. Datasett mappe som er google driver er zip-mappe format, den må pakke opp.



Figur 70: Unzip

Nå trenger vi å lage en tilpasset konfigurasjonen. (Det er kjempe viktig i å sørge skrive riktige filbane), for at hele trening prosessen er avhengig av denne filen. Linje 3 og 4 i Figur 71 viser hvor filene ligger og i linje 9 skrives det antall klasser, i våre tilfelle det er bare en klasse, da står det 1. linje 10 indikerer navnet til objektet vi har annoterte.

Før vi begynner med trening av modellen, Konfigurasjon filen må plassere i data mappen som er yolov5 mappe. Nå kopierer vi filbane til custom_data.yaml og limer rett etter data som vi ser nede i bildet. Se på Figur 72.
Det er noen begrepet i bildet ovenfor som er kjempe viktige å vite om.

custom_data.yaml

```
1
2
3 train: /content/yolov5/Train1_Data/images/train/
4 val: /content/yolov5/Train1_Data/images/val/
5
6
7 # Classes
8
9 nc: 1  # number of classes
10 names: ["Salmon"]
```

Figur 71: konfig fil

```
# train yolov5s on custom data for 100 epochs
# time its performance
%%time
%cd /content/yolov5/
!python train.py --img 640 --batch 16 --epochs 100 --data /content/yolov5/data/custom_data.yaml  --weights 'yolov5s.pt'
```

Figur 72: Skript for trening av modell

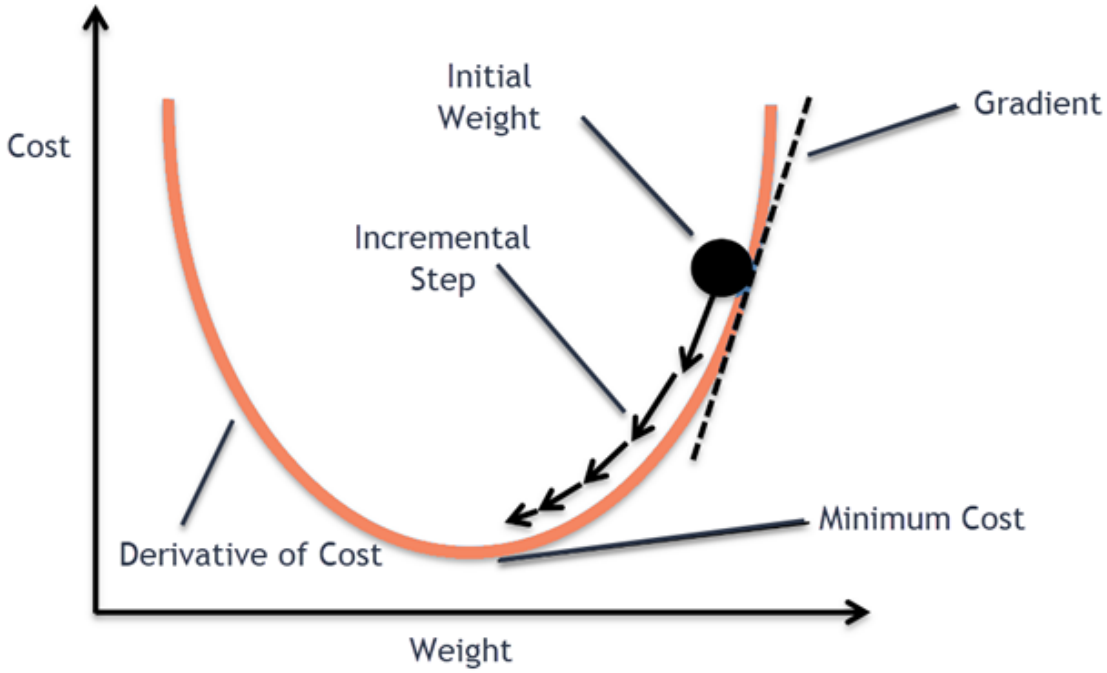
- **Image size:** størrelsen på bildene på hvilken modell som skal trenes, Standard størrelsen er 640.
- **Batch size:**
  Batchstørrelsen er antall prøver behandlet, før treningsmodellen er oppdatert. Antall batch størrelse kan være mer enn 1 og mindre enn antall av prøver i trening datasettet.
- **Epochs size:**
  Epoker er et begrep som brukes i maskin læring og den angir antall passeringer av hele datasettet som brukes i trening av modellen maskin algoritme har fullført. For å kunne få best mulig resultatet, så det er kjempeviktige å trene modellen med høyre batch og epoker størrelse.
- **Data:** Data er en yaml fil som innholder informasjon om datasett. Vi må skrive den riktige fil plassering i denne filen, for at trening av modellen er avhengig av denne filen.
- **Weights:**

7.7.3 Batch size

Når det er snakk om å trene modellen, det som jobber i bakgrunnen er noe som heter gradient descent som trener modellen vår. Gradient descent er en av meste populære optimalisering algoritmer for å trene maskinlæring modeller, ved å minimalisere feil mellom resultanten vi får og forventende resultat. I matematikk, optimalisering algoritme som gradient descent har oppgaven til å minimalisere/maksimalisere en gitt funksjon f(x) parametrisert av x. I vårt sammenheng er funksjonen den loss funksjon i nevrale nettverket. For å få til dette må vi iterere utførelse av to steger[(22),p.140-141].

1. Først er det å beregne gradienten eller stigningstallet som er den førsteordensderiverte av funksjonen for en gitt punkt.Se på Figur 73

2. Da flytter vi fra punktet i motsatt retning av gradienten der den stiger med en beregnet mengde.Se på Figur 73.



Figur 73: Gradient descent

En analogi vi kan bruke for å beskrive gradient descent er som følges. Tenk at du ønsker å klatre opp en bakke blindt. For å komme opp til toppen med færrest mulig skritt, kan du begynne å ta den største skritt mot den bratteste retningen, noe som du kan gjøre så lenge

du er ikke nær toppen. Det som kommer til å skjer etterværet er at du skal begynne å ta kortere skritt for å unngå å gå over toppen. Prosessen som foregår kan forklares matematisk med gradient. I vårt tilfelle, skjer det her i motsatt vei, fordi vi prøve å havne på bunnen. Prosessen kan blir gjort i tre forskjellige måter:

- **Batch Gradient Descent:**
  Før vi kan forklare batch gradient descent, er det viktige å vite noen stikkord. Sample/eksempel referere til en rad av data. Epoch er antall passeringer av hele trenings datasettet som algoritmen har fullført. Med batch gradient descent tar vi gjennomsnitten av gradientene for alle trenings eksempler og bruker deretter den gjennomsnittlige gradienten for å oppdatere parameterne våre. Så det blir en steg av gradient descent for en epoch(23).

- **Stochastic Gradient Descent:**
  Med batch gradient descent så vi på alle eksempler for alle steger av gradient descent. Problemet med det her kommer nå datasettet er stor. Dyp læring modeller kreves mye data, jo mer data, desto bedre ytelse modellen har. Med stochastic gradient descent tar vi et eksempel om gangen. For å for til en epoch i SGD(23):

  1. Velg et eksempel
  2. Overfør data til nevrale nettverket
  3. Beregne gradienten
  4. Bruk gradienten til å oppdatere weights.
  5. Gjør alle stegene for alle eksempeler innen treningsettet.

- **Mini batch gradient descent:**
  Det her er det vi har brukt, siden det her blir tilbudt når vi skal trene en Yolov5 modell. Mini batch gradient descent er bygget på både batch gradient descent og Stochastic gradient descent. Problemet med SGD er at den bruke kun et eksempel om gangen. Med det kan vi ikke implementere den vektorisert implementasjon. Som er ment for å øke hastigheten på prosessen. Med MBGD bruke vi ikke hele datasettet, eller kun et eksempel. Vi definere en batch som består av et bestemt antall av eksempler som er mindre en datasettet(23).

  1. Velg et mini-batch

2. Overfør data til nevrale nettverket
3. Beregne gradienten for mini-batchen
4. Bruk gjennomsnittlige gradienten til å oppdatere weights.
5. Gjør alle stegene for alle mini-batches som vi lagde.

Vi fant ut gjennom testing av modeller, at med høyere batch-size fikk vi bedre resultat. Vi startet med å trene mange modeller med batch-size 16, fordi vi trodde at vi kunne trene modellen raskere. Som vi hadde rett på, men problemet kom når det var snakk om ytelse. Da begynte vi å trene modeller med høyere batch-size, og den beste modellen vi har fått hittil er med batch-size 50 med 100 epochs.

7.7.4 Evaluering av modell

Innenfor Maskinsyn, en populær metrisk som blir brukt for objektgjenkjenning er mAP (mean average precision). Yolo bruker mAP til å evaluere modeler som har blitt publisert for forskning.
For å få mAP er det viktig å bli kjent med presisjon. Presisjon måler hvor presis din forutsigelse er, ved å måle prosenten der din forutsigelse er rett, Figuren 74 beskriver prosessen(24).

$$Precision = \frac{TP}{TP + FP}$$

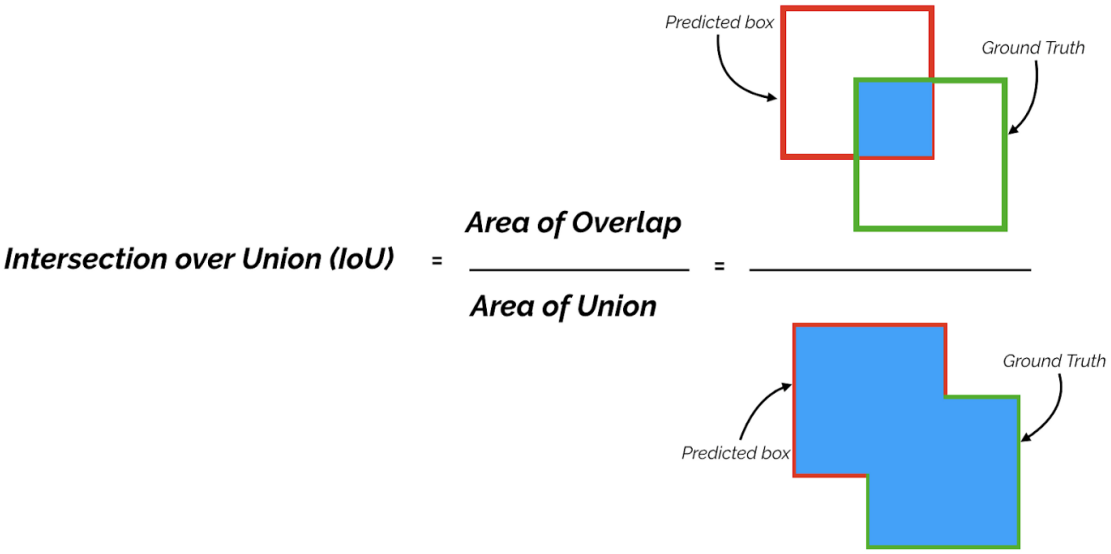
Figur 74: Formelen for å finne presisjon

Der TP er true posistives, det vil si at modellen forutsagt at det finnes objekt der og det var rett. Og FP er false positives, modellen forutsagt at det finnes objekt der og det var feil. Mean average precision er mer kompleks enn å bare kalkuerer gjennomsnitten av presisjoner verdier.For å finne map må vi bli kjent med andre metrikk i tilegg.

Først, innenfor objekt gjenkjenningssystemer er forutsigelse gjort gjennom en bounding boks og klasse label. Se på Figur 75

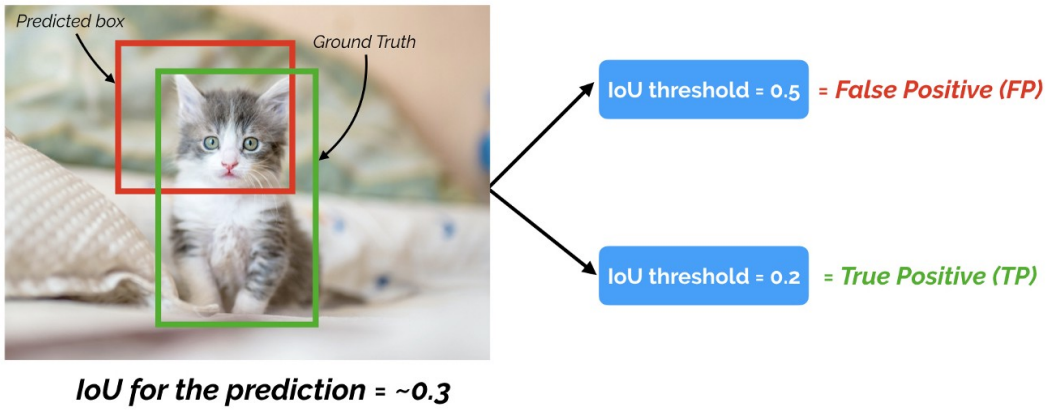
Figur 75: Bounding boks med klassnavn

For hver bounding boks må vi måle overlappen mellom den forutsagt bounding boks og den ekte boundary bokset er målt med IoU (intersection over union)(25).Se på Figur 76.



Figur 76: Formulen av IOU (26).

IoU har mye å si når det gjelder presisjonhet og tilbakekalling. Hvis man har en Iou terskel for = 0.5, og en boundary boks gir en forutsigelse med verdi IoU = 0.8, da blir den forutsigelse regnet som True Positive(TP). Men, hvis vi har en boundary boks med verdi IoU = 0.3 blir det regnet som False Positive(FP). Med å endre IoU terskel vi kan få noe som var tidligere regnet som FP til å bli TP. For eksempel, hvis IoU terskel = 0.3, da endrer vi forutsigelse fra FP til TP (24). Se Formel 77 for terskel.



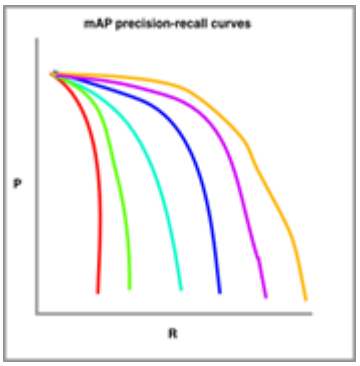
Figur 77: Ulike Terskel påvirkning på gjenkjennelse (26).

Spørsmalet tilbakekalling spørrer er hvor stor andel av de som er egentlige positive blir riktige identifisert. Formelen for å finne tilbakekalling vises under. FN er false negative, det vil si at modellen kunne ikke predikere at objektet var der. Se på Figuren 78.

$$Recall = \frac{TP}{TP + FN}$$

Figur 78: formulen av tilbakekalling

Da kan vi lage en presisjon-tilbakekalling kurve. Her kan vi finne den AP(average precision) med å finne området under kurven, og mAP blir da den gjennomsnitten av AP. For å finne mAP må vi tegnet en rekke av PR(presisjon-tilbakekalling) kurver med varierende IoU terskel for å få flere AP da tar vi gjennomsnitten av det(27). Se på Figuren 79.



Figur 79: Gjennomsnitt av AP

Overfitting:

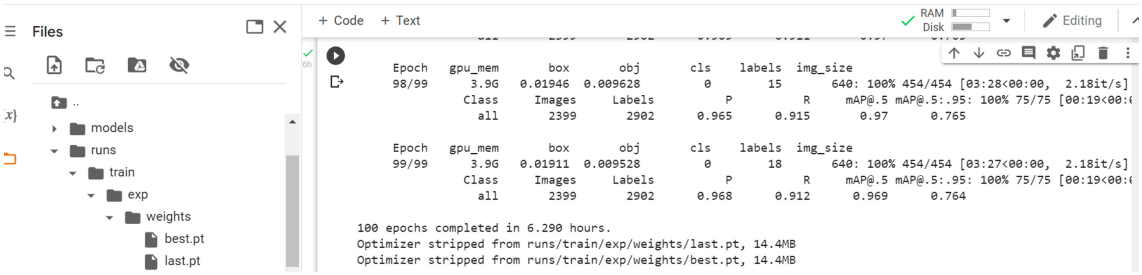
Det finnes to scenarier der vi må ha oppmerksomhet, en av dem er Overfitting. Det her er noe som skjer under trening av modellen. Der vår modell klarer å kjøre veldig bra på vår trenings data, men kjører dårlig på data som ikke er sett under treningen. Det vil si at modellen har klart å memorere trenings data istedenfor å lære forhold mellom fisken i bildet

og labellen(28). Det er enkelt å se når modellen er Overfitted, med å se gjennom trenings metrisk. Hvis nøyaktighet som er målt gjennom trening settet er veldig bra og validerings nøyaktighet som er målt gjennom validering settet er ikke bra. Det er en indikasjon på at modellen er Overfitted.

Underfitting:
Underfitting er det motsatte der modellen klarer ikke å huske forhold mellom fisken i bilde og labellen. Den generere en høy feilrate på både trening settet og validering settet(29). For å unngå dette må vi øke kompleksitet til modellen. Måten vi skal gjøre det er å øke antall weights og biases. Det vil si å øke antall lag og nummerer av nevroner i et nevrale nettverket(30).

7.7.5 Resultat

Etter fullføring av trening, det vi får er en best.pt og last.pt, filene er plassert i run/train/-exp/weights mappen. Videre, best filen brukes til å teste yolov5 modellen og bruke den i prosjektet om det er trent godt nok mens last.pt filen brukes til å trene videre om modellen er ikke godt nok til å bruke i prosjektet.



Figur 80: resultat av treningen

- **Best.pt**
  Best filen inneholder den beste vekt ytelsen som er lagret under trening
- **Last.pt**
  Last filen er vektene fra den siste treningsepoken.

7.7.6 Testing av modell

Det er ulike måter å teste modellen etter fullførte av treningen ved å bruke **detect.py** fil.

- Ved å bruke et bilde
- Video
- Webkamera

Modellen vår ble testet på bilder, videoer og webkamera for sikre at vår modell fullfører kraven. Bildet7.7.6 nedenfor er resultat av testing modellen på oppdage fisk på bilder.

Modellen vår oppfyller kraven vi har fått av arbeidsgiveren som var at systemet skal klare

Figur 81: **Oppdaget**

Figur 82: **Originale**

å oppdage fisk i en meter, men om det skal brukes til å oppdage fisk i over flere meter ville være en utfordring.

Utfordringen kan løses opp ved:

- Modellen må trenes videre med høyre batch størrelse
- høyre epoke
- Større datasett

7.8 Telling og sporing

Når vi startet å implementere telling, tenkte vi å bare telle antall detections som vi får i et video. Denne metoden hadde fungert bra, hvis vi hadde brukt bilder istedenfor video. Problemet vi får når vi prøve det på video, er at vi må kjører inference på hvert ramme. Det vil si at for hvert ramme vi får en ny detection, så for et video som har et objekt og varer i 5 sekunder. Kommer vi til å få tellingen på rundt 20, selv om det finnes bare et objekt i videon.Se på Figuren 83 84.

Figur 83: **Starten**

Figur 84: **Slutten**

For vårt system må vi kunne kjøre inference på en video, fordi vi klarer ikke å gjette når fisken er i ramme for å ta et bilde etter den blir oppdaget av sensorene. Løsningen på probelmet er å implementere objekt sporing først. Objektsporing er å estimere hvor et objekt ligger over en rekke av sammenhengende bilder (31).

En generell løsning for å spore er å bruke gjenkjennelser fra vår objektgjenkjenner der den prosessere gjenkjennelser for hvert ramme, som vi tenkte før. Men vi har lyst til å vite om et bestemt objekt er i bildene og for hvor mange rammer. Det vil si å vite x antall rammer der objektet er i scenen. Vi vil også bruke en matchende strategi for å kunne vite om objektet i forrige ramme er det samme som den i nåværende ramme.Se på Figur 85 86.

Figur 85: **Start posisjon**

Figur 86: **Slutt posisjon**

Vi skal bruke en prediktor for boundary bokser bevegelser. Vi må kunne predikere aspekter

på en boundary boks som tilsammen kalles vi for en tilstand. Tilstandene består av koordinater for den boundary boks center og farten. Jo mer rammer vi ser av the boundary boks, jo mer presis vi kan predikere de tilstandene til the boundary boks.

Når vi har en tilstand for en boundary boks kan vi predikere en mulige region for hvor objektet kan ligger i neste ramme. Vår objekt detector skal søke i det region, for å se om det ligger objekt som ligner på objektet som lå i forridge ramme, og det hjelpes til å oppdatere den nye tilstanden til boksen som blir brukt på neste. Vi iterere prosessen over alle rammer, resultatet er at vi får sporet et objekt og plasseringen av objektet over alle rammer. Denne metoden av sporing heter tracking by detection.Se på Figuren 87.

Figur 87: Tracking by detection (32).

I vårt system bruker vi en algoritme som heter deepSORT, det er bygget på den tidligere SORT (Simple Online and Real-time Tracking). Den har vist imponerende resultat på den MOT problemet (Multiple Object Tracking)(33). Se på Figuren 88.

Figur 88: MOT: sporing av flere objekter (34)

For **MOT** hvert ramme har mer en et objekt for å spore. En generelle løsningen er delt i to steger:

- Detection: Det første vi må gjøre er å detect alle objekter som er i ramme. Det kan være en eller flere detection.
- Association: Etter vi har fått alle detections for et ramme, da må vi match det til en lignede detections for det forridge ramme .

I **SORT**, er den generelle løsningen videre delt:

- Detection: Vi bruke vår custom yolov5 model for å kjører detections på første steget "X".
- Estimation: Den mellomliggende steg før Association består av en estimation modell. Kalman filter blir brukt for å estimate en tilstand vektor som består av noen parametere av vår mål. Nemlig, posisjon og fart som er basert på en dynamisk/measurement model. Hvis det er ikke noe detection av en sporingsobjekt over en viss terskel for sammenhengende rammer, blir det regnet som enten mistet eller ut av ramme(33).
- Association: Med Kalman filter blir det estimert den fremtidige lokasjon av objekt X +1. Det trenges til å bli optimalisert ved bruk av den korrekt posisjon. Det vil si å detect hvor objektet ligger i ramme X +1. Det blir løst ved å bruke den Hungarian algoritme.

Nå er det på tid til å innlemme deep learning teknikker for å gjøre SORT alogritmen om til deepSORT. Deep nevrale nettverket networks gjøres det mulig for SORT til å estimere den objektes lokasjon med mye høyere presisjon. Det er fordi de nettverkene kan nå beskrive trekk av målbildet.

DeepSORT med vår custom yolov5 modell, gjøres det mulig å kunne sporet fisken som kommer forby kameraet vårt. Nå kan systemet telle fiskene, med å telle antall detections(35).

Problemet vi fikk med denne metoden stammer fra noe som skjer i estimation steget: ***Hvis det er ikke noe detection av en sporingsobjekt over en viss terskel for sammenhengende rammer, det blir regnet som enten mistet eller ut av ramme***

I vårt tilfelle var vår terskel veldig kort. Under test, kunne vi se at når fisken blir detected får den en id. Hvis detector mister fisken, og den klarer å detect den før den terskel utløper, får fisken det samme id. Men hvis terskelen utløper da får den fisken en ny id. Det betyr at fisken kan fortsatt bli telt mer enn en gang, hvis detector er ikke perfekt. Vi hadde ungått det problemet hvis vi hadde et bedre modell. Sporing er knyttet til detector, hvis detector klarer ikke å detect fisken i hvert ramme, da kan ikke sporing gjør noe med det.

For å løse dette, bruke vi en ROI (Region of Interest). Vi lager en region på videoen, som teller fisker som passere det. I vårt tilfelle har vi et linje, og hvis fisken passere det linje blir det telt. Med det har vi minsket sjansen for å ha en feil telling.

7.9 Opplastning av data

7.9.1 MySQL

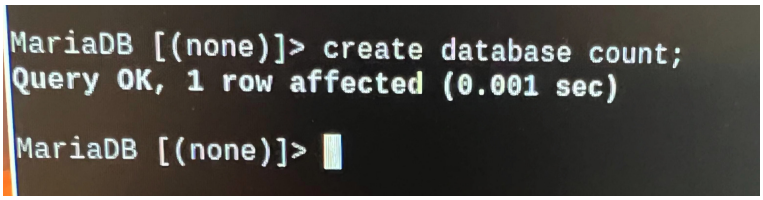
For at vi kunne lagre dataene våre, så trenger vi en database. en database er en applikasjon som lagrer en samling av data fra ulike kilder. I prosjekt vårt, lagering av dataene er en av hoved krav, dermed trengte vi å sette opp MYSQL i raspberry pi. Videre viser hvordan å sette opp mysql, phpmyadmin i raspberry pi-en.

1. Raspberry pi må oppdateres og oppgraderes
    - sudo apt update
    - sudo apt upgrade
2. Nå raspberry pi er klar for å installere MYSQL server
    - sudo apt-get install mariadb-server
3. MYSQL serveren er installert. nå må vi sikre den ved å angi et passord.
    - sudo mysql_secure_installation

Nå mysql er klar til bruk. for å overføre data inn i database, da må det lages database og tabeller. Det skjer på to måter. Det er enten vi lager direkte i kommandoterminal eller bruker vi phpmyadmin å komme inn in mysql database å lage database og tabeller der. Hvis

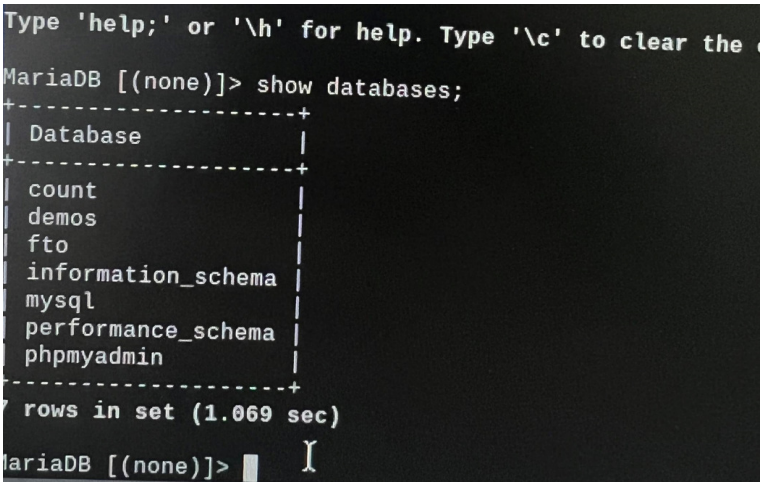
vi velger å gå gjennom phpmyadmin, da må vi laste ned apache. videre viser begge måtene å sette opp. Nå viser vi måten å lage database og tabeller gjennom terminal:

- følgende kommando-en ved pi terminalen gir oss tilgang til å lage, endre eller slette database.
  - Sudo mysql -u root -p
- Her valgte vi å bruke passord for å kunne få tilgang til serveren. Etter vi har fått tilgang, så har vi muligheten til endre, slette og lage nye databaser.



Figur 89: Lage database

- Nå har vi en database med count i MYSQL database. for å sjekke alle databaser. Nede i figuren vises at vi lagde to database som er count og fto.



Figur 90: Databases

Laging av database og tabeller direkte i phpmyadmin kreves at phpmyadmin skal være installert i raspberry pi-en. Installing av phpmyadmin skjer ved hjelp av denne kommando-en.

- Sudo apt-get install phpmyadmin

Vi testet sending av text filen til database ved hjelp av crontab som fungerte som den skal.

- Hva er crontab?
  Det er en list av kommandoer som vi vil kjøre på regelmessig etter tidsplan. Vi brukte crontab for å kjøre PHP scripten etter tidsplanen vi hadde lagt for å sende inn telle text filen til databasen. Med crontab vi kan velge hvor ofte scripten skal kjøre for å kunne sende inn data fra raspberry pi til databasen.

Etter masse arbeid med MySQL og apache fant vi ut lagring av videoer i en database er ikke smart. Det anbefales sterk å unngå å lagre videoer i database. Isteden valgte vi å jobbe å finne måten å lagre dataene vår i skylagring som dropbox, google cloud. Derfor valgte vi å jobbe videre med skylagring som dropbox og google drive. Det er to typer av data som vi trenger å passe på. Det er text file som innholder alle fisketallet og videoer.

7.9.2 Skylagrings

Data som vi har samlet på raspberry pi kan bli sendt og lagret på cloud. Det finnes mange skylagrings tjenester som har de nødvendige skylagrings API for at raspberry pi-en kan kommunisere. Den generelle metode for å bruke skylagrings på raspberry pi er:

- Å registrere for en konto hos en skylagringsleverandør.
- Konfigurer Raspberry pi for å kommunisere med skyen helst via å bruke metoden skylagrings API har.
- Samle inn data som Raspberry pien har prosessert.
- Koble til skyen og last opp dataene, helst med å bruke python.

Dropbox Dropbox er en av de meste kjente skylagring, og det var den første vi satt opp for raspberry pi-en. Dokumentasjon var enkelt for å registere, og for å konfiguere med raspberry pi for å kommunisere. Vi klarte å koble raspberry pien til dropbox, og den kunne laste opp dataene våres til riktige folders på dropbox. For å få til dette måtte vi generere en tilgangstoken, og sett den inn til koden. Der skal den kunne har fått permisjon til å koble til dropboxen for å kunne laste opp data.

Problemet kom når vi fant ut at den tilgangstoken varte kun i 4 timer. Det er noe som er ikke aktuelt for et system som skal være ute over lange perioder. Da fant vi ut at dropbox tilbyr flere typer av tilgangstokener, det vi hadde var en kortlevd tilgangstoken som dropbox har gjort til standard for sikkerhet. Langvarige tilgangstokener er det vi trengte, men dropbox har gjort det ikke tilgjenglige lenger. Det som man skal bruke for langsiktige tilgangstokener er å bruke refresh tokens. Vi gikk gjennom dokumentasjonen for å sette opp refresh tokens, men var forvirret på hvordan man kunne gjøre det gjennom python, ikke html. Etter en stund med å prøve å få til med refresh tokens. Valgte vi å prøve noe annet før vi brukte opp tida vi har satt opp for database.

Google Drive Google Drive var en annen alternative som vi valgt å jobbe med etter dropbox. Å sette opp er på samme måtte, forskjellgen er hvordan vi skal koble opp til driven fra raspberry pi. Det var lite dokumentasjon på hvordan man kobler opp, men når vi klarte til slutt. Så vi at det var faktisk mer simpel å jobbe med google drive, selvom det bar ikke mye dokumentasjon om det. Med google drive trengte vi ikke å generere tokens, som det er vist i koden koblingen til driven trengte ikke tokens. Så vi trodde at google drive brukte langvarige tokener. Men, etter en stund så vi at den brukte refresh tokener fordi når vi testet med å sende data. Var det en kall til å refreshe tilgangstoken, deretter ble data vår sendt. Til slutt, har vi valgt å jobbe videre med google drive, men vi har gjort koden som kobles til dropbox og database tilgjengelig for videre arbeid.

8 Testing

I denne delen av dokumentasjonen skal vi gjennomgå hvordan vi planlegger å teste produktet vårt, hvordan vi klassifiserer testene og rapportene av hvordan testene har gått.

8.1 Testplan

Testplanen beskriver hva slags tester som skal foregå og hvordan vi skal gå fremover med testing. Testplanen skal gi en god beskrivelse over kontrollstrategier som skal legges til grunn gjennom validering og verifisering. Testing brukes til å sikre og kontrollere at systemet oppfyller design og andre krav spesifikasjoner. Til sist skal alt dokumenteres og legges til i test rapporten.

Tabell 22: Test metoder.

Funksjons test:	Hovedfunksjonene blir testet, kontroll av feiltilstander, test for verste tilfelle.
Inspeksjons test:	Visuell testing og dokumentasjon.

Testing blir gjennomført for å evaluere at systemene samsvarer med kravspesifikasjonen. En abstraksjon av testing er å dele opp kravene inntil to test metoder. Innenfor hver test metode er det flere teknikker og verktøy som kan bli brukt. Test metodene er funksjons test og inspeksjonstest og er forklart i Tabell 22.

8.1.1 Funksjonelle og ikke-funksjonelle krav

For å teste funksjonelle og ikke-funksjonelle krav innenfor systemet har vi settet opp to tabeller som vi har kategorisert med test nummer, krav nummer, test type og test metode. Funksjonelle tester er en test på oppførselen til systemet og funksjonene til systemet. I Tabell 23 har vi settet opp tester av funksjonelle krav med gjenkjennings nummer slik at vi kan lett finne fram til riktig krav og test hvis vi skal referere til de i fremtiden.

Tabell 23: Test av funksjonelle krav.

Test nr.	**Krav nr.**	**Test type**	**Test metode**
T01	FK01	Funksjons test	Kjøre test av gjenkjennings modellen på fisk for å kunne gjenkjenne laks på en gunstig måte.
T02	FK02	Funksjons test	Kjøre test for å se hvor godt systemet teller fisk som har blitt tatt bilde av.
T03	FK03	Inspeksjons og funksjons test	Sette opp sensorene og måle avstanden de detekterer på.
T04	FK04	Inspeksjons og funksjons test	Verifisere at batteriet lader opp resten av systemet.
T05	FK05	Inspeksjon og funksjons test	Verifisere at oppladning systemet er gunstig og lett oppladbart.
T06	FK06	Inspeksjons og funksjons test	Kjøre test om systemet klarer å sende live bilder til UI/database.
T07	FK07	Inspeksjons og funksjons test	Test systemet med tre fiskearter for å se om de blir gjenkjent.

I Tabell 24 har vi settet opp testingen av kravene som ikke er funksjonelle, også kjent som de ikke-funksjonelle kravene. Generelt er disse testene på komponentene sine bruksområder og hvordan de går i drift, ikke hvordan de spesifikt oppfører seg eller de gitte funksjonene til systemet.

Tabell 24: Test av ikke-funksjonelle krav.

Test nr.	**Krav nr.**	**Test type**	**Test metode**
TI01	IFK01	Inspeksjons test	Utføre en undervanns test for huset.
TI02	IFK02	Inspeksjons test	Utføre en spyle test på hovedsystem huset.
TI03	IFK03	Inspeksjons og funksjon test	Teste avlastningen på system husene slik at ledningene sitter godt.
TI04	IFK04	Inspeksjons og funksjons test	Komponent test for at det ikke er feil i komponentene.

8.2 Test rapport

I denne seksjonen av rapport skal vi vise alle testrapportene vi har gjort gjennom prosjekt perioden sammen med resultatene og konklusjonene vi har kommet til ved testingen av systemet vårt. Vi skal teste kravene vi har lagt opp til i Kravspesifikasjon som er seksjon 2.3 av dokumentet, altså funksjonellekrav, ikke-funksjonellekrav og tekniskekrav. Disse kravene skal bli testet via to testmetoder, funksjonstest og inspeksjonstest som er snakket om i Test planen i seksjon 7.1. I Tabell 25 har vi lagt opp til måten disse testene skal dokumenteres.

Tabell 25: Testrapport mal.

Test navn:	
Test nr.:	
Test ansvarlig:	
Test deltakere:	
Dato:	
Utstyr:	
Godkjennings krav	
Feilmelding:	
Konklusjon:	
Resultat:	

8.2.1 Tester

I denne delen av rapporten skal vi vise frem alle testene som har blitt gjort i prosjektperioden.

Tabell 26: Test av sensorene 1.

Test navn:	Sensor test 1
Test nr.:	TI04
Test ansvarlig:	Jafar Alami
Test deltakere:	Daniel A. T. Poverud, Gard Johan Tollefsen, Jafar Alami, Mustafa Hassan
Dato:	02/05/2022
Utstyr:	Strømforsyning, 4x sensorer, 3x motstander, brødbrett, kabler og multimeter.
Godkjennings krav	Alle sensorene som blir testet individuelt skrur seg på når de detekterer bevegelse.
Feilmelding:	Ingen
Konklusjon:	Alle sensorene fungerer hver for seg.
Resultat:	Godkjent.

I Tabell 26 gjorde vi en lett komponent test for å passe på at sensorene ikke hadde noen produksjon feil.

Tabell 27: Test av sensorene 2

Test navn:	Sensor test 2
Test nr.:	T03
Test ansvarlig:	Jafar Alami
Test deltakere:	Daniel A. T. Poverud, Gard Johan Tollefsen, Jafar Alami
Dato:	02/05/2022
Utstyr:	Strømforsyning, sensor, 3x motstander, brødbrett, kabler, målebånd og skrutrekker.
Feilmelding:	Sensorene blir veldig unøyaktig om 85cm blir oversteget.
Konklusjon:	Sensorene kan detektere nøyaktig opptill 85cm, lengere er ikke optimalt men det fungerer.
Resultat:	Godkjent

I Tabell 27 har vi testet rekkevidden på sensorene siden den målene vi fikk fra oppdragsgiver var at den er vanligvis 1 meter høy.

Tabell 28: Test av sensorene 3.

Test navn:	Sensor test 3
Test nr.:	TI04
Test ansvarlig:	Jafar Alami
Test deltakere:	Daniel A. T. Poverud, Gard Johan Tollefsen, Jafar Alami
Dato:	03/05/2022
Utstyr:	Strømforsyning, 4x sensor, 12x motstander, brødbrett, kabler og multimeter.
Godkjennings krav	Alle sensorene skrur seg på når de detekterer bevegelse.
Feilmelding:	Ingen
Konklusjon:	Alle sensorene fungerte samtidig, og skrudde seg på når de detekterte bevegelse.
Resultat:	Godkjent

I Tabell 28 bare testet vi at de fungerte når vi koblet dem i parallell.

Tabell 29: Test av trådløst internett.

Test navn:	Trådløst internett
Test nr.:	TI04
Test ansvarlig:	Jafar Alami
Test deltakere:	Gard Johan Tollefsen, Jafar Alami, Mustafa Hassan.
Dato:	25/04/2022
Utstyr:	Datamaskin, SIM7600G-H (B) og 3G/4G/5G SIM kort.
Godkjennings krav	SIM7600G-H (B) Skal i første omgang gi oss tegn til å kunne kobles til internett.
Feilmelding:	Generelt bare "AT COMMAND ERRORS".
Konklusjon:	Glemte å låse opp SIM kortet i minicom-programmet som vi bruker, etter det fikk vi tilgang til internett.
Resultat:	Godkjent

I Tabell 29 vi testet SIM hatten på en datamaskin slik at vi vet den fungerer.

Tabell 30: Test av oppladning system.

Test navn:	Oppladning system test
Test nr.:	T04 og T05
Test ansvarlig:	Jafar Alami.
Test deltakere:	Daniel A. T. Poverud, Gard Johan Tollefsen, Jafar Alami.
Dato:	16/05/2022
Utstyr:	Batteri, DC-DC Converter, 4 pin header, kabler, Raspberry Pi og 4x sensorer.
Godkjennings krav	Vi skal få en output av 5 volt fra vårt 12 volt batteri for å lade Raspberry Pi, og skal.
Feilmelding:	Vi får feilmeldingen "Low Voltage Battery" av Raspberry Pi-en siden den ønsker 5.1 volt optimalt, samtidig som vi inspiserer
Konklusjon:	Raspberry Pi-en får 5 volt fra oppladning systemet og sensorene får nødvendig strøm.
Resultat:	Godkjent

I Tabell 30 testet vi at oppladning systemet fungerte med resten av systemet via deltester.

Tabell 31: Test av gjenkjennings modell.

Test navn:	Modell test
Test nr.:	T01
Test ansvarlig:	Jafar Alami.
Test deltakere:	Mustafa Hassan, Jafar Alami.
Dato:	15/04/2022
Utstyr:	Ågevideo.mp4 (video som ble sendt fra oppdragsgiver), Best100.pt modell
Godkjennings krav	For å ha kravet godkjent må vi ha null falskt positiv.
Feilmelding:	Ingen
Konklusjon:	Vi har nå en modell som kan gjenkjenne fisk, med null falskt positiv på vår test video.
Resultat:	Godkjent

I Tabell 31 kjørte vi en test med fiske gjenkjennings modellen for å passe på at den gjorde jobben riktig.

Tabell 32: Test av fisketelling.

Test navn:	Fisketelling test
Test nr.:	T02
Test ansvarlig:	Jafar Alami.
Test deltakere:	Mustafa Hassan, Jafar Alami.
Dato:	01/05/2022
Utstyr:	Test video(Test videoen viser en fisk som passeres et linje), Yolo$_{d}eepSORT_{P}ytorch$
Godkjennings krav	Skal telle fisken etter den har passert den designerte linjen.
Feilmelding:	Ingen.
Konklusjon:	Vi har minsket sjansen for å telle samme fisken mer enn en gang, ved å implementere sporing og en ROI også kjent som Region of Interest.
Resultat:	Godkjent

I Tabell 32 testet vi modellen som teller fisk, og den var gunstig.

Tabell 33: Test av skylagring.

Test navn:	Skylagrings test
Test nr.:	T06
Test ansvarlig:	Jafar Alami.
Test deltakere:	Mustafa Hassan og Jafar Alami.
Dato:	06/05/2022
Utstyr:	Data, UploadToGdrive. Py script
Godkjennings krav	System skal klarer å sende data til Google Drive automatisk.
Feilmelding:	Ingen.
Konklusjon:	Systemet klarer å sende bildet fra appliksjon til Google Drive automatisk.
Resultat:	Godkjent

I Tabell 33 har vi testet at bildene av fisk blir lastet opp til Google Drive etter de har blitt opptelt.

Tabell 34: Test av flere fiske arter gjenkjenning.

Test navn:	Flere arter fisk
Test nr.:	T07
Test ansvarlig:	Jafar Alami.
Test deltakere:	Mustafa Hassan, Jafar Alami.
Dato:	15/05/2022
Utstyr:	Modell
Godkjennings krav	Skal klare å gjenkjenne fisk og å klassifisere artene ørret, abbor og gjedde.
Feilmelding:	Har ikke gjort det.
Konklusjon:	Vi hadde ikke tid til å lage en modell som gjenkjenner tre arter fisk.
Resultat:	Ikke godkjent

I Tabell 34 bare nevner vi at vi ikke kan fullføre testen siden vi hadde ikke tid.

Tabell 35: Test av kamera hus.

Test navn:	Hus test 1
Test nr.:	TI01 og TI03
Test ansvarlig:	Jafar Alami.
Test deltakere:	Gard Johan Tollefsen, Mustafa Hassan og Jafar Alami.
Dato:	19/05/2022
Utstyr:	Webkamera og kamera hus.
Godkjennings krav	Kamera huset skal være vanntett og ledningene skal sitte godt.
Feilmelding:	Ingen.
Konklusjon:	Kamera huset ble fylt med vann, ikke noe vann lakk ut av huset og ledningen satt godt.
Resultat:	Godkjent

I Tabell 35 testet vi lett hvor vanntett huset er, samt at ledningen er passende avlastet.

Tabell 36: Test av sensor hus.

Test navn:	Hus test 2
Test nr.:	TI01 og TI03
Test ansvarlig:	Jafar Alami.
Test deltakere:	Daniel A. T. Poverud, Gard Johan Tollefsen, Mustafa Hassan og Jafar Alami.
Dato:	19/05/2022
Utstyr:	Sensor hus og 4x sensorer.
Godkjennings krav	Sensor huset burde være vanntett og ledningene skal sitte godt.
Feilmelding:	Ingen.
Konklusjon:	Sensorhuset er vanntett og avlastningen er passende.
Resultat:	Godkjent

I Tabell 36 har vi testet hvor vanntett sensor huset er og avlastningen på ledningene.

Tabell 37: Test av hovedsystem hus.

Test navn:	Hus test 3
Test nr.:	TI02
Test ansvarlig:	Jafar Alami
Test deltakere:	Daniel A. T. Poverud, Gard Johan Tollefsen, Mustafa Hassan og Jafar Alami.
Dato:	20/05/2022
Utstyr:	Hovedsystem huset.
Godkjennings krav	Hovedsystem huset skal være regn tett.
Feilmelding:	Ingen.
Konklusjon:	Vi måtte gi opp på hovedsystem huset
Resultat:	Ikke godkjent

I Tabell 37 å lage en luke til huset ble for innviklet, så ekstern veileder gav oss rådet å bare gi opp.

Tabell 38: Test av hele systemet.

Test navn:	Helsystem test
Test nr.:	T01, T02, T04 og T06
Test ansvarlig:	Jafar Alami
Test deltakere:	Daniel A. T. Poverud, Gard Johan Tollefsen, Mustafa Hassan og Jafar Alami.
Dato:	18/05/2022
Utstyr:	Sensor system, oppladning system, webkamera, gjenkjenning og telle system.
Godkjennings krav	Systemet skal fungere som planlagt.
Feilmelding:	Ingen.
Konklusjon:	Systemet skrudde seg på når sensorene detekterte ett objekt, webkamera tok bilde av fisken vår, modellen kjørte gjenkjenning og telle programmet på bildet, bilde ble lastet opp på Google Drive og telt.
Resultat:	Godkjent

Til sist, i Tabell 38 fikk vi gjort den endelig system testen, og fikk godkjent den i forhold til kravene våre.

9 Konklusjon

Denne delen av rapporten er for å konkludere hvordan vi har jobbet med bachelor prosjektet gjennom denne perioden. Videre skal vi diskutere hvordan vi har tatt for oss den administrative delen av oppgaven, den tekniske delen av oppgaven og egenvurderingen vår.

9.1 Administrativ konklusjon

I starten av prosjektet ble det lagt aller mest vekt på den administrative siden av bacheloren. I denne seksjonen av rapporten skal vi vurdere det administrative vi har gjort i prosjektperioden, som inkluderer planlegging, møter og presentasjoner. Det administrative har forbedret prosjektet og har hjulpet oss fullføre på en effektiv måte.

9.1.1 Planlegging

Det aller første vi gjorde innenfor prosjektet var planlegging av hvordan vi skulle arbeide fremover. Etter mange diskusjoner om hvilken prosjekt modell vi skulle bruke kom vi frem til konklusjonen at vi skulle bruke Unified Process, som er hvordan vi jobbet fremover gjennom prosjektperioden. Denne prosjektmodellen har hjulpet oss på en effektiv måte når vi møtte sperrer i veien. Vi har nå erfaringen av at en god plan er nødvendig for et slikt stort prosjekt.

9.1.2 Møter

Gjennom prosjektperioden har vi hatt ukentlige møter med intern og ekstern veileder, hvor vi har en løs agenda som var generelt å gi en oppsummering av hva gruppen har gjort siden sist og hva vi skal gjøre videre. Samtidig som vi spurte spørsmål om meningen de hadde til prosjektet eller mulige endringer. Gruppen hver Fredag hadde også et lite 15 minutters internt møte, som bare var en kort oppsummering av arbeidet gjort.

Etter hvert møte innen en 24 timers periode sendte vi, med noen unntak, møtereferat og oppfølgingsdokument til intern veileder og bare møtereferat til ekstern veileder som avtalt. Møtereferat er en kort oppsummering av hva som ble snakket om og/eller avtalt på møte. Mens oppfølgingsdokument er hvor vi ukentlig har dokumentert hva vi har gjort den uken, hva vi planlegger å gjøre neste uke, hvordan går prosjektet i forhold til planen og laget timeskriving på hvor lang tid det har tatt å gjøre jobben for alle gruppemedlemmer.

9.1.3 Presentasjoner

I løpet av prosjektet har vi hatt en første presentasjon og en andre presentasjon, etter vi har levert inn oppgaven skal gruppen ha en tredje presentasjon til sist. Første presentasjon var tidlig inni prosjektperioden siden vi hadde den etter tre uker. Der gruppen snakket i 20 minutter om oppgaven vår, planlegging og fremdriftsplanen. Andre presentasjonen var mye av det samme, men vi fylte ut alt vi hadde gjort så langt og hadde en konkret ide av hva vi skulle gjøre fremover. Nå til tredje presentasjon skal gruppen fremføre en 40 minutters presentasjon på 03/06/2022, delt inni to deler som er teknisk og salg. Tilbakemeldingene vi fikk etter presentasjonene var hjelpsomme og lærerikt, og vi skal jobbe hardt for å presentere en god tredje presentasjon.

9.2 Teknisk konklusjon

I denne seksjonen av rapporten skal vi konkludere den tekniske delen av prosjektet. Hvordan vi har fullført jobben med maskinvare og programvaren til systemet vårt.

9.2.1 Maskinvare

I løpet av prosjektperioden har gruppen gjort mange komponentvalg og vurderinger innenfor maskinvaren til systemet. Vi startet med å kjøpe inn en raspberry pi 4 model b, siden en del av oppgaven var å bruke en kompakt pc til å kjøre programvare delen av prosjektet. Det gikk hovedsaklig bra med denne raspberry pi-en, men systemet ble litt tregt. Deretter, gjorde vi et valg på kamera-et systemet skulle bruke. Vi originalt valgte GoPro HERO9 Black kamera-et men gjennom utallige timer arbeid fant vi ut at dette kamera-et var for innviklet å sette opp, dermed måtte vi gå til plan B med et greit webkamera istedenfor. Sensor utvalget var det vanskeligste valget av alt innenfor maskinvare. Innenfor sensorer er det et stort utvalg av forskjellige typer, og vi endte opp med Fotoelektriske sensorer som har også blitt brukt i andre fisketrapper som nevnt tidligere. Til slutt for maskinvaren vår er oppladning systemet, som er et stort batteri 12V batteri med noen små komponenter for å senke spenningen til 5V. Sammenkoblet er disse hoved komponentene maskinvare systemet til prosjektet vårt.

9.2.2 Programvare

Gjennom prosjektperioden har vi brukt mye tid til opplæring og trening av mange forskjellige modeller for å gjenkjenne fisk. Men ikke alle har vært like gode, og generelt er det tregt å gjøre på raspberry pi-en, når vi trente en god modell var den fortsatt treg. Dermed, prøvde vi å bruke Intel Compute Stick 2 som løsning for å øke prosesseringen men etter et stund valgte vi å gå videre i et annet retning. Et automatisk langsiktig system som oppdateres bruken vær gang gjenkjennelse av fisk forgårs. Etter at modellen vår har gjenkjennet fisken og programvaren har tatt tellingen av fisken, så blir video-en sendt til Google Drive for lagring. Gruppen nådde målet når modellen og maskinvaren ble koblet sammen til ett system.

9.2.3 Videre arbeid

Selv om vår prosjektgruppe nå er ferdig med prosjektet betyr det ikke at ingen ting kan bli forbedret med tid. Det er fortsatt ett par ting som kan endres eller forbedres, som kompakt pc-en. Det å bytte ut raspberry pi-en med en Jetson Nano vil forbedre hastigheten til bildeprosessering og gjøre systemet mer effektivt. Skifte kamera-et fra plan B webkamera-et til ett bedre kvalitets kamera. Og til slutt endre gjenoppladning systemet fra å bytte mellom to store batterier til et system drevet av solpanel.

9.3 Refleksjoner

Til sist skal hele gruppen reflektere på prosessen vi nå er imellom og samarbeidet vi har trengt for å klare å lage ett fullstendig produkt.

9.3.1 Daniel Andreas Tokle Poverud

Min hovedrolle innenfor prosjektet var gruppeleder, samtidig som jeg var maskinvare ansvarlig i samarbeid med Gard Johan Tollefsen. Her kom litt av erfaringen min som gruppeleder på andre årets Ingeniørrollen sammen med all opplæring på Elektroingeniør linja til godt bruk.

Som gruppeleder var jeg hovedansvarlig for mesteparten av planleggingen i de første 4-6 ukene av prosjektet samtidig som jeg var hovedansvarlig for mesteparten av diskusjonene når det kom til utvalg av diverse arbeidsmodeller, arbeidsoppgaver og møter med veilederne. Og som maskinvare ansvarlig sammen med Gard Johan Tollefsen valgte vi ut komponentene til systemet og satt komponentene sammen for å lage ett helt system.

Å jobbe i gruppe gjennom dette halvåret har vært spennende, utfordrende og lærerikt. Jeg er fornøyd med framgangen vår fra planlegging, til teori så til et praktisk system. Hvor jeg har testet evnene mine til å jobbe med en praktisk oppgave sånn som dette og gruppen vår sin evne til å samarbeide. Gruppen sin evne til å samarbeide er en av hoved grunnene til at vi klarte dette prosjektet, gjennom det gode og det dårlige. Sammen med prosjekt modellen vår Unified Process som har gitt oss tidsfrister og arbeidsoppgaver å gjøre, for å lage et komplett produkt.

Til sist ønsker jeg først og fremst å takke gruppen min som har jobbet så bra i lengre perioder, veilederne og lærerne som har hjulpet oss når vi trengte det. Takk for hjelpen slikt at vi kunne lage et velstående prosjekt.

9.3.2 Gard Johan Tollefsen

Nå som prosjektet er over ønsker jeg å reflektere på erfaringene og opplevelsene jeg har hatt gjennom dette halvåret. Dette har vært en stor utfordring for meg og hele gruppen vår, hvor vi har jobbet mye sammen i lange perioder.
Som maskinvare og dokument-ansvarlig har jeg hatt hovedansvaret for alt av maskinvare i systemet og hovedansvaret for alt som har med dokumentering å gjøre.
Det har vært utrolig interessant oppgave og prosjekt gruppe, vi har fått knyttet vår teoretiske kunnskaper opp mot praksis. Dette prosjektet har gitt meg innsikt i hvordan det er å jobbe med flere personer om samme oppgave, som jeg er takknemlig for.
Takker til alle sensorer og veiledere som har hatt oss, har vært en spennende oppgave.

9.3.3 Jafar Alami

Uten tvil, dette prosjekt har vært lærerikt, morsomt og ikke minst utfordrende gjennom dette halvåret. Som testing og programvare ansvarlig har jeg hatt hovedansvaret for av test og programvare sammen med Mustafa Hassan. Gjennom dette halvåret, prosjektet har gitt oss masse erfaring om både individuelle og gruppe innsats. I tillegg lærte vi hvordan en virkelig hverdag fungerer for ingeniører i arbeidslivet.

9.3.4 Mustafa Hassan

Etter at prosjektet ble ferdig har jeg fått tid til å tenke over det siste halvåret og jeg har tilegnet meg mange nye kunnskaper om maskin læring, objekt gjenkjenning og raspberry pi generelt.I tilegg til å kunne sette forskjellige teknologier innenfor data og elektro sammen til et komplett fungerende system.
Siden jeg var programvareansvarlig har jeg hatt hovedansvaret for alt som har med programvare og raspberry pi sammen med Jafar Alami.
Prosjektet har gitt meg og resten av gruppa mer erfaring fra virkelige arbeidsmiljøet som vi kan ta med oss videre i livet. Vi har virkelig lært oss å samarbeide det siste halvåret og ser fram til hvor dette fører oss.

9.4 Bidrag

9.4.1 Daniel Andreas Tokle Poverud

Mitt tekniske bidrag var alt av komponent utvalg og sensorer, jeg jobbet i stor del sammen med Gard Johan Tollefsen.

9.4.2 Jafar Alami

Det jeg har bidratt med er en del teknisk, trening av modell, telling , raspberry pi, lagring av data i en skylagring, VNC og testing. Resten finner i hovedrapporten.

9.4.3 Mustafa Hassan

Det jeg har bidratt til gruppe, er en del teknisk ferdigheter.Jeg har jobbet med yolo modellen, sporing/telling, tensorflow object detection, raspberry pi og og lagring av Data fra raspberry pi. Det jeg har skrevet på rapporten er: Telling og sporing, transfer learning,Evaluering av modellen, batch size,lit om verktøy, litt om kravspesifiksjon, en del av test document,en av testplan, skylagring og oppsett av openvino, skylagring, raspberry pi kode og litt av vnc.

9.5 Gard Johan Tollefsen

Bidraget mitt har vært så så si alt som hadde med maskinvaren. Jeg jobbet med også med prosessen sammen med Daniel Andreas Tokle Poverud.

References

[1] J. F. Krarup, "Unified Process Modell for Iterative Utvikling," Available athttps://commons.wikimedia.org/wiki/File:Unified_Process_Model_for_Iterative_Development.svg(22/12/2020).

[2] G. Lewis, "Detaljert Beskrivelse av Unified Process," Available at https://commons.wikimedia.org/wiki/File:UnifiedProcessProjectProfile20060708.png(02/05/2010).

[3] J. Osis and U. Donins, "Chapter 2 - software designing with unified modeling language driven approaches," in *Topological UML Modeling*, ser. Computer Science Reviews and Trends, J. Osis and U. Donins, Eds. Boston: Elsevier, 2017, pp. 53–82. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128054765000022

[4] T. Lin, "Labelimg," *Online: https://github. com/tzutalin/labelImg*, 2015.

[5] P. Skalski, "Make Sense," https://github.com/SkalskiP/make-sense/, 2019.

[6] D. Eatherley, J. Thorley, A. Stephen, I. Simpson, J. MacLean, and A. Youngson, "Trends in atlantic salmon: the role of automatic fish counter data in their recording," 01 2005.

[7] "Setup for raspberry pi," https://www.raspberrypi.com/software/.

[8] "What's the Best Raspberry Pi 4G Modem? Here's a Comparison," Available athttps://jfrog.com/connect/post/whats-the-best-raspberry-pi-4g-modem-heres-a-comparison/.

[9] "Raspberry Pi Hardware," Available athttps://www.raspberrypi.com/documentation/computers/raspberry-pi.html#power-supply.

[10] "SIM7600X-H," Available athttps://www.simcom.com/product/SIM7600X-H.html.

[11] "usb charger faqs," Available athttps://www.cmd-ltd.com/advice-centre/usb-chargers-and-power-modules/usb-and-power-module-product-help/usb-charger-faqs/.

[12] "SIM7600G-H 4G HAT (B)," Available athttps://www.waveshare.com/wiki/SIM7600G-H_4G_HAT_(B).

[13] "Deep learning vs machine learning," https://blog.superannotate.com/content/images/size/w1000/2021/10/deep-learning-vs.-machine-learning.png, Oct 19, 2021.

[14] G. Jocher, A. Stoken, J. Borovec, A. Chaurasia, and L. Changyu, "ultralytics/yolov5," *Github Repository, YOLOv5*, 2020.

[15] A. M. Crescitelli, L. C. Gansel, and H. Zhang, "Norfisk: fish image dataset from norwegian fish farms for species recognition using deep neural networks," *MODELING IDENTIFICATION AND CONTROL*, vol. 42, no. 1, pp. 1–16, 2021.

[16] N. Donges, "What Is Transfer Learning? Exploring the Popular Deep Learning Approach," Available at: https://builtin.com/data-science/transfer-learning(16/06/2019).

[17] T. Gamauf, "Tensorflow Records? What they are and how to use them," Available at: https://medium.com/mostly-ai/tensorflow-records-what-they-are-and-how-to-use-them-c46bc4bbb564(20/03/2018).

[18] D. Oliveira, "Creating TFRecords," Available athttps://keras.io/examples/keras_recipes/creating_tfrecords/(20/03/2018).

[19] Hongkun Yu, Chen Chen, Xianzhi Du, Yeqing Li, Abdullah Rashwan, Le Hou, Pengchong Jin, Fan Yang,Frederick Liu, Jaeoun Kim, and Jing Li, "TensorFlow Model Garden," https://github.com/tensorflow/models, 2020.

[20] R. Prakash, "Guide to Tensorflow Object Detection (Tensorflow 2)," Available athttps://medium.com/swlh/guide-to-tensorflow-object-detection-tensorflow-2-e55ba3cdbc03(5/10/2020).

[21] N. Toure, "Training your Object Detection model on TensorFlow (Part 2)," Available athttps://teyou21.medium.com/training-your-object-detection-model-on-tensorflow-part-2-e9e12714bdf(7/07/2019).

[22] M. Lanham, " Learn ARCore - Fundamentals of Google ARCore: Learn to build augmented reality apps for Android, Unity, and the web with Google ARCore, 1.0nd ed.:Packt Publishing,2018."

[23] H. Singh, "Variants of Gradient Descent Algorithm," Available athttps://www.analyticsvidhya.com/blog/2021/03/variants-of-gradient-descent-algorithm/#:

~:text=In%20the%20case%20of%20Stochastic,parameters%20based%20on%20every%20subset(15/03/2021).

[24] J. Hui, "mAP (mean Average Precision) for Object Detection," Available athttps://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173(7/03/2018).

[25] E. Hofesmann, "IoU a better detection evaluation metric," Available athttps://towardsdatascience.com/iou-a-better-detection-evaluation-metric-45a511185be1(25/08/2020).

[26] S. Yohanandan, "mAP (mean Average Precision) might confuse you!" Available at: https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2(09/07/2020).

[27] J. JORDAN, "Evaluating a machine learning model," Available athttps://www.jeremyjordan.me/evaluating-a-machine-learning-model/#:~:text=The%20three%20main%20metrics%20used,the%20number%20of%20total%20predictions(21/07/2017).

[28] "Model Fit: Underfitting vs. Overfitting," Available athttps://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html.

[29] "What does Underfitting Mean?" Available athttps://www.datarobot.com/wiki/underfitting/.

[30] A. Oppermann, "Underfitting and Overfitting in Deep Learning," Available athttps://medium.com/mlearning-ai/underfitting-and-overfitting-in-deep-learning-687b1b7eb738(18/07/2021).

[31] Vidushi Meel, "Object Tracking in Computer Vision," Available athttps://viso.ai/deep-learning/object-tracking/.

[32] "Tracking by detections," Available athttps://opencv.org/wp-content/uploads/2020/10/Tracking-by-detections-tracks.jpg(10/2020).

[33] S. R. Maiya, "DeepSORT: Deep Learning to Track Custom Objects in a Video," Available athttps://nanonets.com/blog/object-tracking-deepsort/, 2019.

[34] “Multiple People Tracking,” Available athttps://opencv.org/wp-content/uploads/2020/10/Multiple-People-Tracking.jpg(10/2020).

[35] M. Broström, “Real-time multi-camera multi-object tracker using yolov5 and deepsort with osnet,” https://github.com/mikel-brostrom/Yolov5_DeepSort_OSNet, 2022.

Appendices

A Openvino/NCS2

Openvino

Problemet vi fikk når vi kjørte modellen på raspberry pi-en. Var at den behandlet videoen ekstremt tregt. Vi gjorde en del forskning på hvordan man kan løse dette. Vi fant noen løsninger på det, den mest enklest var å kjøpe inn en Coral tpu, men de var utsolgt. Andre løsningen var å bruke et produkt som heter Intel Nueral Compute Stick 2, heldigvis fikk vi tak av intel sticken.

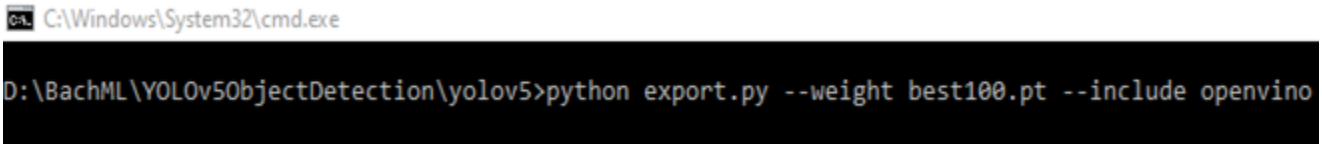
Intel NCS2 er en utviklings plattform, som brukes til å raskt bringe datasyn og AI til IoT og edge-enheter. AI akseleratorer som intel stick 2 VPU(Vision Processing Unit) er nyttig for å akselerere dataintensiv dyplærings inference på edge enheter på en kosteffektiv måte. Måten intel sticken hjelper en edge enheter som vår raspberry pi, er å hjelpe CPU til raspberry pi-en ved å ta over den matematiske byrden som trengs for å kjøre dyplæringsmodeller. Den klarer å kjører dyplærings modeller på lavt kostnad, lavt strømforbruk og høyere hastighet. Planen vi hadde i starten var å kunne kjøre modellen i ”real time” og for å få til dette måtte vi klare å koble sammen systemet vårt med intel sticken **??**.
For å jobbe med intel sticken er det viktig å ha en x86-64 host pc med windows 10 eller Ubuntu (16.04 eller 18.04) for Intel® Distribution of OpenVINO™ toolkit. Intel® Neural Compute Stick 2 (Intel® NCS 2) og vi må installere Intel® Distribution of OpenVINO™ toolkit. For å kunne kjøre på raspberry pi må installere en annen version av openvino som er ment for debian. Problemet vi fikk først var at dokumentasjonen og alle guides vi så på var fra seinest 2021. Problemet med det var at vi fikk en stor oppgradering i 2022 med en OpenVINO API(API 2.0), der vi fikk mye endring på installasjon og oppsett. De hadde en transition guide for å hjelpe migrasjon fra det gamle. Noe som var forvirende, fordi de hadde

forsatt noen dokumentasjon som hørte til før 2022. Det var et stort problem når vi prøvde å kjøre demo modeller og sampler, dokumentasjon på det var for openvino 2021.
Etterhvert valgte vi å prøve å få det til med openVINO 2021.4, det kommer med mye endring. Gjennom hele prosessen hadde jeg lastet ned fire versjoner av python. Tilslutt klarte vi å få det til på openvino 2021 på pc-en. Det vil si at vi lastet vned alle requirments, og kunne kjøre setvars.bat som er miljø oppsett for å kunne kjøre openvino. Nå er det å kjøre vår yolo modell sammen med intel sticken.
For å kunne kjøres på openvino, vår custom modell må bli konvertert til openvino format som er IR(intermediate representation). For å gjøre dette må vi bruke openvino sin mo script for å konvertere og optimalisere modellen vår. For å gjøre dette med yolov5 er det en del arbeid man må gjøres først. Vi måtte prøve flere måter å konvertere og at den fikk kjørt på openvino. Vi klarte å konvertere på to forskjellige måter, men kun en fungere som det skal. Den enkelste måtte å konvertere til IR er gjennom Yolov5 sin export script, etter vi lastet ned alle requirements kan vi bruke den som vises i Figur91.

```
Usage:
    $ python path/to/export.py --weights yolov5s.pt --include torchscript onnx
openvino engine coreml tflite ...
```

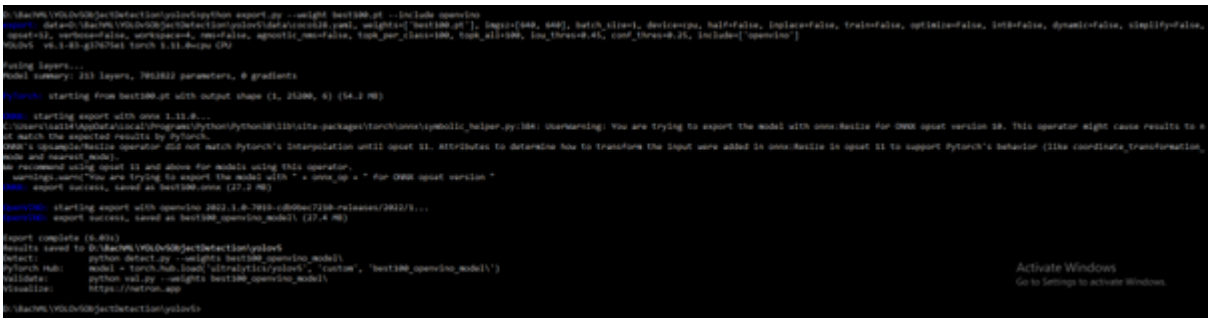
Figur 91: Bruk av Yolov5 export script

I vår tilfelle har vi trent en modell og har gitt den navnet best100, her kan du bruke ditt eget modell, men for oss det blir som vises i Figur 92:



Figur 92: Hvordan vi exportere til openvino

Outputen vi får fra dette er som Figur 93:



Figur 93: Output av openvino export

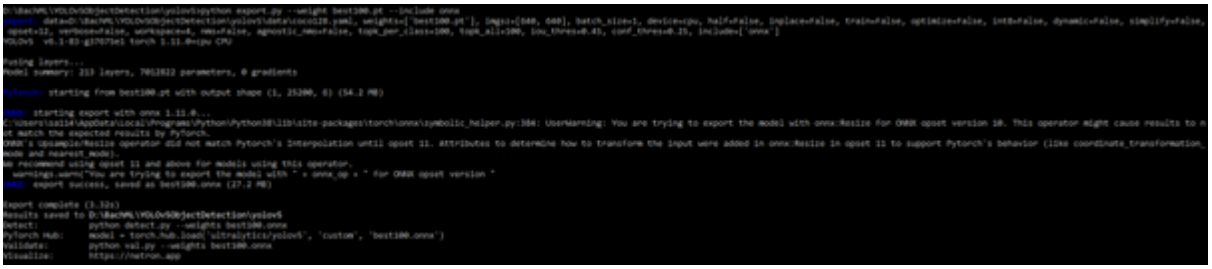
Vi fikk konvertert modellen til IR format, men når vi kjørte den gjennom openvino med scriptet som finnes på https://github.com/violet17/yolov5_demo, fikk vi ikke det til. Da prøvde vi den andre metode som fungerte, men det krevde en del for å få det til i forhold til yolov5 sin metode.

Andre metoden starte på samme sted, og det er export.py på yolov5. Denne gangen, skal vi ikke konvertere til openvino, men til onnx format. Før vi eksportere, må vi endre på opset versjon i export.py filen, siden openvino 2021 støtter ikke ONNX opset versjon 12 så vi må endre det til opset 10 slik i Figur 94:

```python
def export_onnx(model, im, file, opset, train, dynamic, simplify, prefix=colorstr('ONNX:')):
    # YOLOv5 ONNX export
    try:
        check_requirements(('onnx',))
        import onnx

        LOGGER.info(f'\n{prefix} starting export with onnx {onnx.__version__}...')
        f = file.with_suffix('.onnx')

        torch.onnx.export(
            model,
            im,
            f,
            verbose=False,
            opset_version=10,
            training=torch.onnx.TrainingMode.TRAINING if train else torch.onnx.TrainingMode.E\
            do_constant_folding=not train,
            input_names=['images'],
            output_names=['output'],
            dynamic_axes={
                'images': {
                    0: 'batch',
                    2: 'height',
                    3: 'width'},  # shape(1,3,640,640)
                'output': {
                    0: 'batch',
                    1: 'anchors'}  # shape(1,25200,85)
            } if dynamic else None)
```

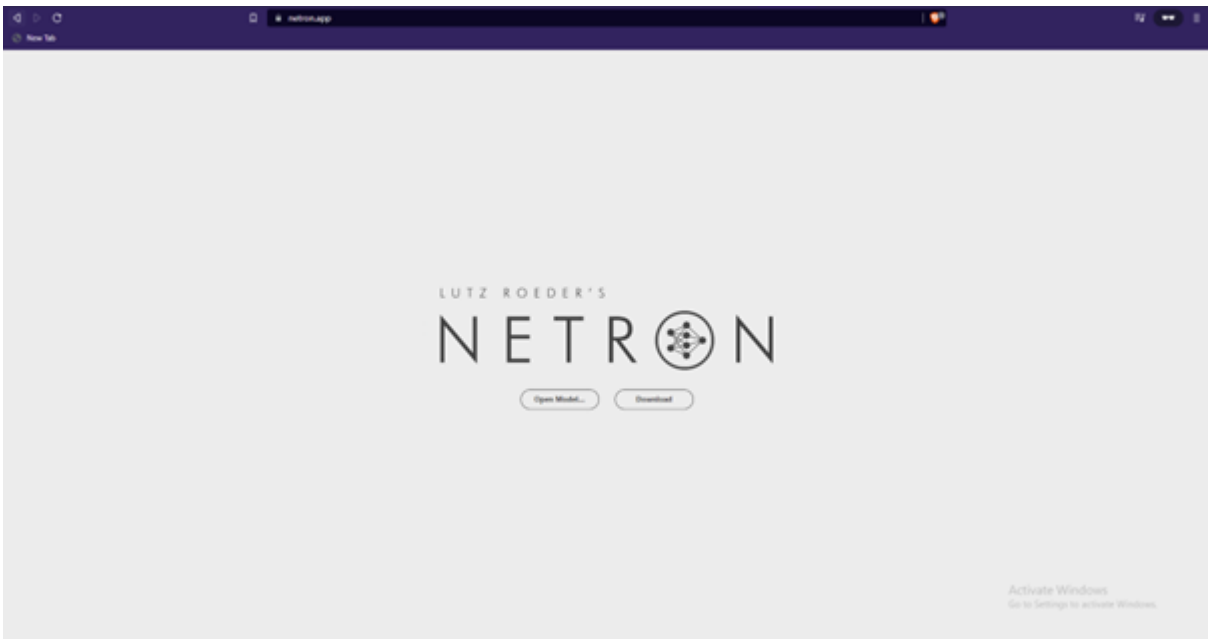
Figur 94: Endring til opset 10

Da kan vi konvertere det til onnx format, på samme måte som før bare endre openvino til onnx som i Figur 95.



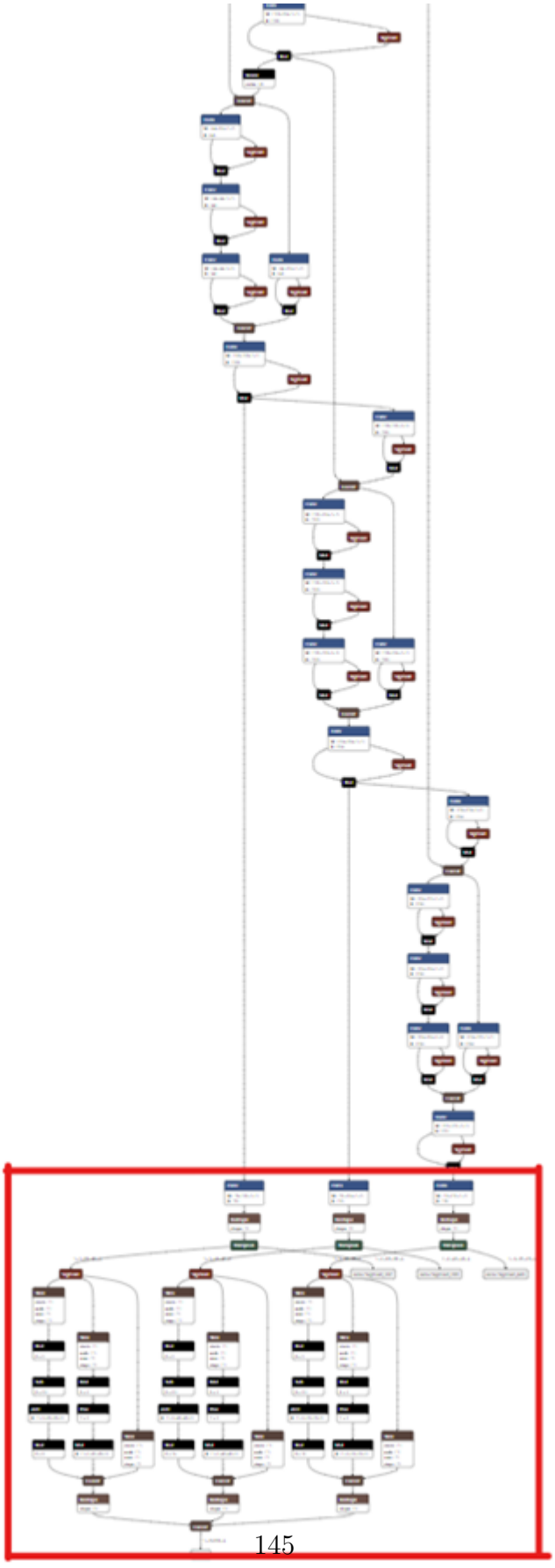
Figur 95: Outputen vi får ved konvertering til ONNX

Etter konvertering skal vi få en best100.onnx fil i file explorer. Inne på terminalen kan vi se mot slutten en link til netron.app. Så gå inn på nettleseren og skrive in netron.app, og da får vi en nettside som vises i Figur 96.

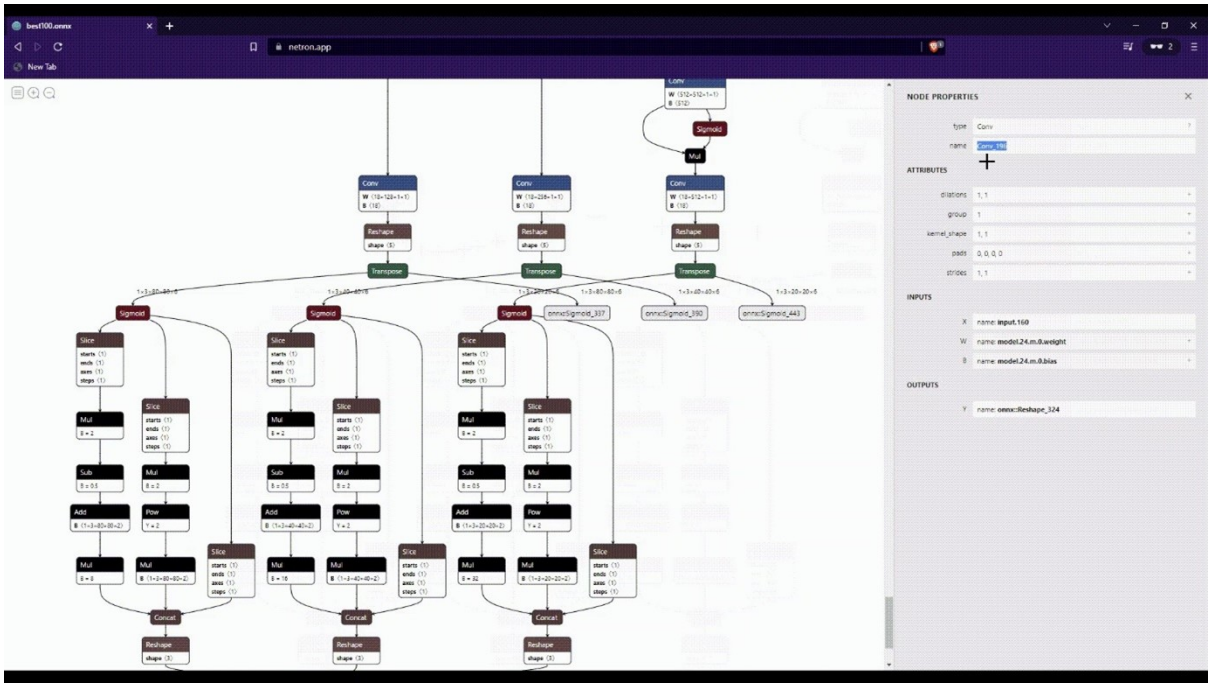


Figur 96: Netron nettsiden

Trykk på Open Model..., og finner den nye best100.onnx fil. Du skal få en veldig stor graf, men det som er viktig å skrive ned er de siste tre conv som ligger i Figur 97 og 98.



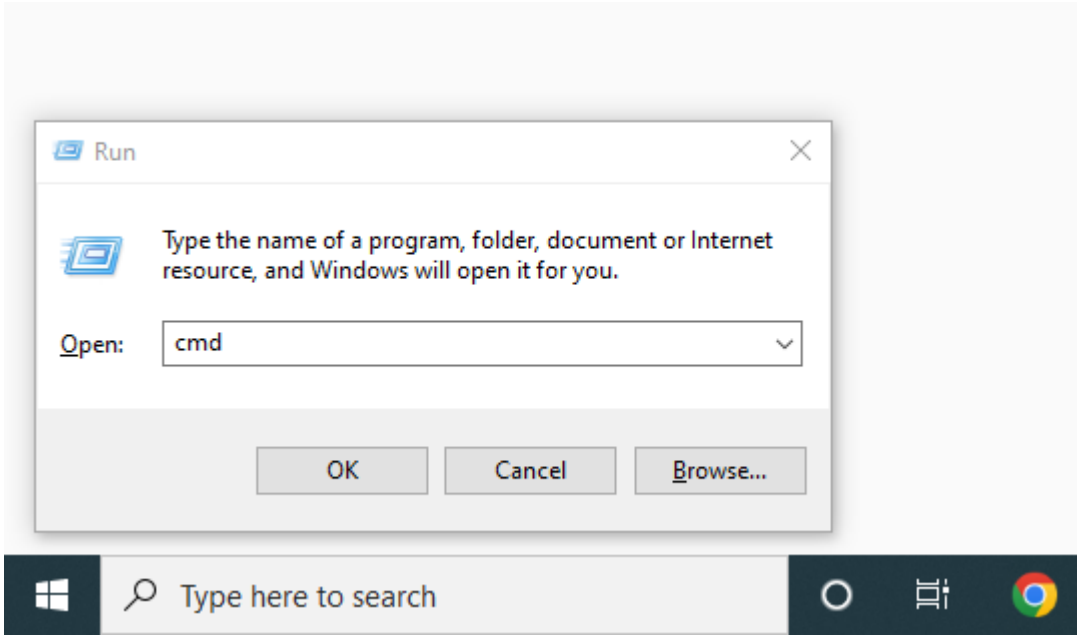
Figur 97: graf for all lag



Figur 98: Siste tre Conv som skal skrives ned

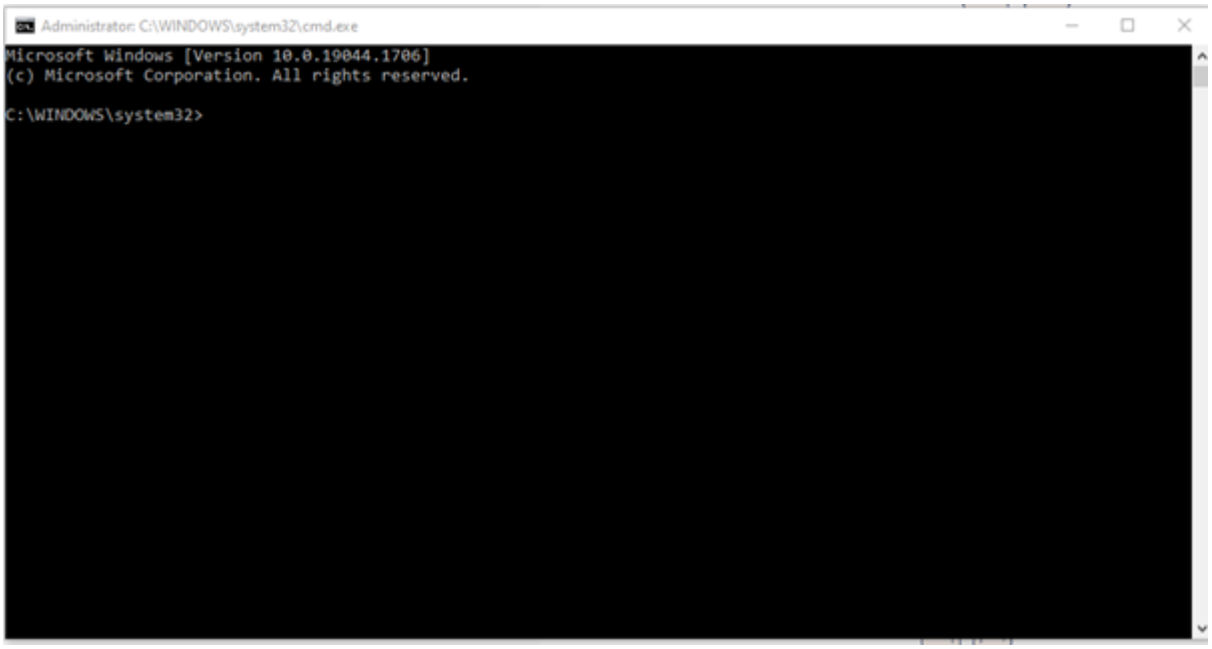
I vårt tilfelle må vi huske på Conv_196, Conv_230 og Conv_264 for senere.

Nå må vi gå inn på openvino 2021.4 og initialisere setvars.bat som openvino trenger. Et problem som kan skje er at vi får ikke tilgang, for å unngå det må vi åpne terminalen som administrator. Det finnes flere måter å åpne terminal som administrator, enklest for å dokumentere er å trykke på Windows + R da skal få det som ligger i Figur 99.



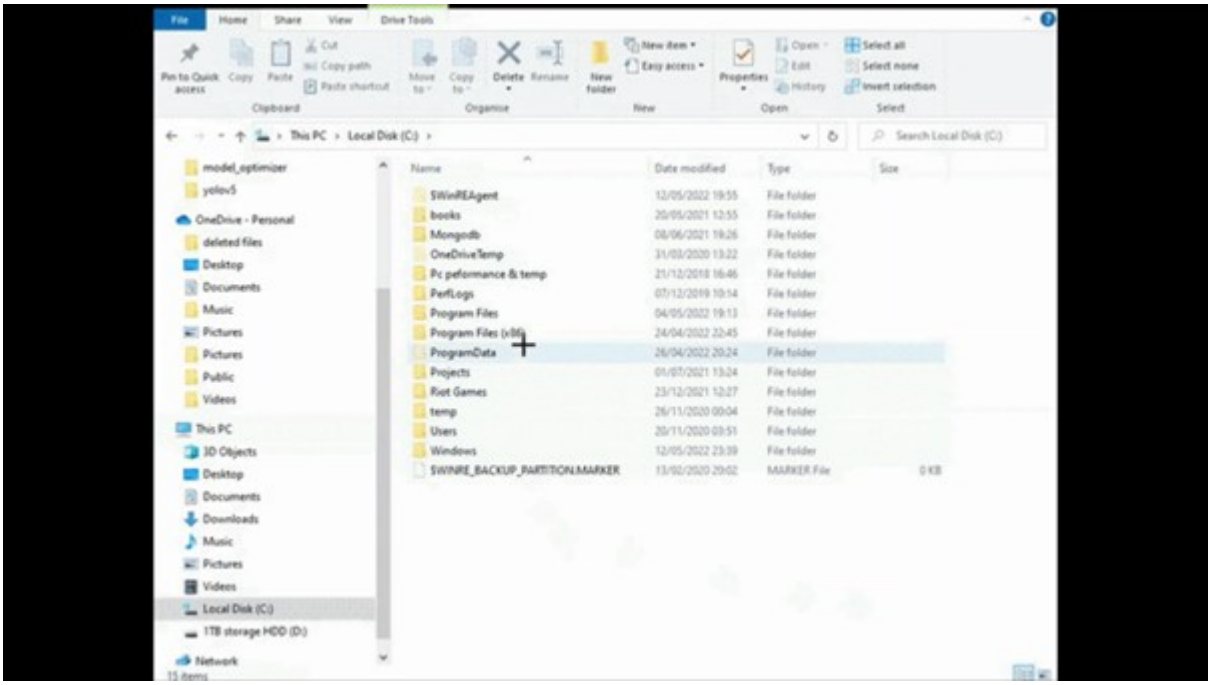
Figur 99: Windows + R

Her skal du skrive cmd og tykket på ok for å åpne terminal, men for at vi skal får administratortilgang. Skal du trykke på Control + Shift + Enter før du trykke på ok, da skal du få en prompt fra windows der du skal trykke på yes. Vi får en UI slik i Figur 100:



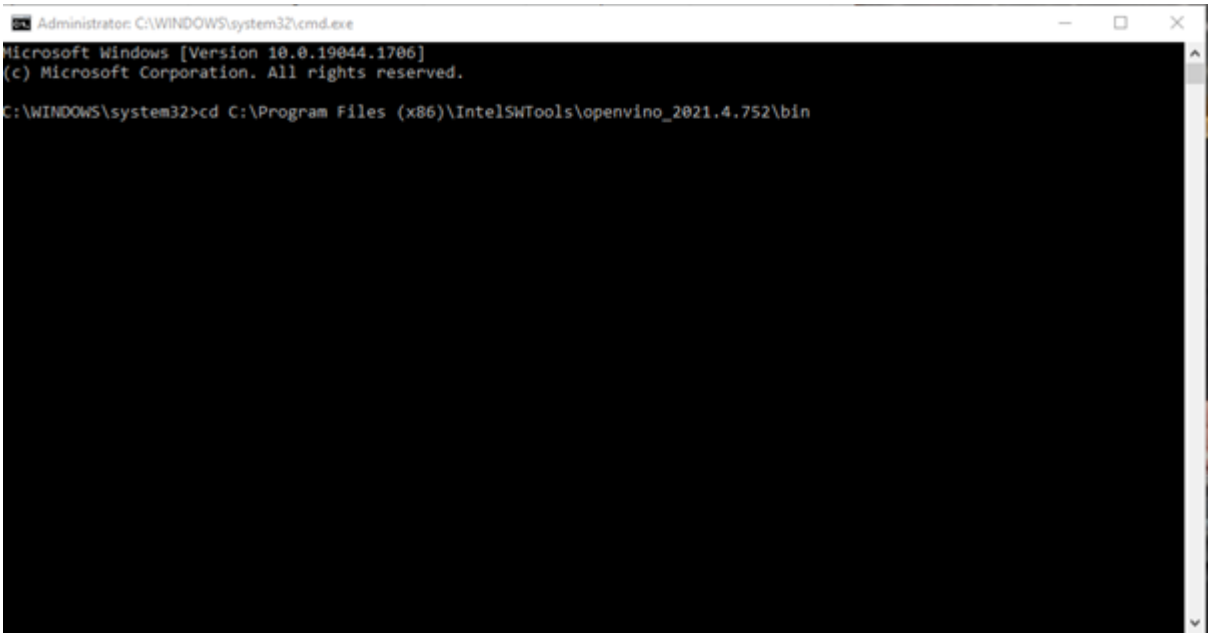
Figur 100: Terminal med admin

Nå skal du inn på file explorer, og gå til den filkatalog som du har lastet ned openvino. Vi har lastet den ned til default plass og vises nedenfor i Figur 101:



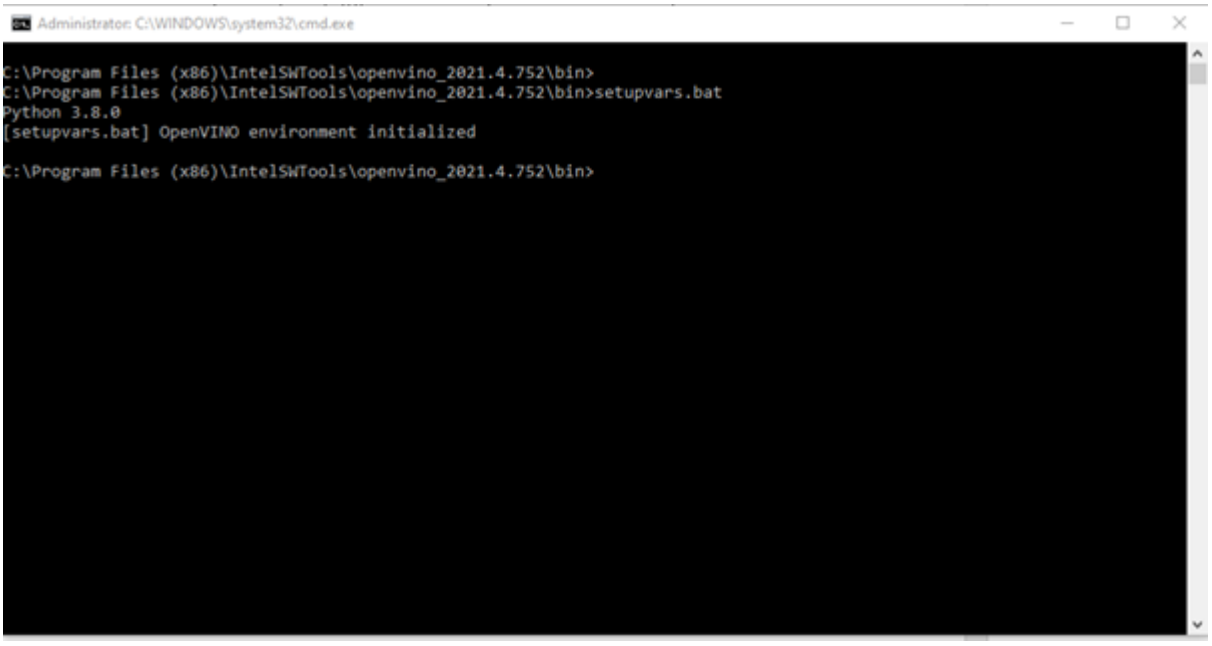
Figur 101: Bildet på hvilket fil vi skal velge

En video som ligger på Openvino/nsc2 oppsett.docx vises på hvor man kopieres adressen fra, men adressen som må kopieres er `C:\ProgramFiles(x86)\Intel\openvino_2021.4.752\bin` Nå skal du kopier adressen og gå tilbake til terminalen, skriv cd og lim inn adressen som i Figur 102:



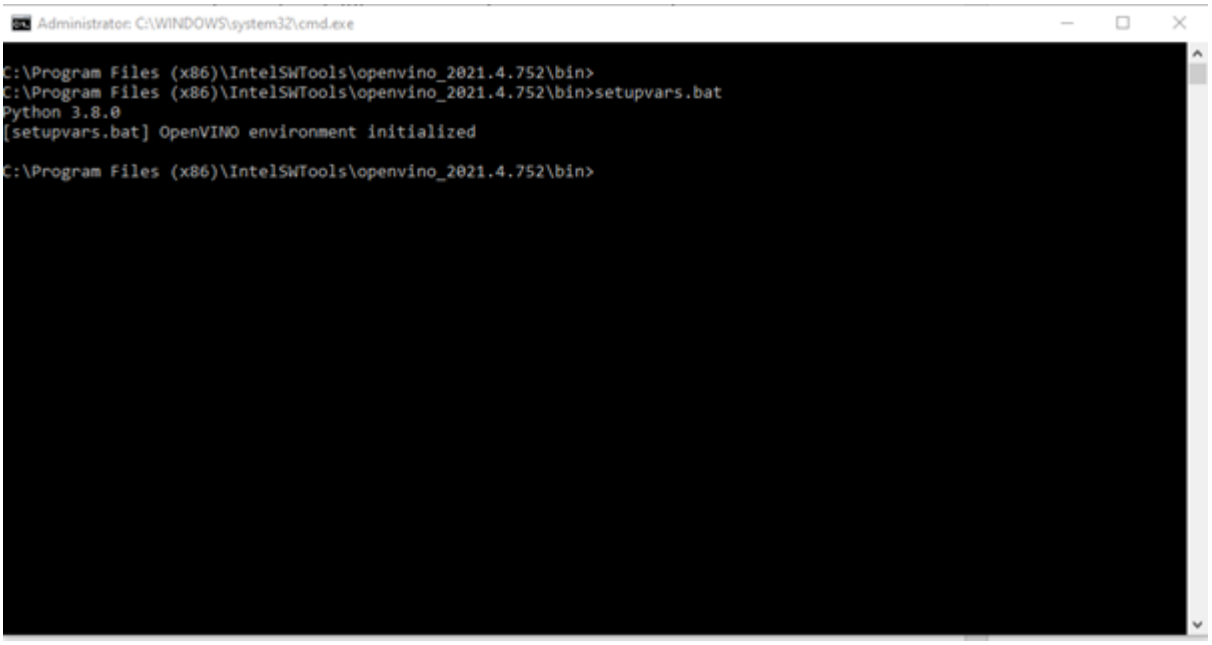
Figur 102: For å komme inntil setupvars.bat folder

Trykk på enter, og skriv setupvars.bat. Og vi skal få den outputen nedenfor, det vil si at miljøet er satt opp for openvino, outputen skal ses som Figur 103:



Figur 103: Vi kjørte setupvars.bat og miljøet er satt opp

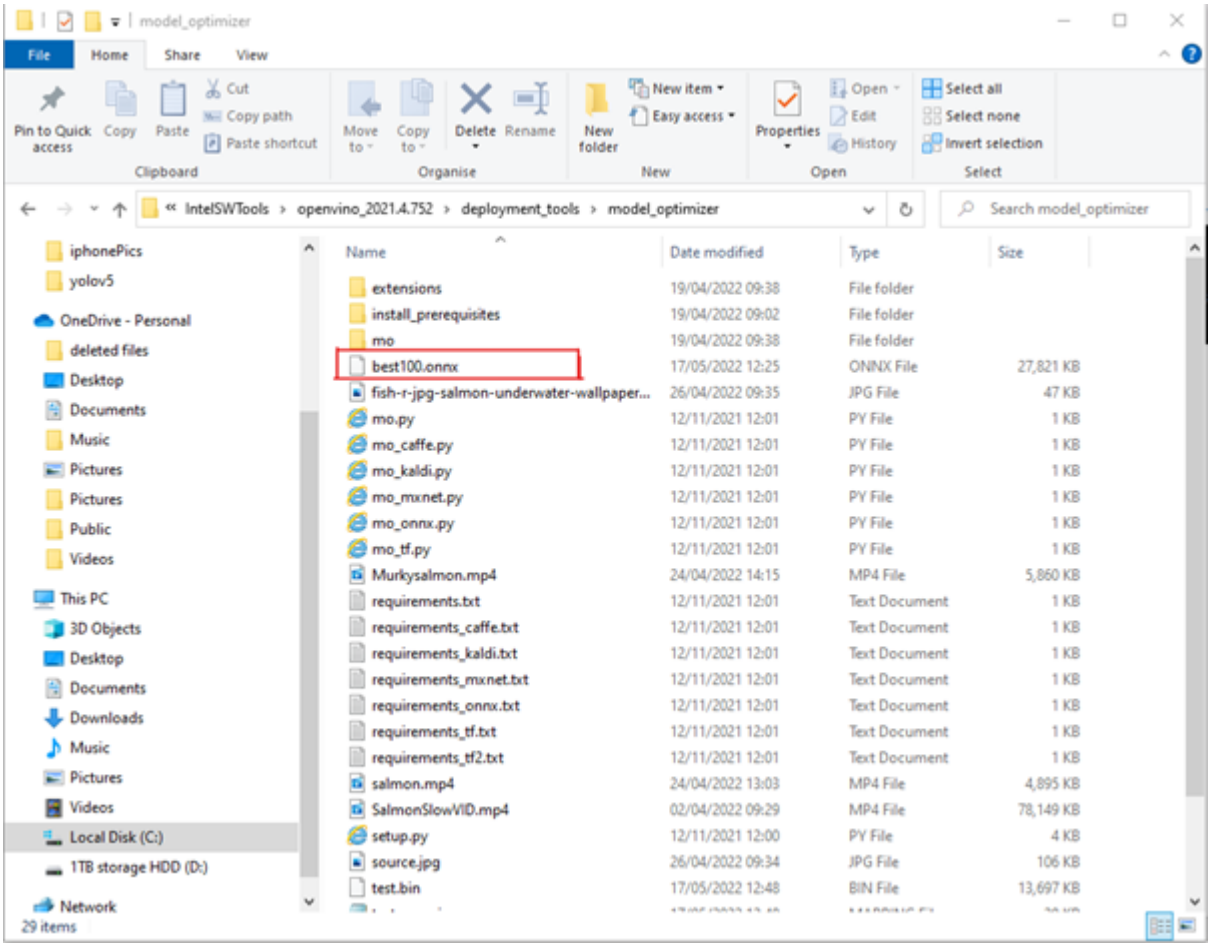
Nå skal vi tilbake til filplasseringen, og gå inn til folderen::
`C:\ProgramFiles(x86)\Intel\openvino_2021.4.752\deployment_tools\model_optimizer`
Kopier adressen og gå tilbake til terminal 104:



Figur 104: Lim adressen i terminalen.

Skriv "cd" og lim inn adressen.

Nå skal vi begynne å flytte vår best100.onnx fil fra yolov5 til `C:\ProgramFiles(x86)\IntelSWTools\openvino_2021.4.752\deployment_tools\model_optimizer` slik i Figur 105.



Figur 105: Markering av Best100.onnx fil.

For oss til å konvertere den modell til openvino IR, må vi bruke kommandoen 106:

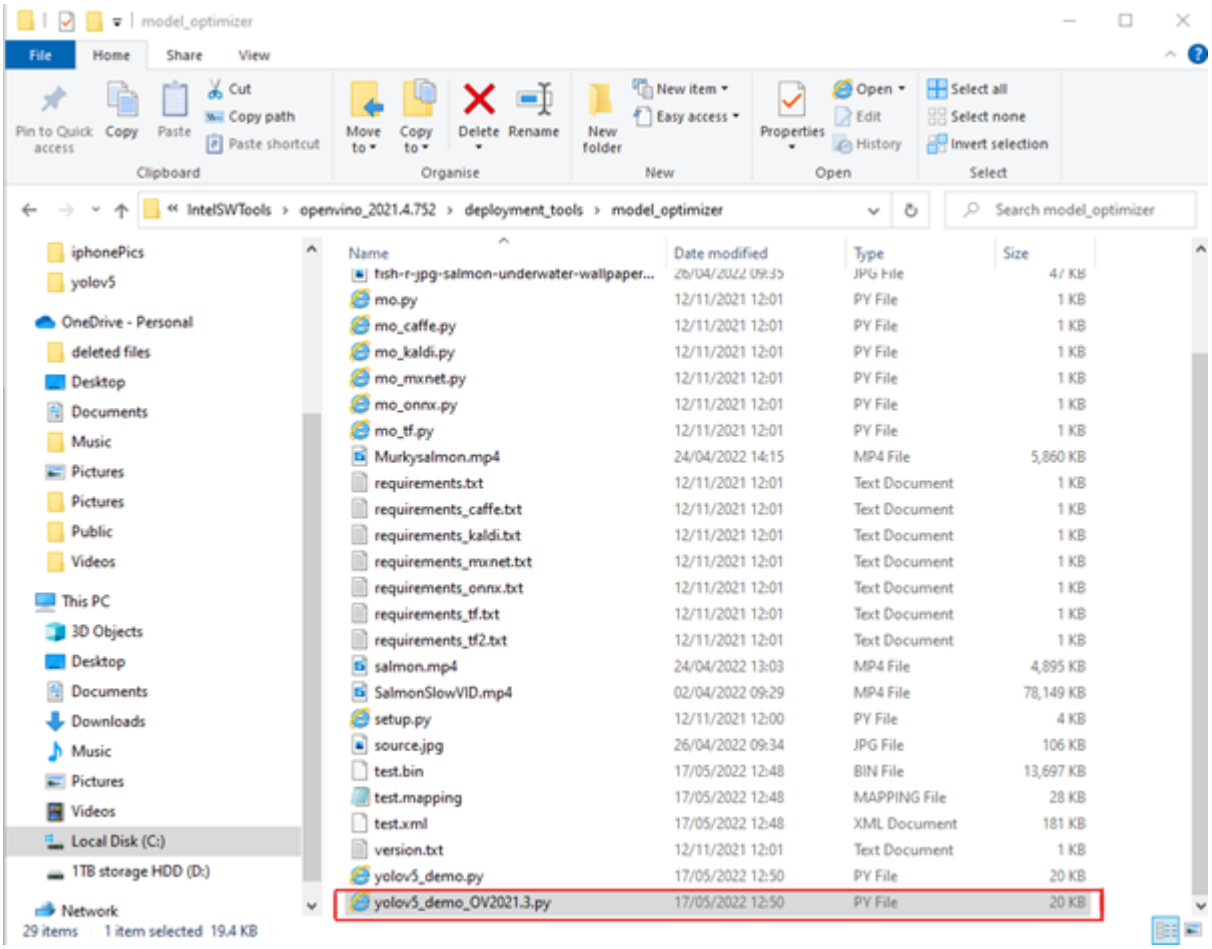
```
C:\Program Files (x86)\IntelSWTools\openvino_2021.4.752\deployment_tools\model_optimizer>python mo_onnx.py --input_model
=best100.onnx --model_name=test -s 255 --reverse_input_channels --output Conv_196,Conv_230,Conv_264 --data_type=FP16
```

Figur 106: Kommandoen for Openvino konvertering.

python mo_onnx.py –input_model=best100.onnx –model_name=test -s 255 –reverse_input_channels –output Conv_196,Conv_230,Conv_264 –data_type=FP16
Du skal endre på litt, –input_model = "Din modell" og –output Conv_(Første Conv) , Conv_(Andre Conv) , Conv_(Tredje Conv). Tallet som skal være inn i parenteser er de vi fikk fra netron.app. Med den kommandoen skal vi få tre nye filer:

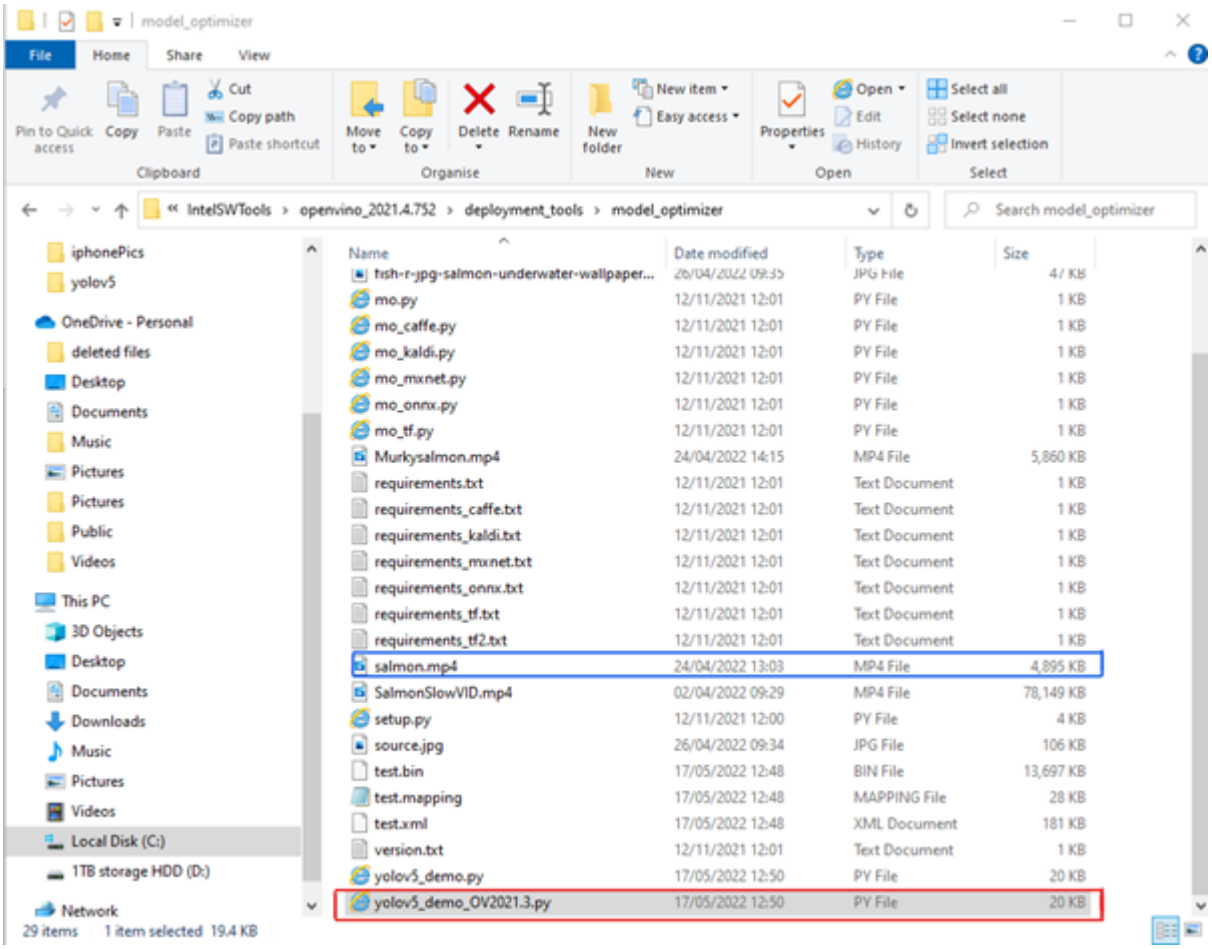
- test.xml
- test.bin
- test.mapping

Hvis vi mangler noen moduler, er det bare å pip install modulen. For eksempel, hvis vi mangler onnxruntime skal vi skrive, pip install onnxruntime.
Nå har vi konvertert modellen, neste er å teste om de funker som det skal. For å kjøre modellen må vi bruke en custom script som ble skapt av violet17 på github(https://github.com/violet17/yolov5_
Du skal laste ned yolov5_demo_OV2021.3.py fra google driven til model_optimizer folder som i Figur 107.



Figur 107: Vising av yolov5_demo_OV2021.3.py i rikitig folder.

Last ned en test video inn på samme folder som i Figur **??**.

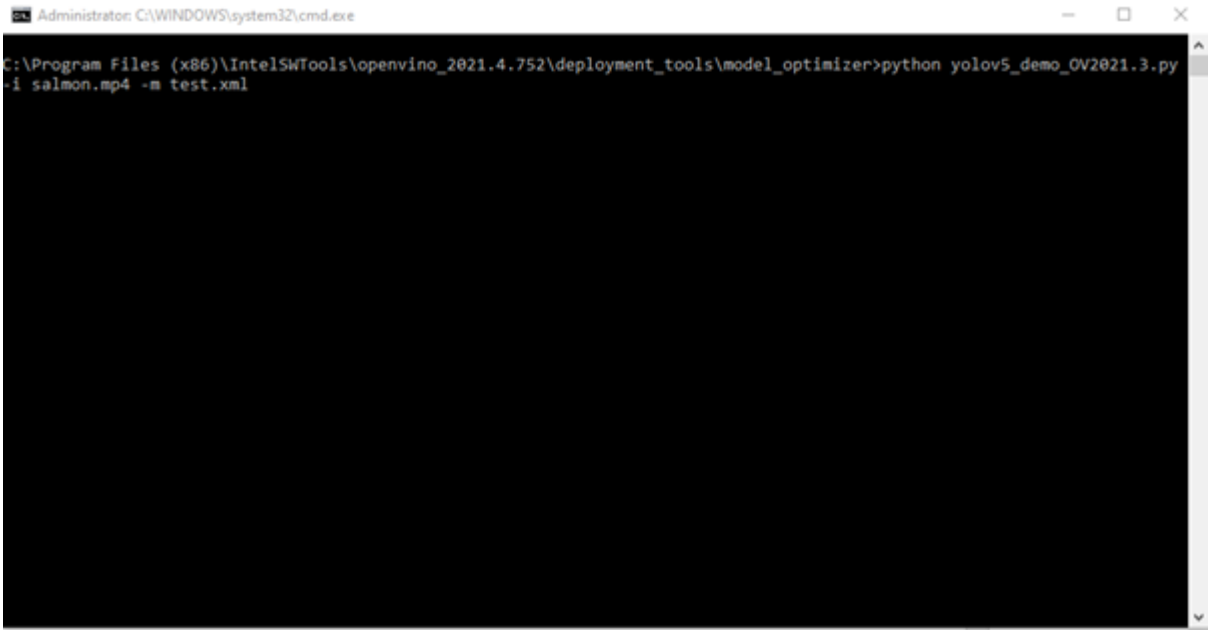


Figur 108: Video i samme folder som yolov5_demo_OV2021.3.py.

Nå skal vi inn til terminalen, og skrive: python yolov5_demo_OV2021.3.py -i salmon.mp4 -m test.xml.

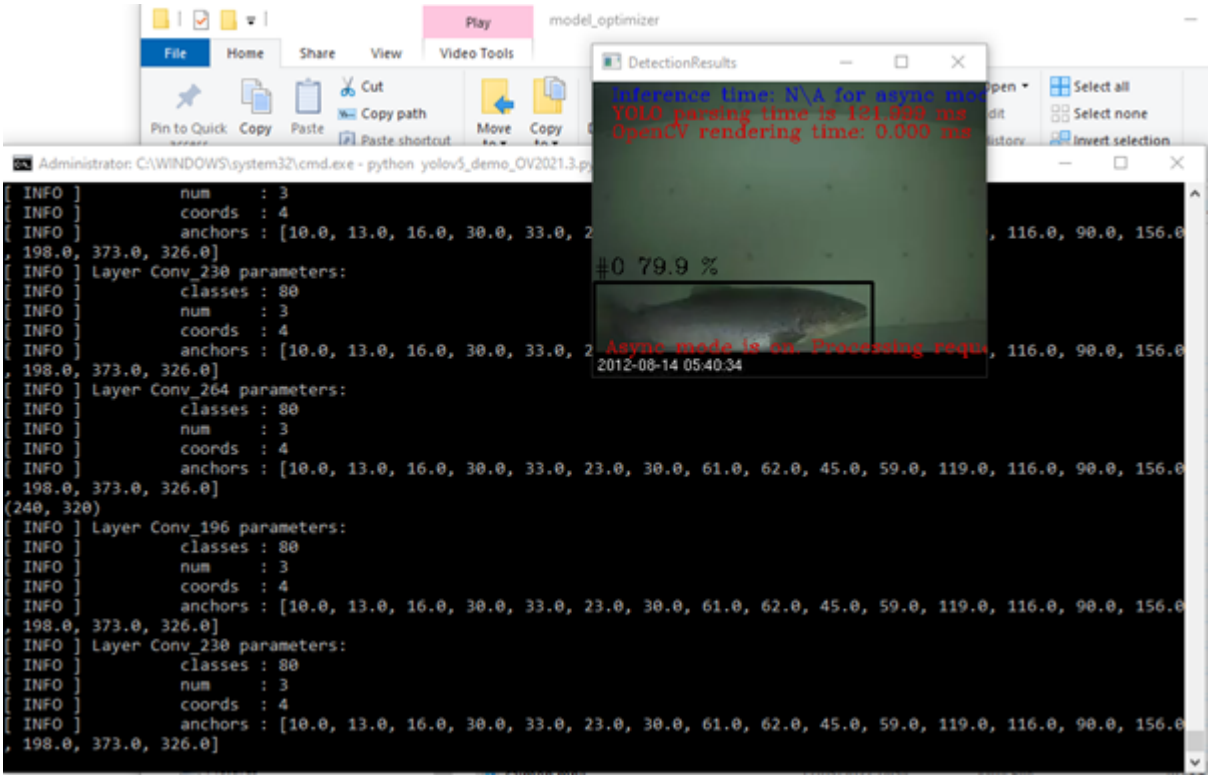
- -I = source input(cam for webcam, video og bilder)
- -m = Model (vår test.xml fil)
- -d = devices(Myriad)

Devices er valgfri,det vil si at hvis vi la det står, blir vårt modell kjørt på CPU, for å å kjøre på intel sticken må spesifisere -d Myriad. Se på Figur 109.



Figur 109: Kjøre modellen med den computer stick og en test video .

Da får vi outputen i Figur 110:



Figur 110: Output video fra Computer stick test.

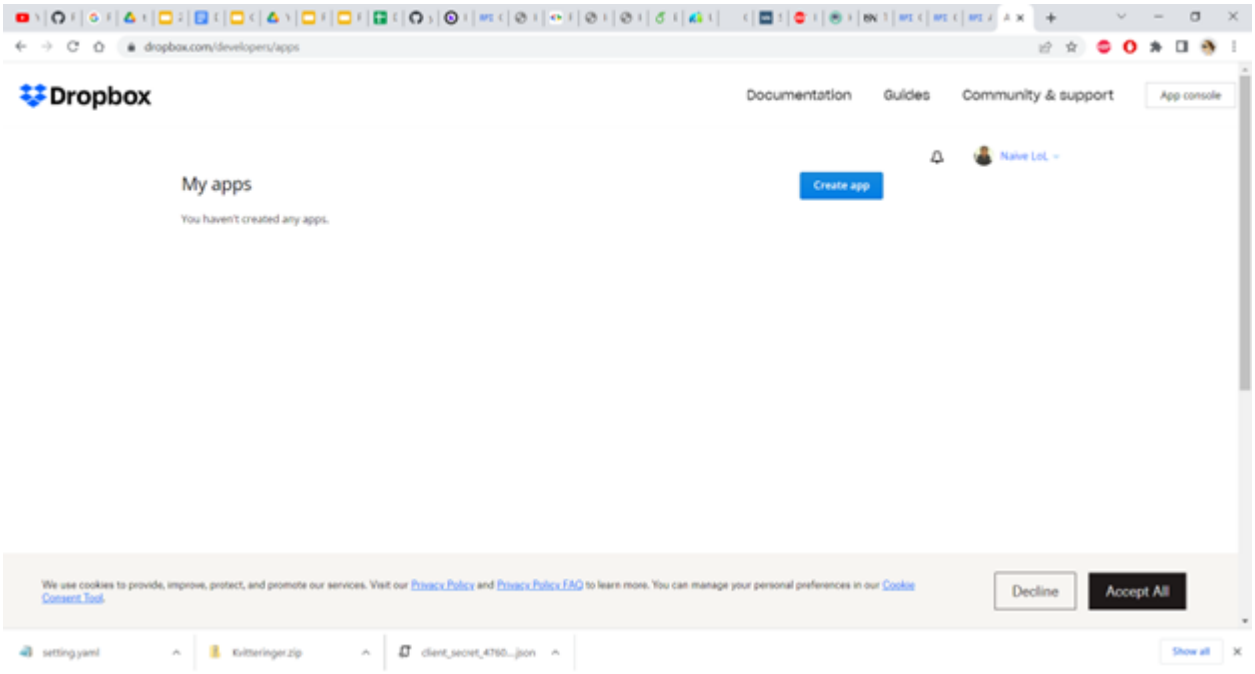
Nå har vi satt opp computer sticken med openvino på windows. Det her er alt vi klarte å få til, vi hadde ikke god tid mot slutten av prosjektet og vi visste ikke hvordan vi hadde implementert den delen av prosjektet med andre delen (Sporing og telling). Så vi valgte å gå videre med å implementere alt på raspberry pi (sensor/camera/modellen/tracking/counting) og få til sending av vår data til en UI.

B Oppsett av DropBox

For å kunne sette opp dropbox, måtte vi allerede lastet ned git. For å laste ned git skal du bare skrive ” sudo apt install git” inn på terminalen. Dropbox er den først skylagring som vi har jobbet med, det var greit til å sette opp og få last opp data fra raspberry pi til dropbox. Problemet vi fikk handlet om tilgangstokens som ble nevnt i hovedrapporten. Det finnes en løsning på det problemet, men vi sleit med å få det til. Vi valgte å gå videre med gdrive, fordi de tok over tilgangstoken håndtering. Nå jeg skal gå gjennom hvordan man setter opp dropbox med en kort-levd tilgagngstoken, det vil si at vi må manuelt

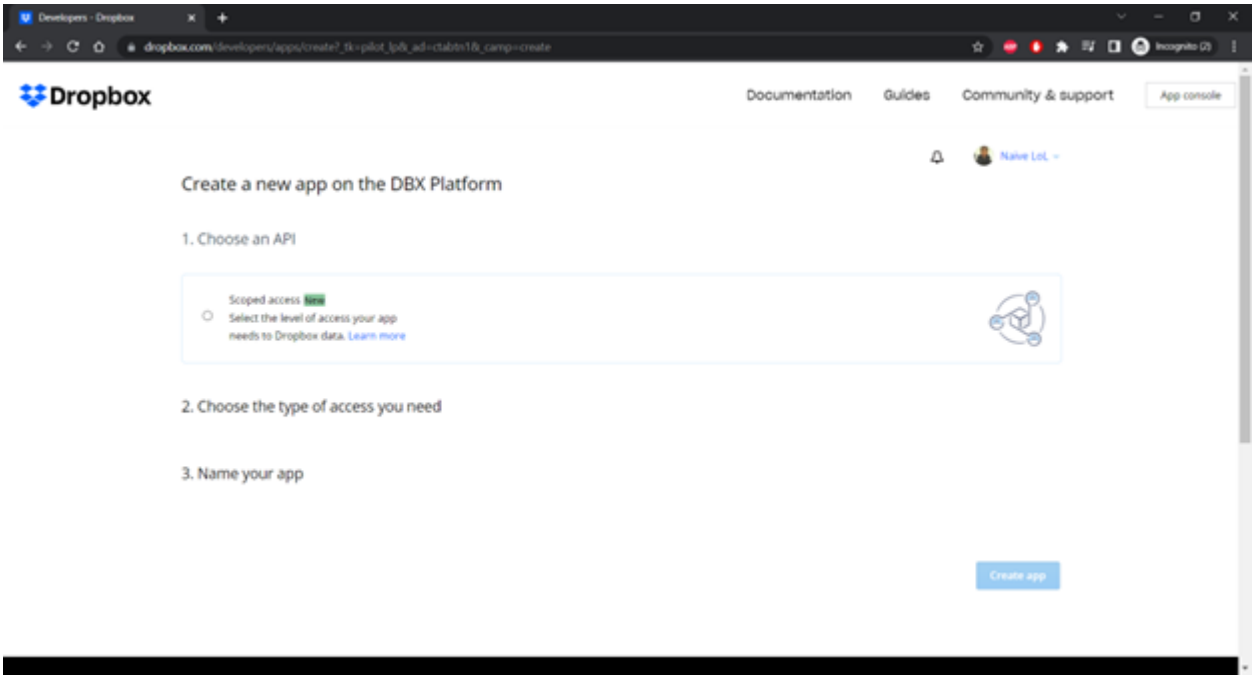
generere en tilgangstoker hver 4 timer for at raspberry pi kan laste opp data til dropbox. Det er noe vi kan sikkert løse etterhvert, men for nå har vi kommet så langt. Vi har følgt malen til https://pimylifeup.com/raspberry-pi-dropbox/, men går litt mer i detaljer for å nå opplasting.

Henting av Dropbox api Det først man skal gjøre er å lage en konto hos dropbox, det er noe som dropbox har gjort enkelt for oss til å lage. Etter vi har lagt en konto hos dropbox, skal vi trykke på linken her: `https://www.dropbox.com/developers/apps`. Vi skal da kommer til en side som vises i Figur 111 :



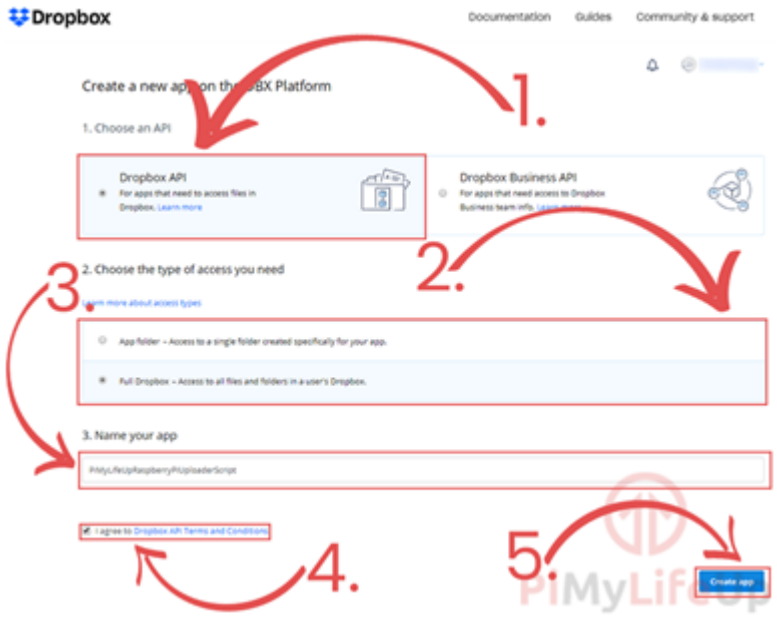
Figur 111: Nettsiden til Dropbox.

Trykk på der det står ”create app” da side ser som i Figur 112:



Figur 112: Lage en nytt App.

Følg instruksjonen som blir vist i Figur 113:



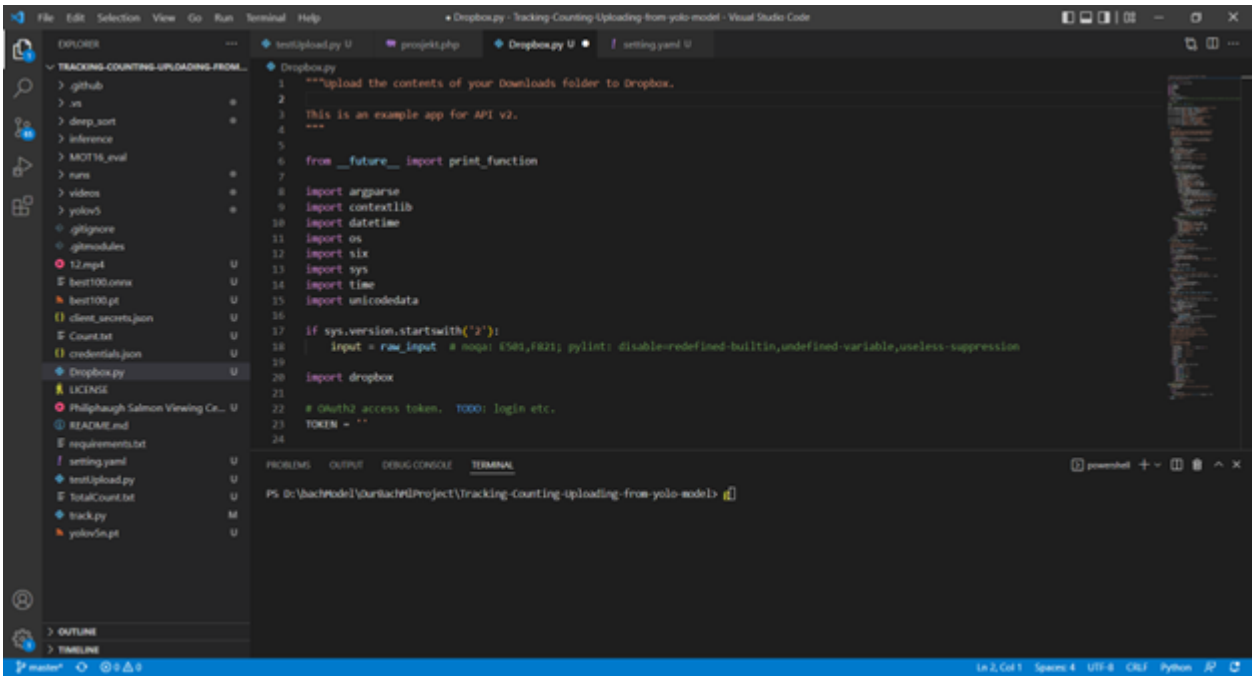
Figur 113: Instruksjon for å lage en Dropbox app.

Når du har lagd en app skal du gjør som følgende:

Gå inn til Oppsett av DropBox.docx for å se på gif.

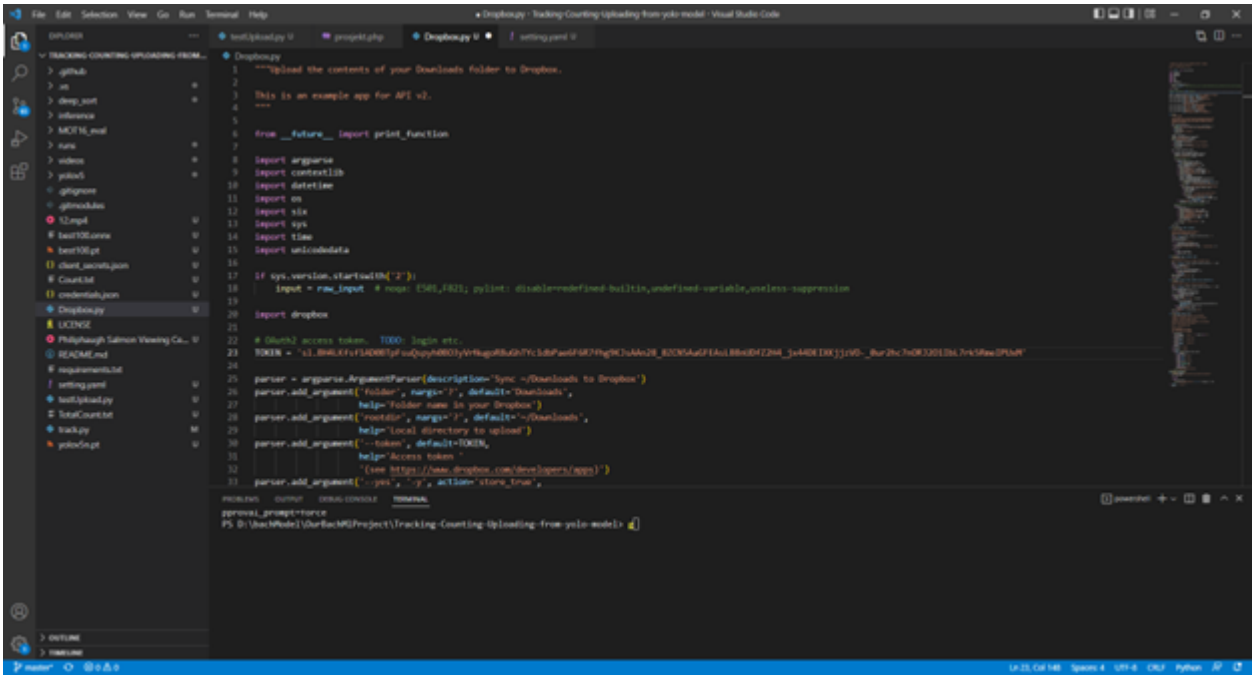
Da går vi over til dropbox sdk repository for å bruke scripts til å laste opp filer (https://github.com/dropbox/dropbox-sdk-python). Først skal laste ned dropbox til pi-en, ved å skrive pip install dropbox inn på terminalen.

Da skal vi kopiere scripten som ligger inn på: https://github.com/dropbox/dropbox-sdk-python/blob/main/example/updown.py Deretter skal vi lage en ny fil innen vscode og limmet koden der som vises i Figur 114



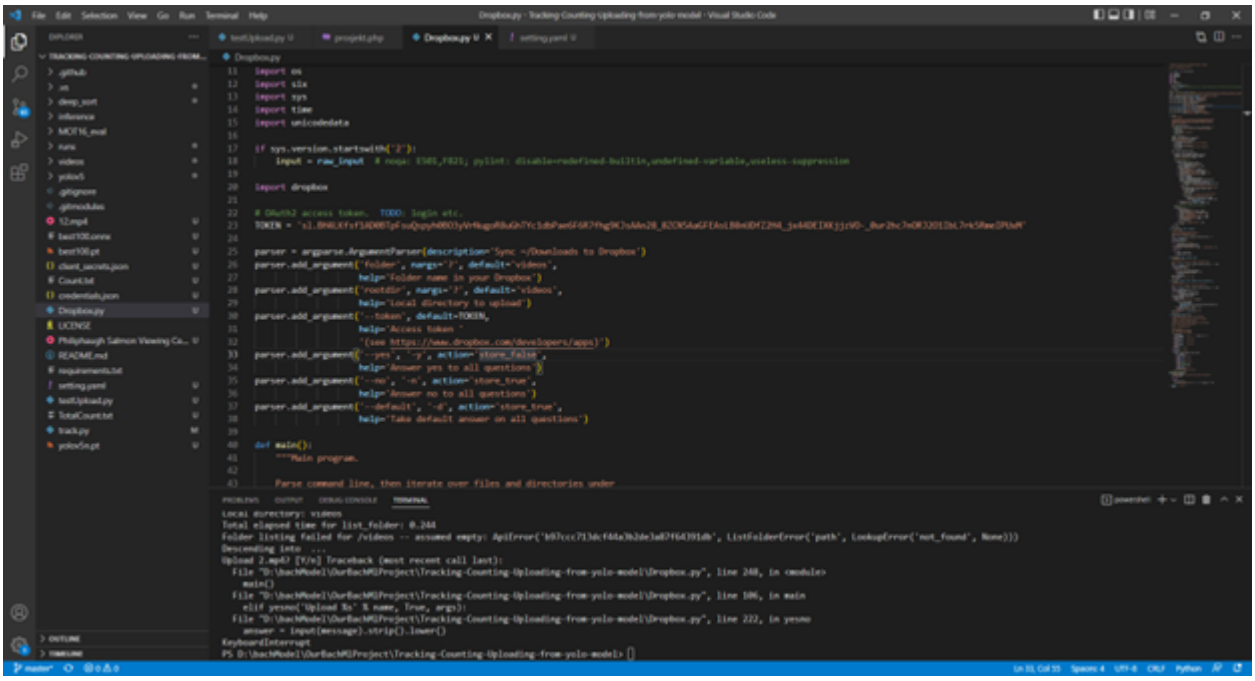
Figur 114: updown.py kode inn på vscode.

Der jeg har merket skal vi endre på, token skal være hva vår tilgangstoken er. Det vi fikk fra console appen på starten, det vises i Figur 115. (merkeringen vises i Oppsett av DropBox.docx)



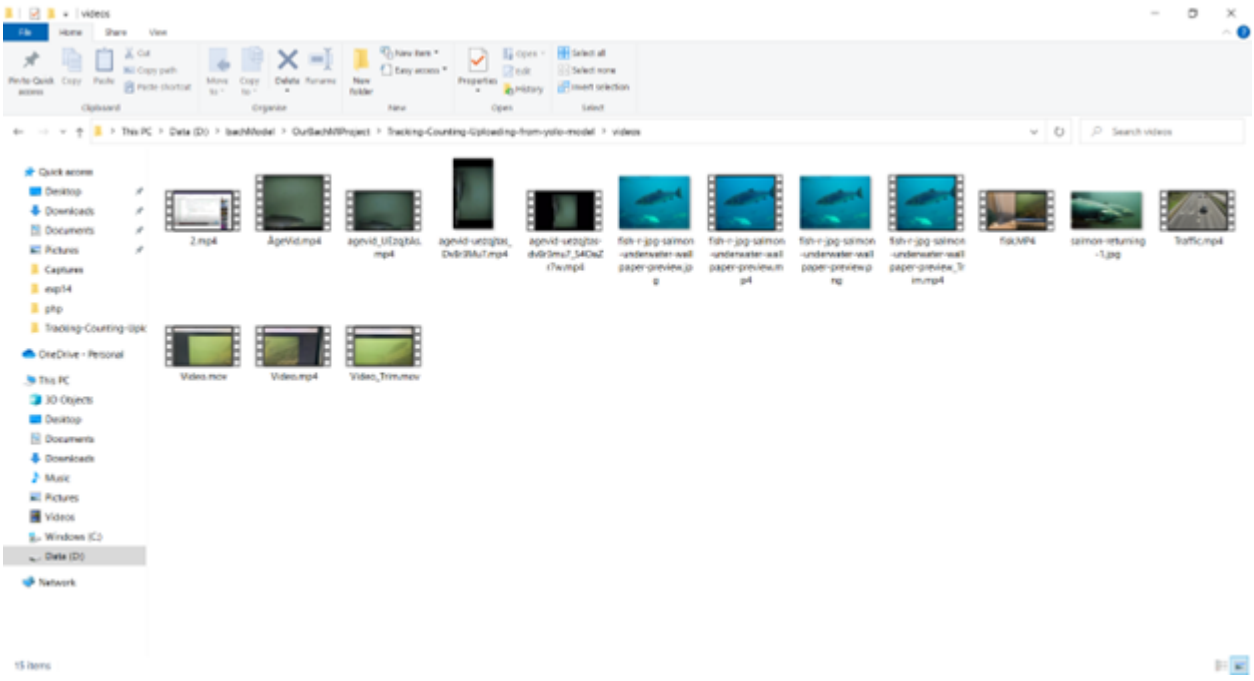
Figur 115: Endring av Tokens.

Da skal du laste opp alle filene som ligges i video folder til dropbox, for å gjøre dette må vi endre litt på scripten ovenfor til, det vises i Figur 116:



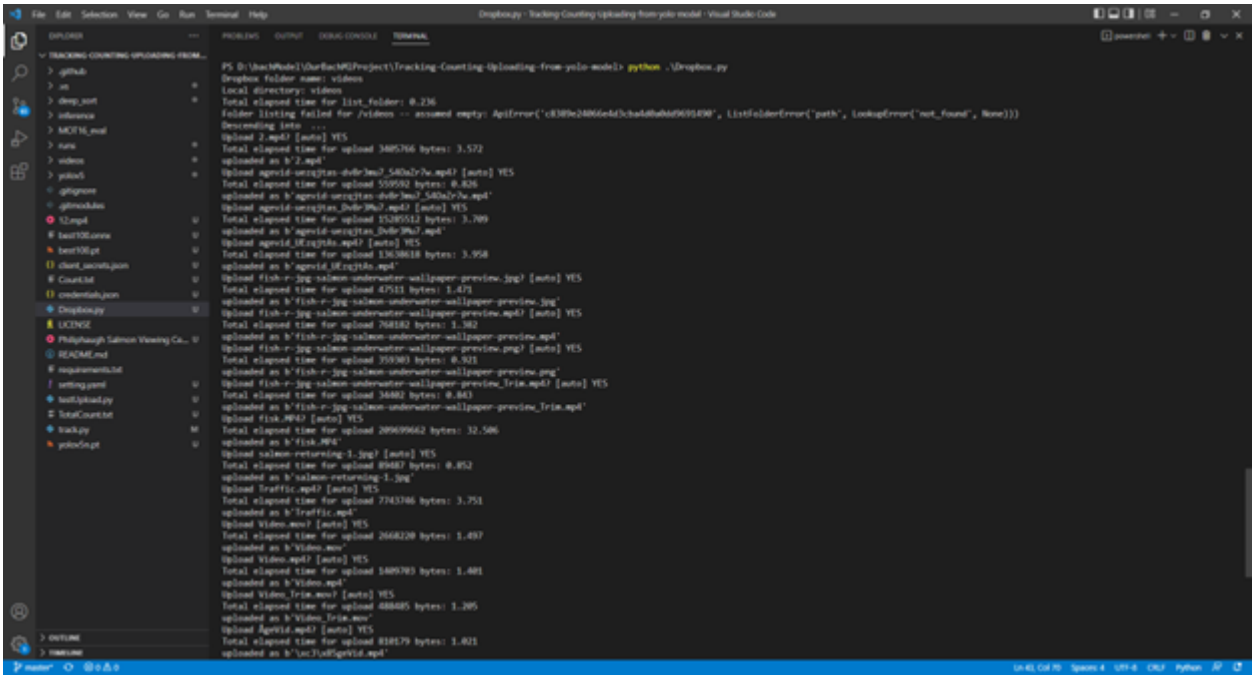
Figur 116: Endringen på hvor vi skal sende data.

Vi har endret på hvor vi skal laste filene opp i dropboxen med første pilen. Hvis vi har ikke dannet en videos folder, den skal gjøre det for oss. Neste pilen peker på hvor i pc-en, folder vi har lyst til laste opp ligger. Dette oppsettet har blitt gjort på windows, derfor har vi det sånn. Men hvis vi hadde vært på pi-en, hadde vi brukt '/home/bachelor/Documents/Yolov5_DeepSort_Pytorch/runs/track' istedenfor. Tredje pilen, gjøre at du trenger ikke å skrive ja på alle filene koden har lyst til å laste opp. Den skal bare laste opp alt som ligger i folderen automatisk (pilene vises i Oppsettet av DropBox.docx).
I video folderen har vi filer, det er vist på Figur 117.



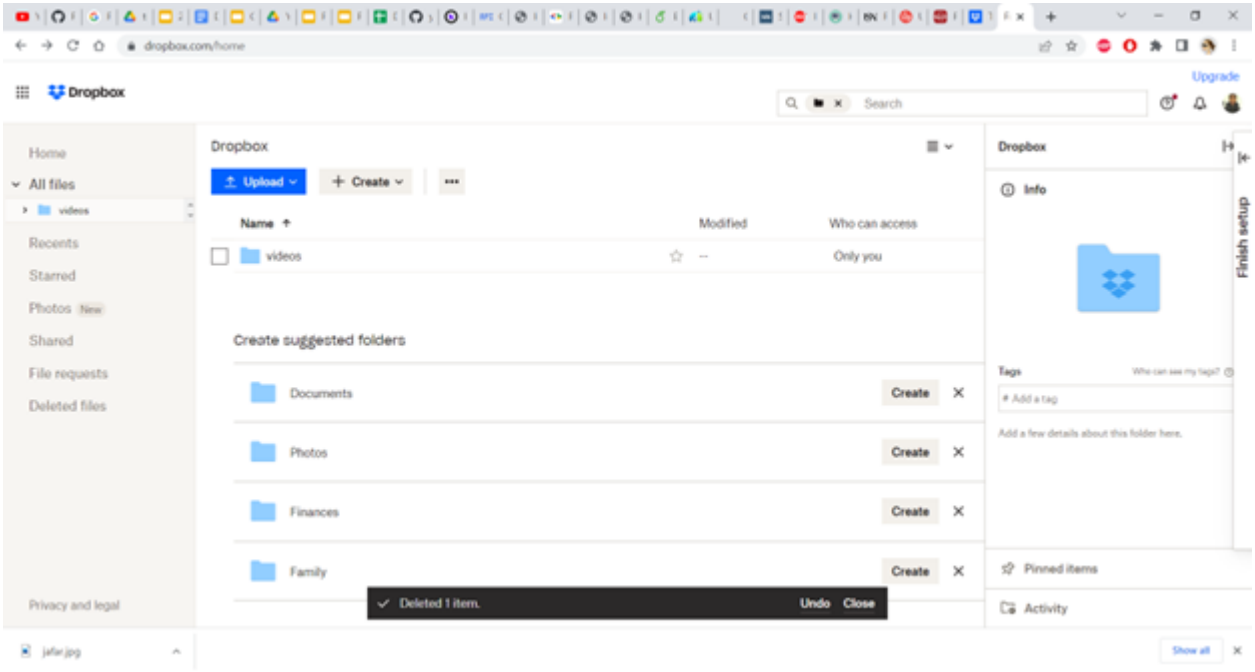
Figur 117: Alle videoer på video folderen.

Terminalen når jeg har skrevet kommandoen **python Dropbox.py** er vist på Figur 118:

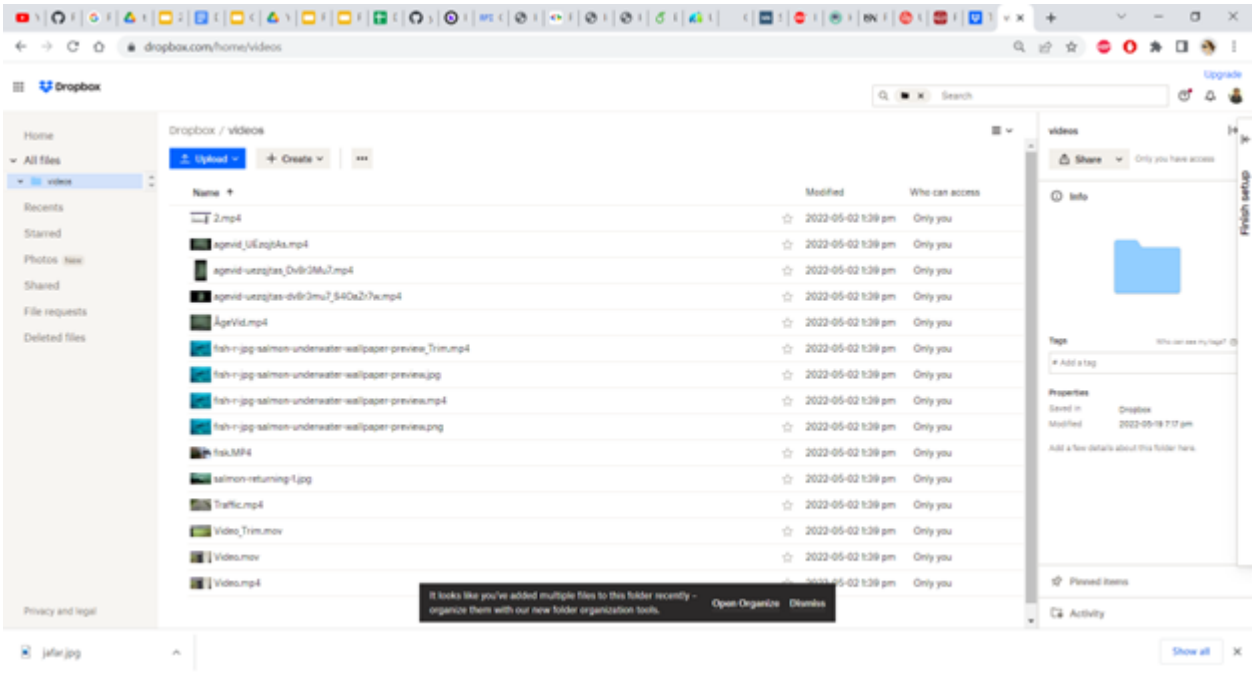


Figur 118: Terminalen etter vi har kjørt opplastningen.

Resultanten på dropbox er vist i Figur 119, 120 :



Figur 119: Resultanten på Dropbox.

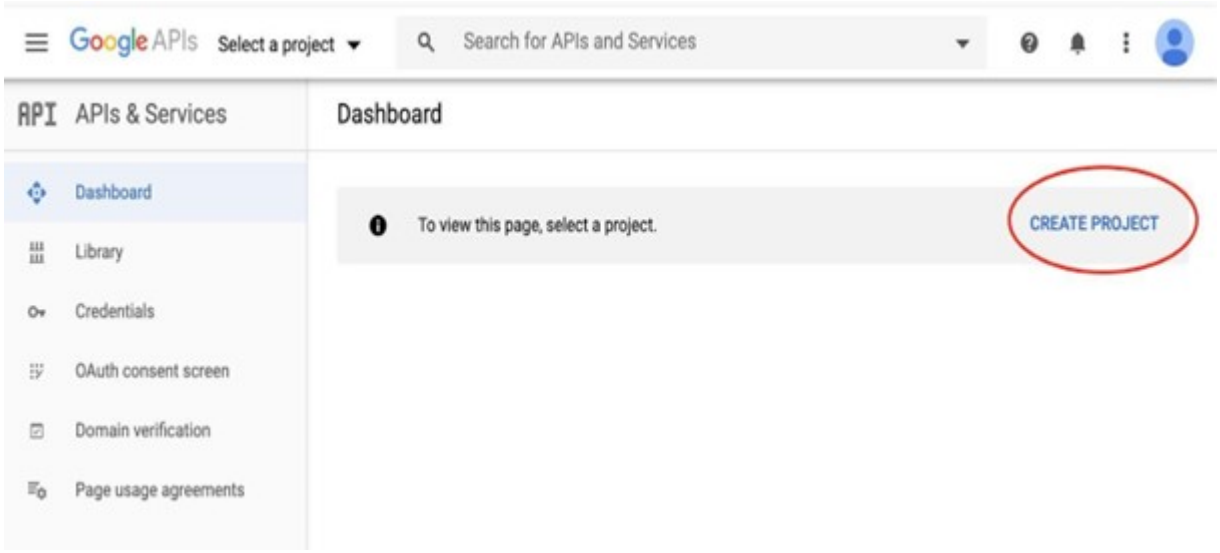


Figur 120: Inn til den nye videos folder.

C Oppsett av Google Drive

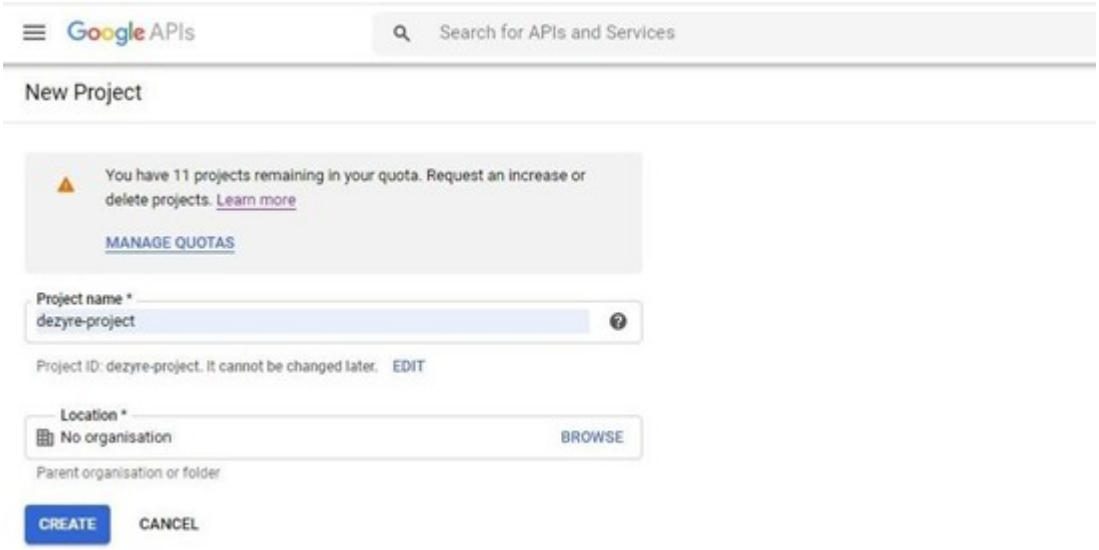
Målet er å koble raspberry pi-en til google drive, det en del steg vi må gjøre før vi kan begynne å kode. Først må vi få Authentication filer for Google Service API, som gjøres det mulig å få tilgang til google drive via koden vår. Hvordan man gjøres det er som følgende:

1. Du må først lage en ny projekt på Google Developer Console ved å trykke på "CREATE PROJECT", Figur 121 viser måten vi sette det opp.



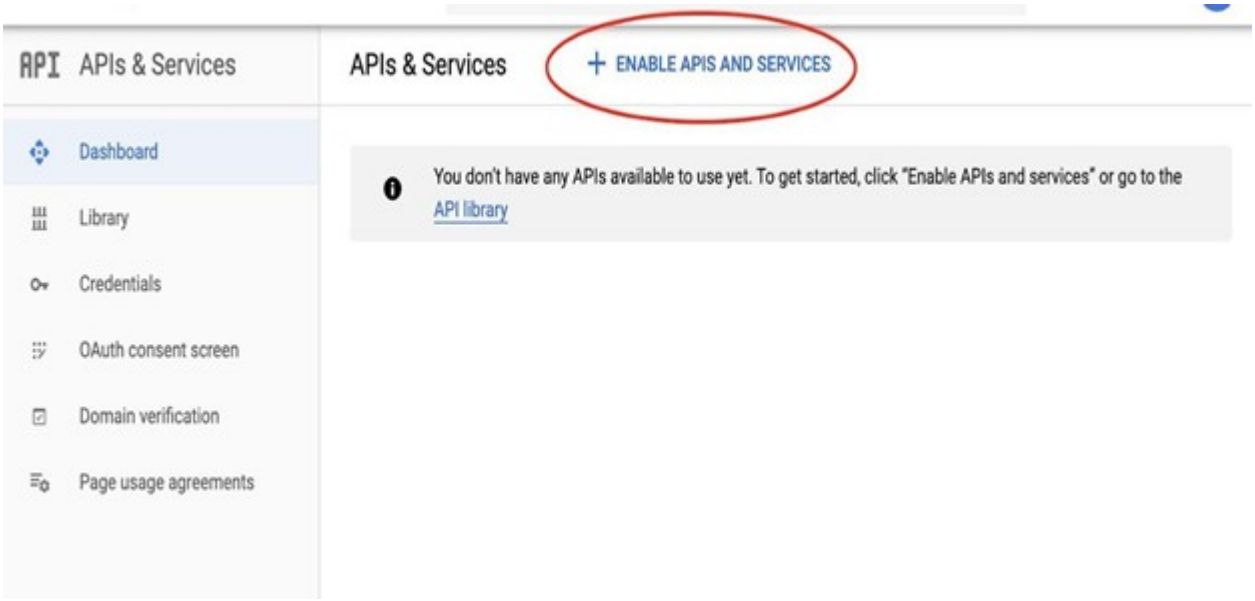
Figur 121: Måten vi lage et nytt projekt på Google drive.

I Figur122 har vi gitt navnet til vår projekt for FTOgoogleV2, fordi det var vår andre forsøk for å det til.



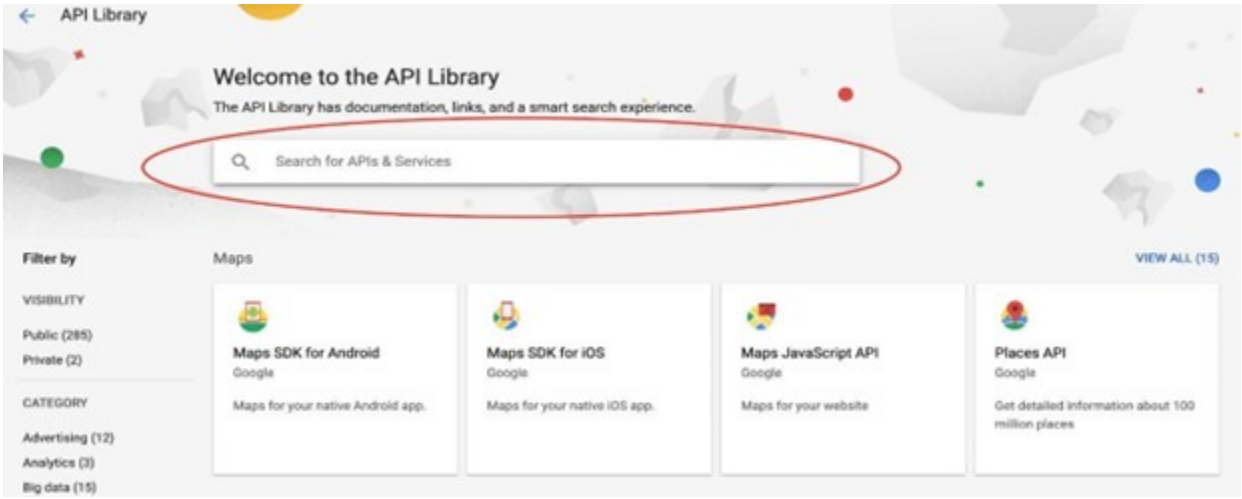
Figur 122: Gi navn til vårt prosjekt.

2. Vi må slå på API og Services ved å trykke på “ENABLE APIS AND SERVICES” som det vises i Figur 123.



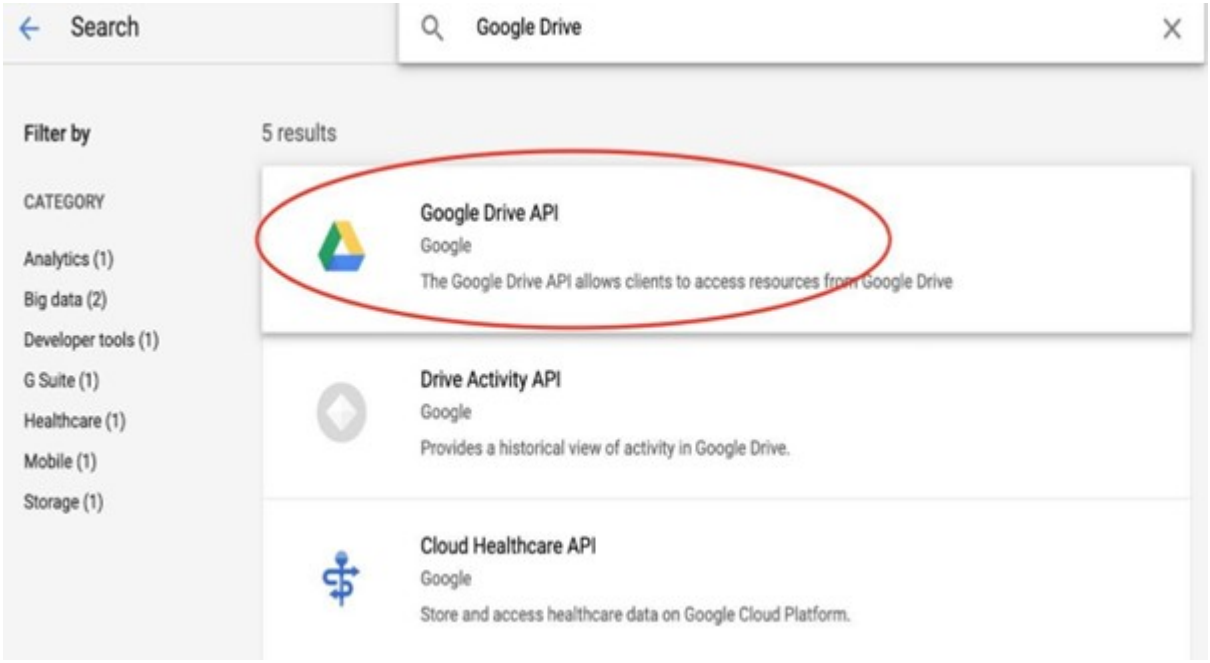
Figur 123: Slå på api og services inn i Google Cloud.

Da skal den spørre om hvilken API library vi skal bruke, se på Figur 124.



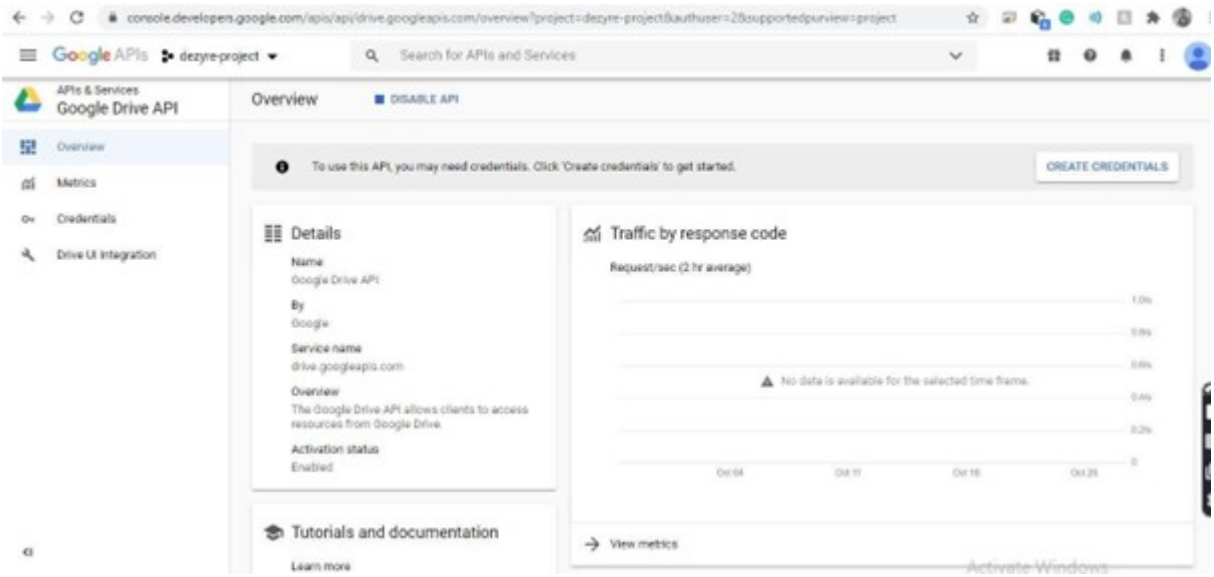
Figur 124: Her skal vi søke på hvilket api vi trenger.

Søk for “Google Drive” i den search bar og vi skal få svaret som ligger på Figur 125.



Figur 125: Merkering av Google Drive API.

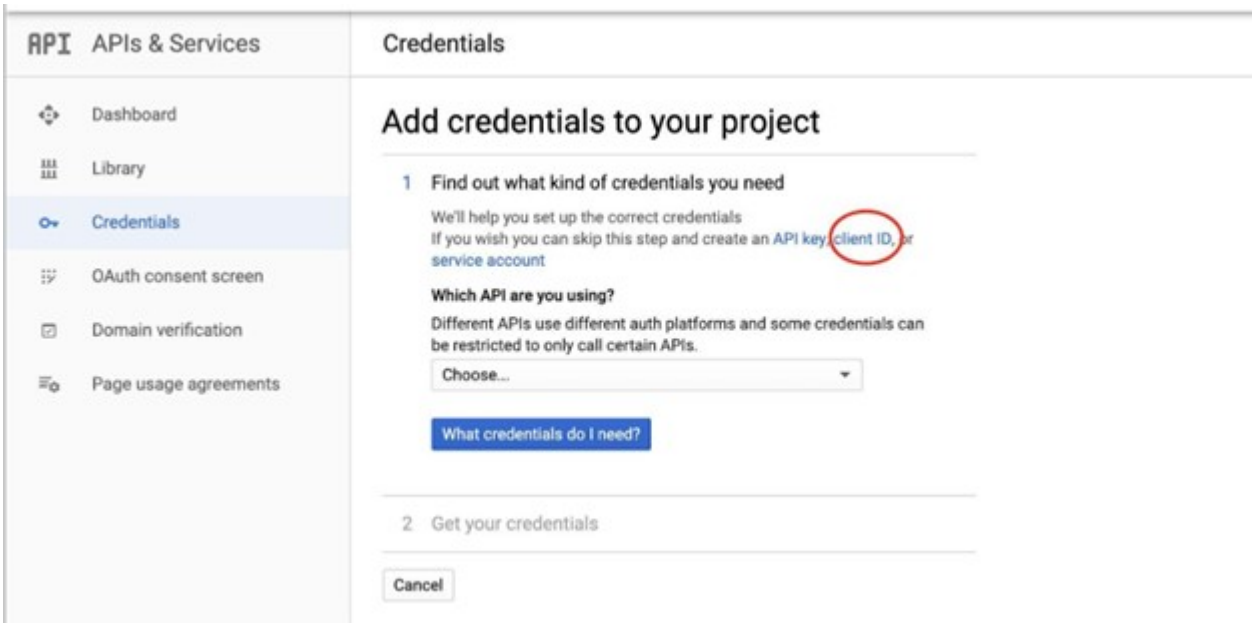
Trykk på "Google Drive API" og vi skal få samme side som på Figur 126.



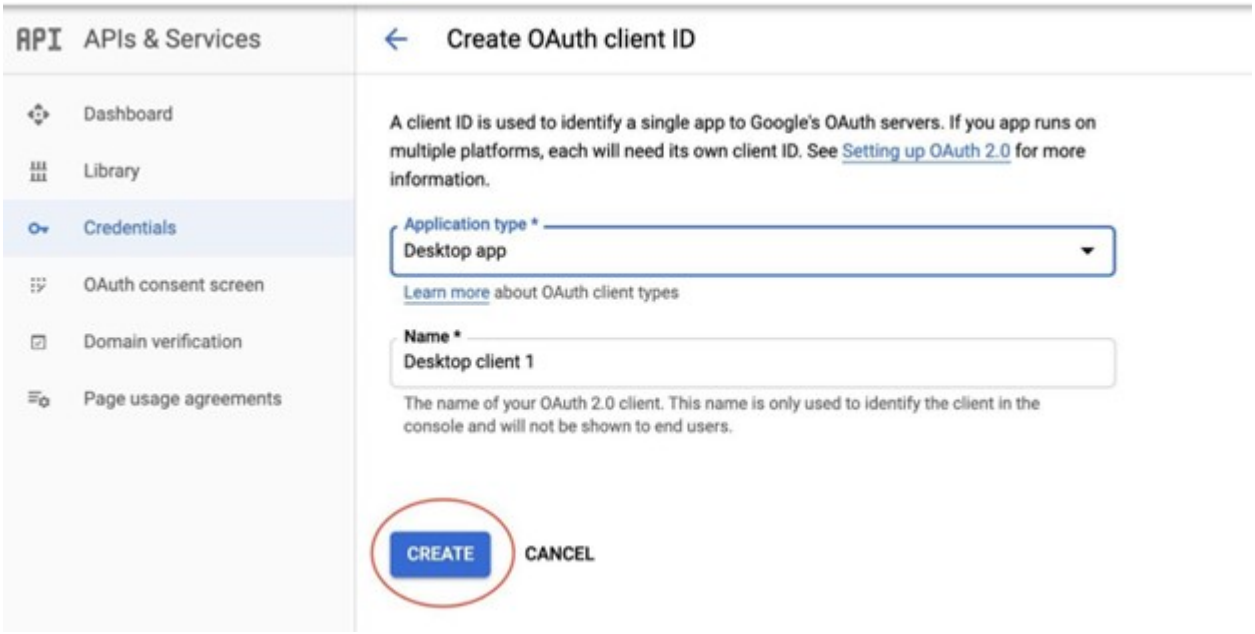
Figur 126: Side for å slå på API.

Trykk på “ENABLE”, da har vi slått på Google Drive API service.

3.Nå skal vi lage credentials. For å gjøre det skal vi trykke på “CREATE CREDENTIALS” icon som blir vist i Figur 127 og 128.

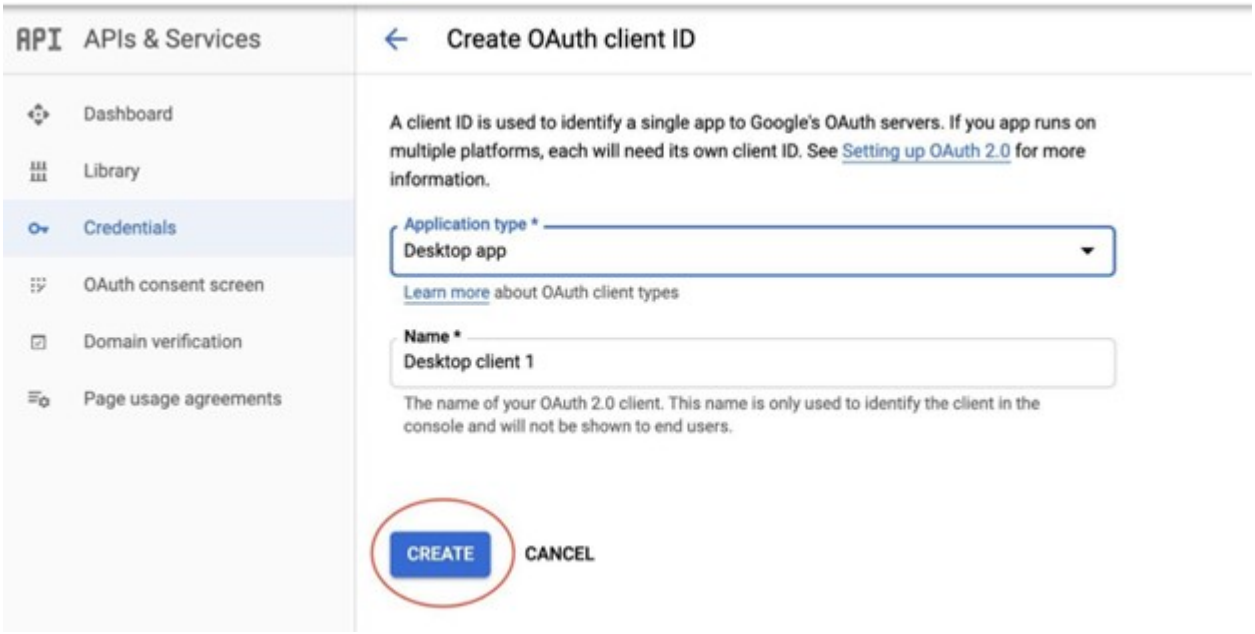


Figur 127: Trykk på Client ID.



Figur 128: Lag credentials.

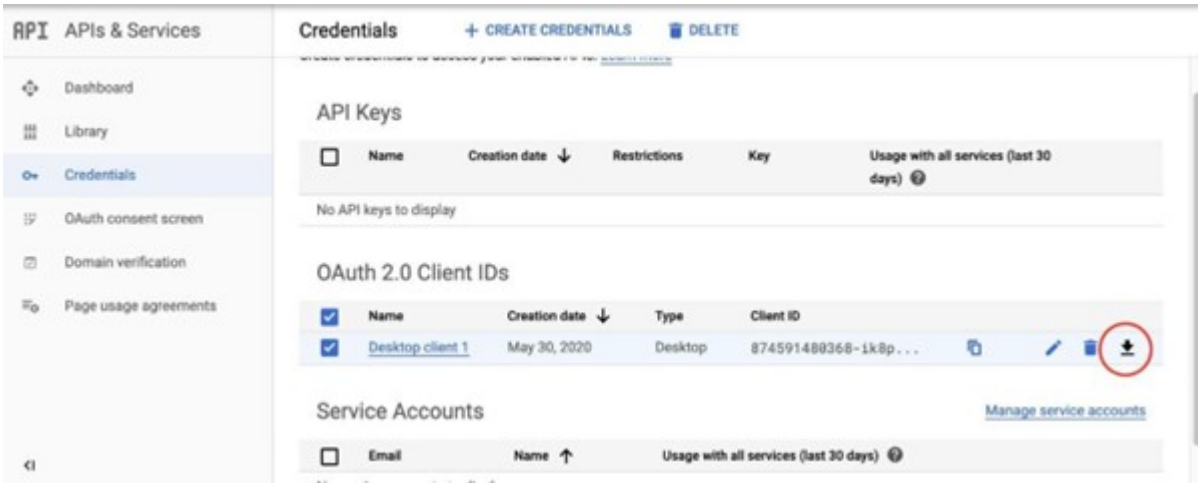
Som det vises i bildet ovenfor, vi skal trykke på "client ID". Det er viktig å lage en clientid, fordi det er det som python programmet trenger for å koble til. På application type* skal vi velge Desktop app. Da skal du trykke på "CREATE" og last ned JSON file som er vist på Figur 129.



Figur 129: Trykk på create.

En mer detaljert video ligger på oppsett av GoogleDrive

Den JSON fil er den vi trenger or å koble med via python kode. Navnet på filen skal være client_secrets.json. Hvis vi får en fil med navn client_secrets_(476056397614-llt10thbcn74tbrjc71sk64e2v13c7 der lang navn med tall er din client id, og den kan du fjernet, pass på at tilslutt navnet til filen blir client_secrets.json som vises på Figur 130. Legg filen på samme sted som yolo sporing og telling folder.



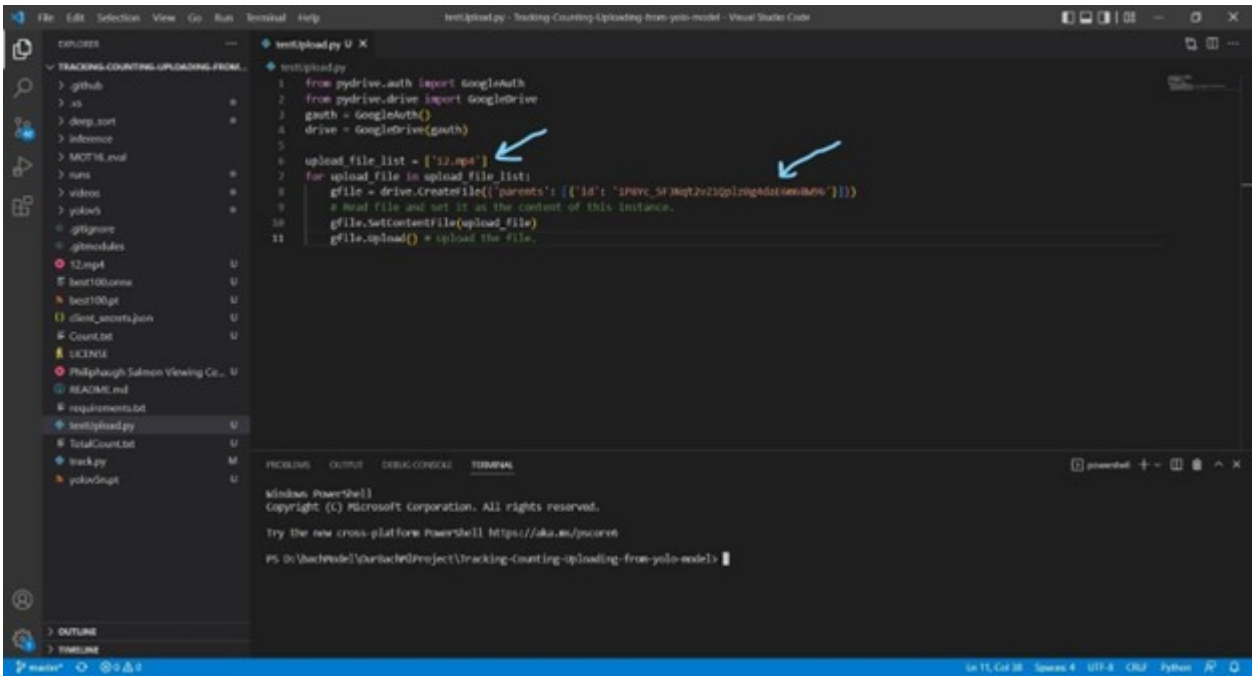
Figur 130: Last ned JSON fil.

Installer PyDrive Modul

Når vi har fått JSON fil for å ha tilgang til Google drive, skal vi installer en python biblotek. Åpne en ny terminal og gå til der JSON filen ligger, i vår tilfelle det er i samme folder som yolo/sporing og telling. Da skal du installere pydrive med **pip install pydrive**.

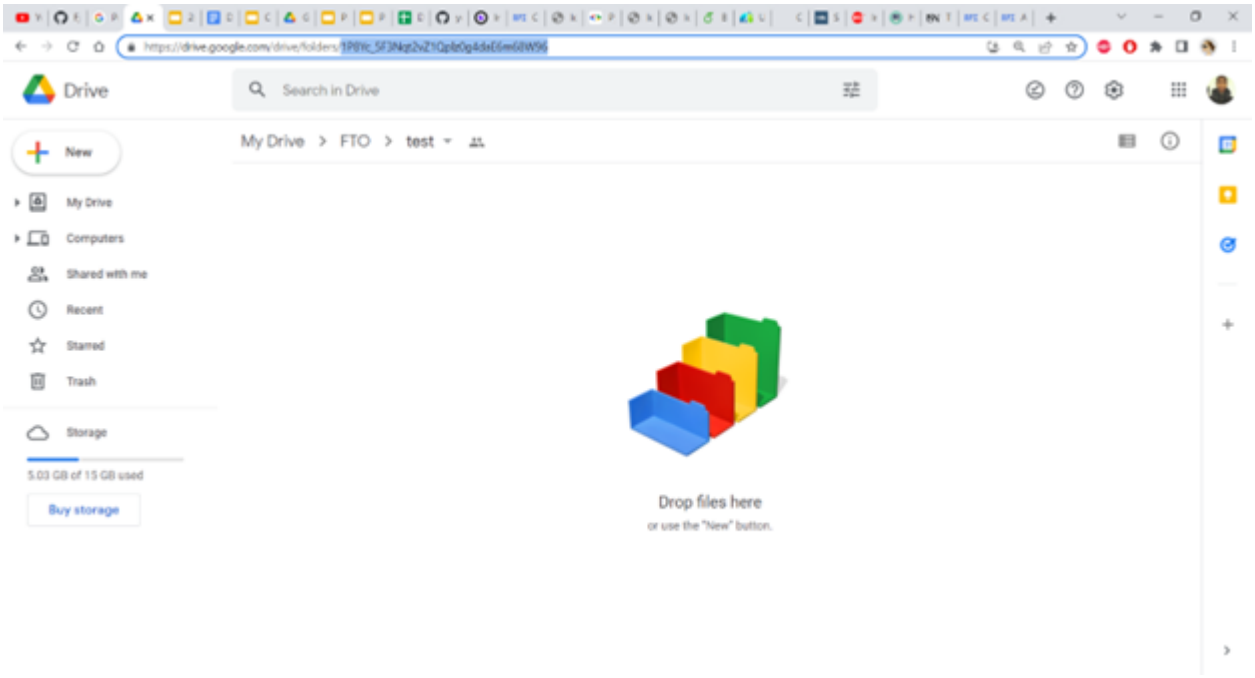
— PyDrive ved bruk av pip install pydrive.

Da skal vi lage en ny fil inni på folderen, og kall den for testUpload.py. Du skal skrive koden som ligger under i testUpload.py fil som vises i Figur131.



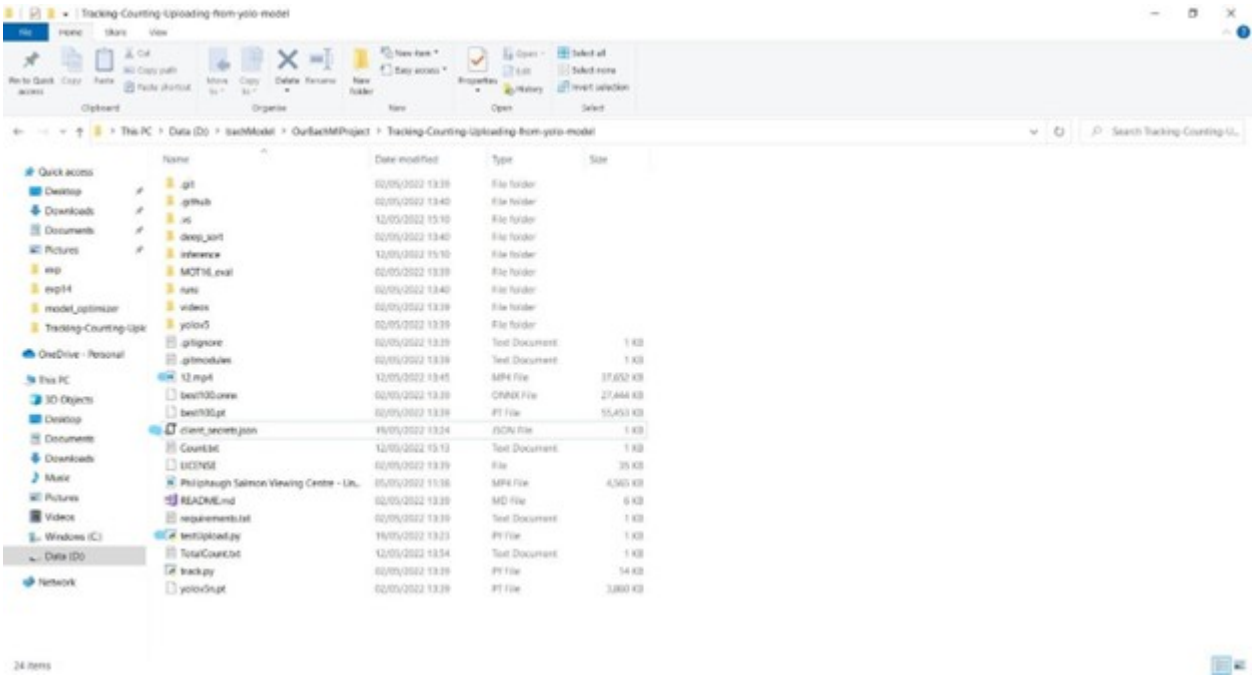
Figur 131: testUpload.py første endring.

Der 12.mp4 er bare en test fil vi skal sende, du kan ender det til hva som helst. Men det er viktig at det ligger i samme folder. Den andre peker er der vi spesifisere hvilken folder på google drive vi skal laste opp filen til. Hvordan man finner den id er å kopiere det som står i search bar når du er i folderen på google drive. Her har jeg brukt en test folder på google drive for å teste om filen har blitt lastet opp. Se på Figur 132



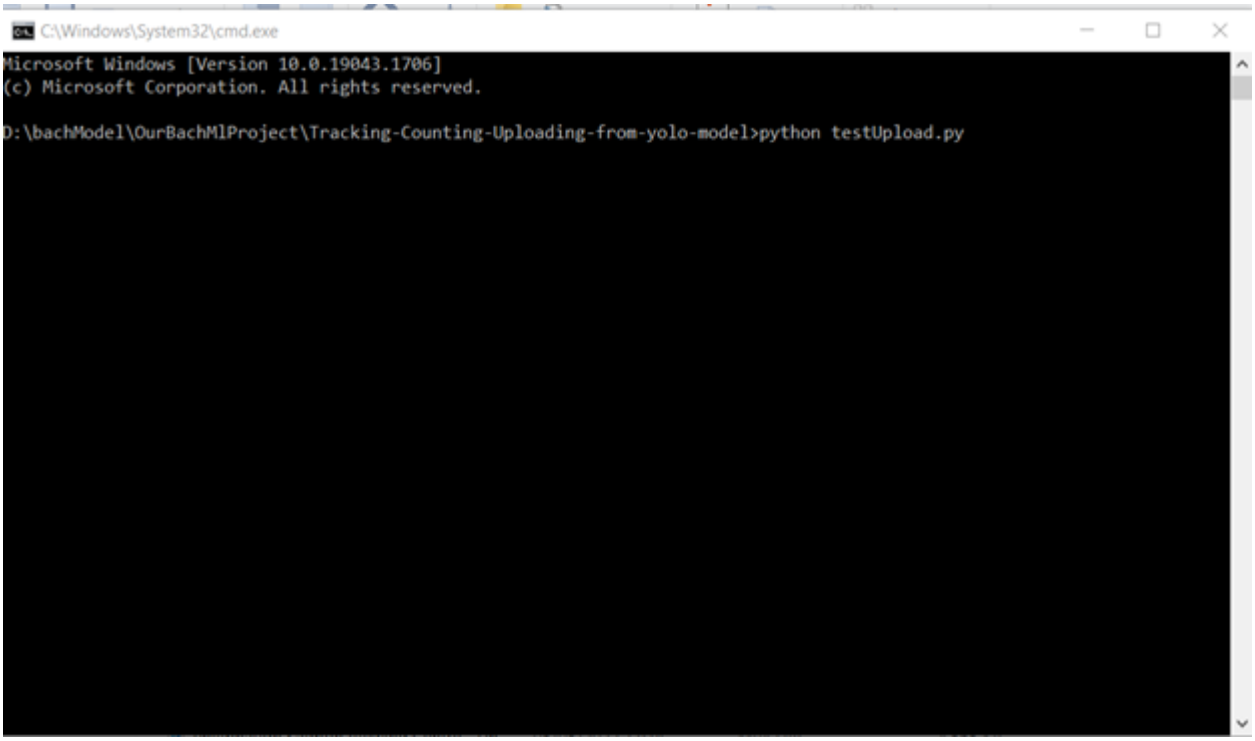
Figur 132: Test folder.

Før vi kjører programmet, skal fil strukturen din være som det er vist i Figur **??**, jeg har farget de filene som blir brukt.(Merkerte bildet ligger i Oppsett av Google drive)



Figur 133: Filstrukturen før opplasting til Google drive.

Nå skal vi kjører programmet , ved å skive kommandoen python testupload.py. se på Figur 134

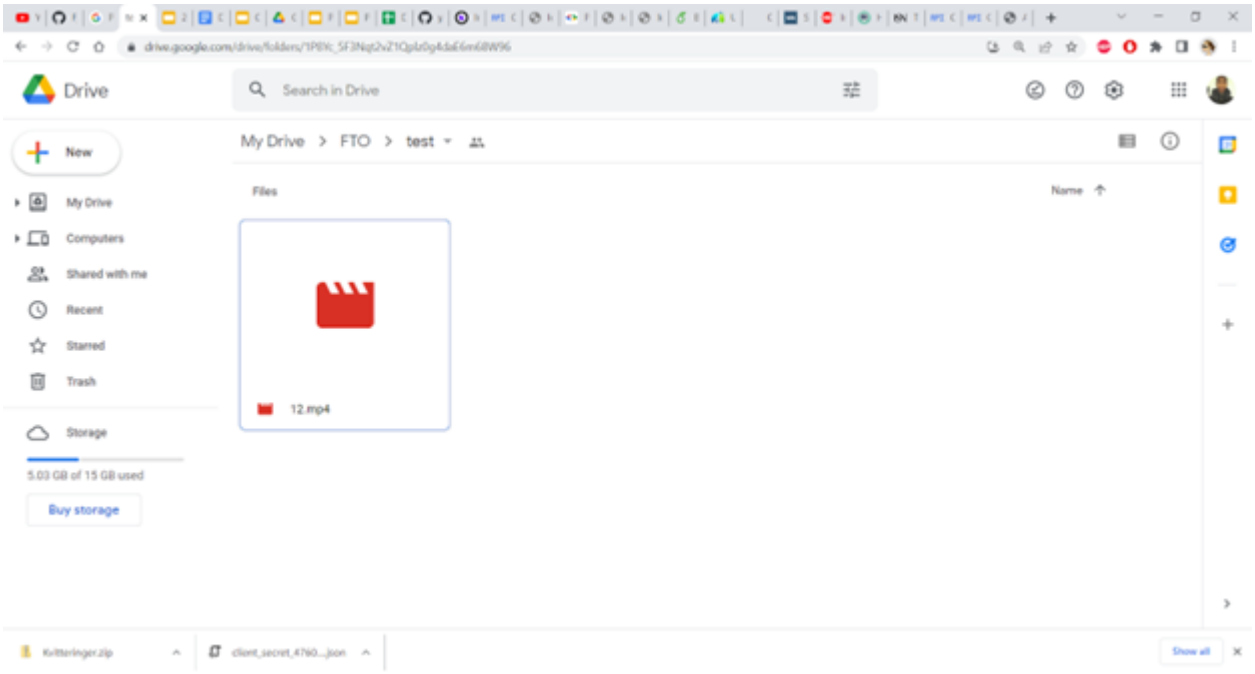


Figur 134: Python command for testUpload.py.

Da skal det åpnes en nettleser, og den videon nedenfor forklares hva du må gjøre for å koble opp.

En detaljert video ligger på Oppsett av GoogleDrive

Når vi kommer til side der det står The authentication flow has completed har vi da fikk koblet opp til driven. Hvis du går inn på test folder i FTO, skal vi har fått filen. Se på Figur 135.



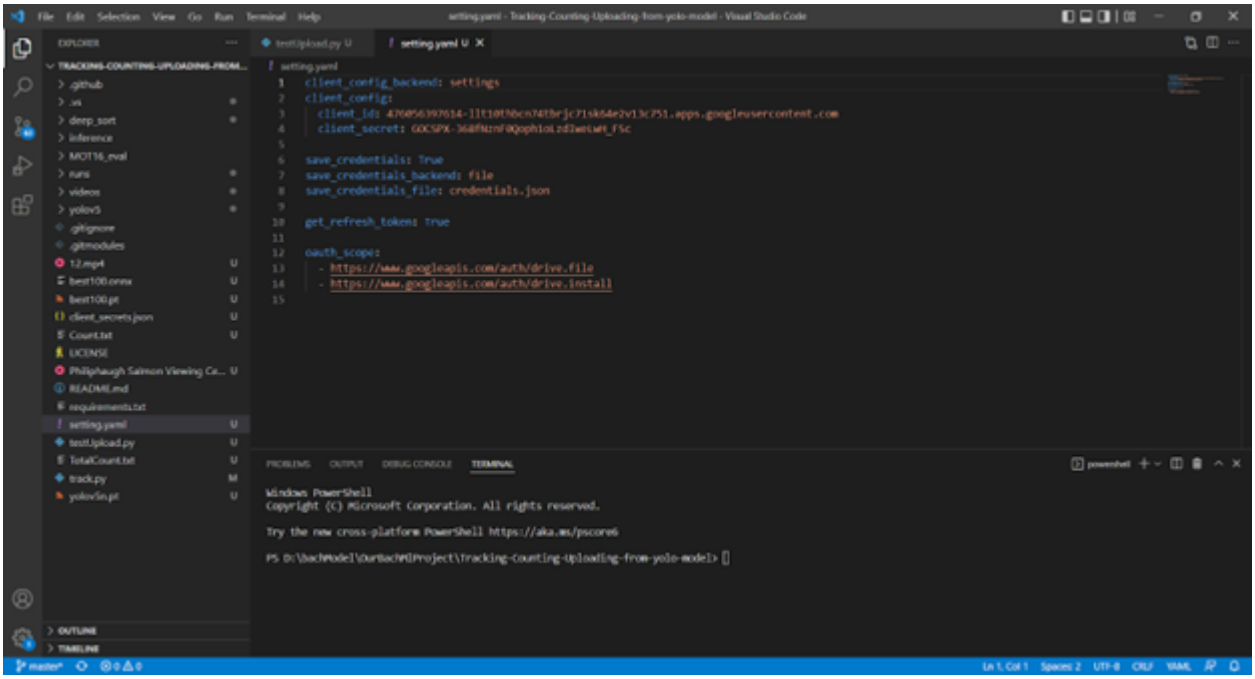
Figur 135: Test folder etter opplastning.

Problemet nå er at hver gang du kjører programmet, må du gå gjennom nettleseren for å skrive inn passord og gi tilgang. Det her blir et problem for noe som skal være automatisk. For å unngå det må vi lage en settings.yaml fil, som skal lagres alle credentials. Yaml filen vises i Figur 136:

```
client_config_backend: settings
client_config:
  client_id: your_client_id
  client_secret: your_client_secretsave_credentials: True
save_credentials_backend: file
save_credentials_file: credentials.jsonget_refresh_token: Trueoauth_scope:
  - https://www.googleapis.com/auth/drive.file
```

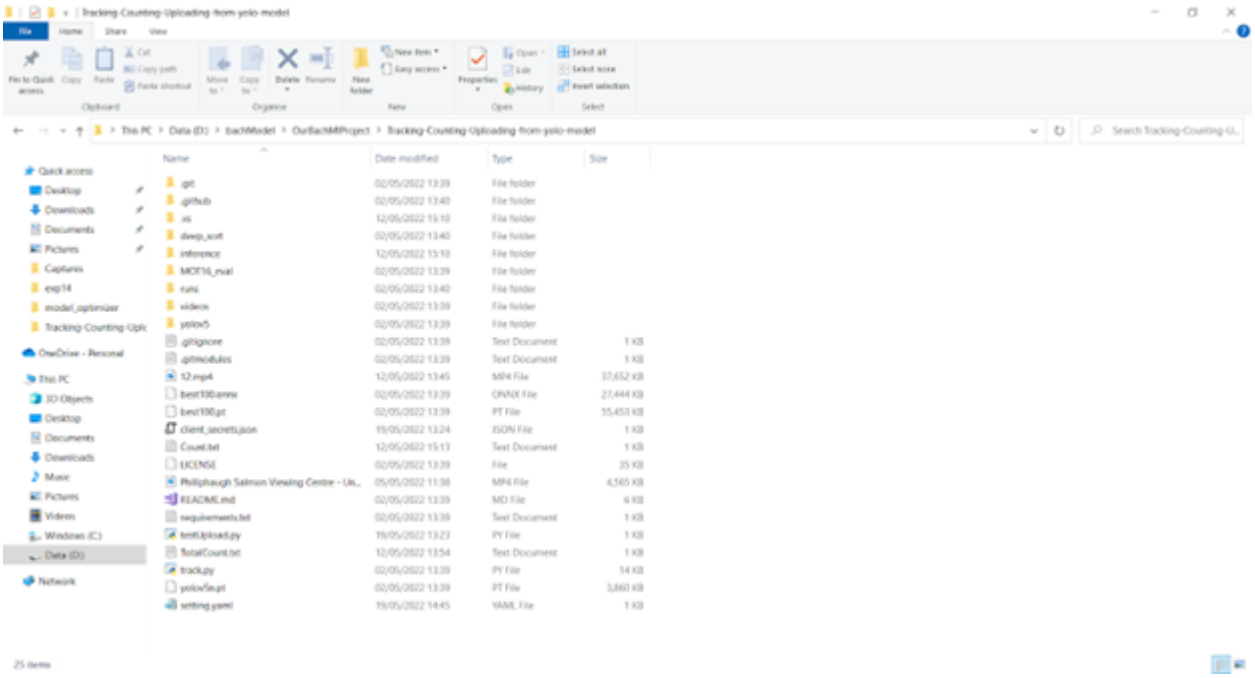
Figur 136: Standard YAML fil.

Det som vi skal endre er cliend_id: og client_secret, som du kan finne ved å klikke på redigeringsikon som vises nedenfor. Setting yaml filen skal være i samme folder som client_secrets.json, når du kjøre programmet på nytt, skal du gå gjennom nettleseren en gang til. Da blir en credentials.json fil dannet, og det er nå ikke nødvendig å gå gjennom nettleseren, det skjer automatisk. Men, før du kjører programmet på nytt må vi endre litt på testUpload.py. Se på Figur 137.



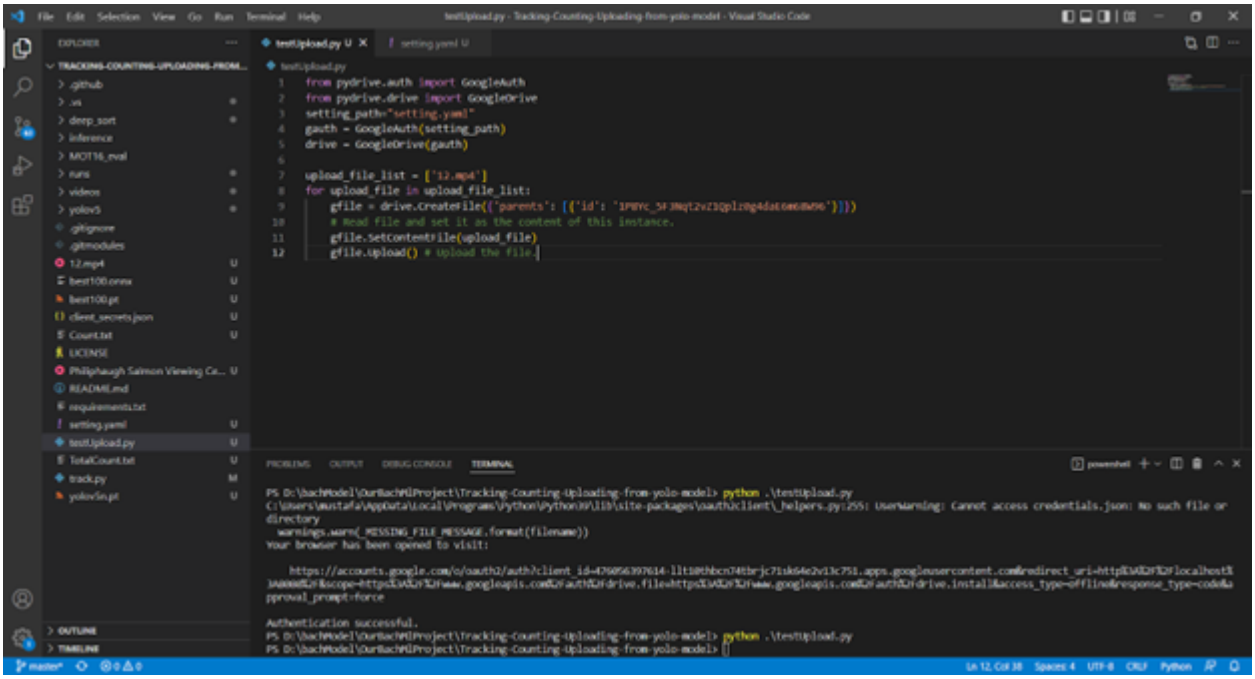
Figur 137: Redigert YAML fil.

Filstrukturen vi nå ha vises i Figur 138 før vi kjører programmet på nytt:



Figur 138: Filstrukturen før andre test.

testUpload.py vises i Figur139:

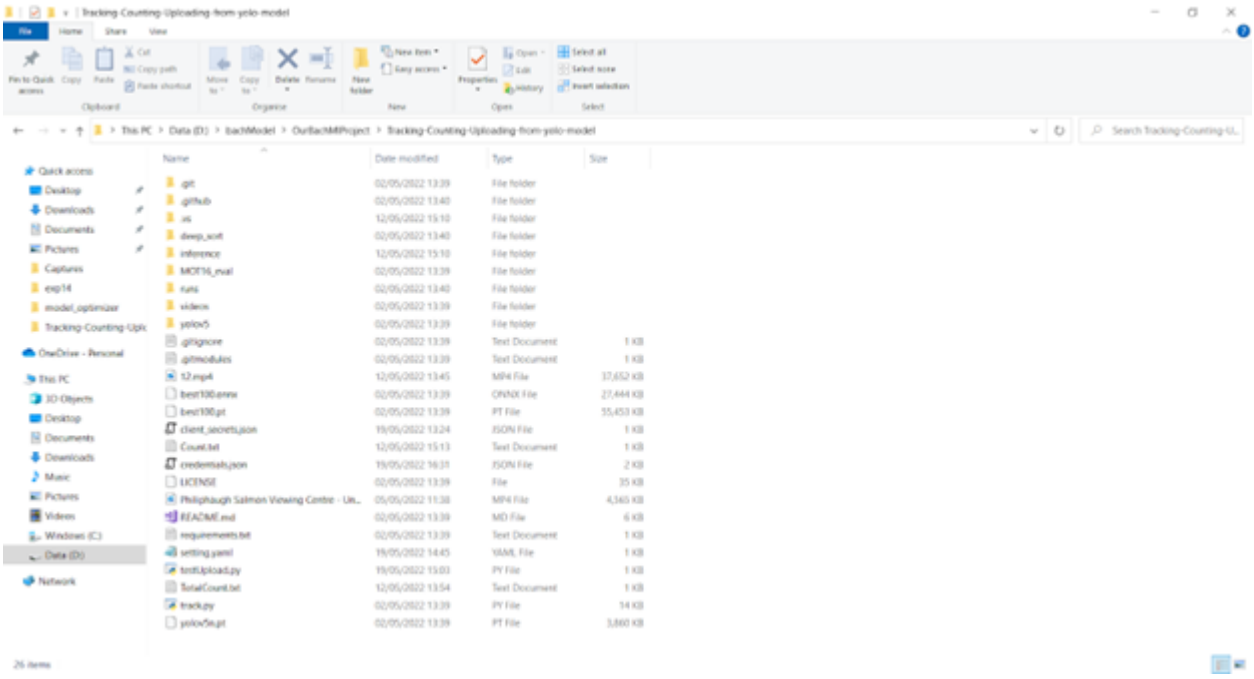


Figur 139: testUpload.py andre utgave.

Vi må endre koden for å inkludere setting.yaml. Da skal du kjøre programmet på nytt, den skal åpne en nettleser som du skal gjøre det samme som før. Men denne gangen skal du få spørsmål om samtykke, der skal du godtatt. Til slutt skal du få en side der det står The authentication flow has completed.

En mer detaljert video ligger på Oppsett av GoogleDrive

Da skal vi få en credential.json og vi trenger ikke å gå inn på nettleseren for å laste opp, filstrukturen vises på Figur 140.



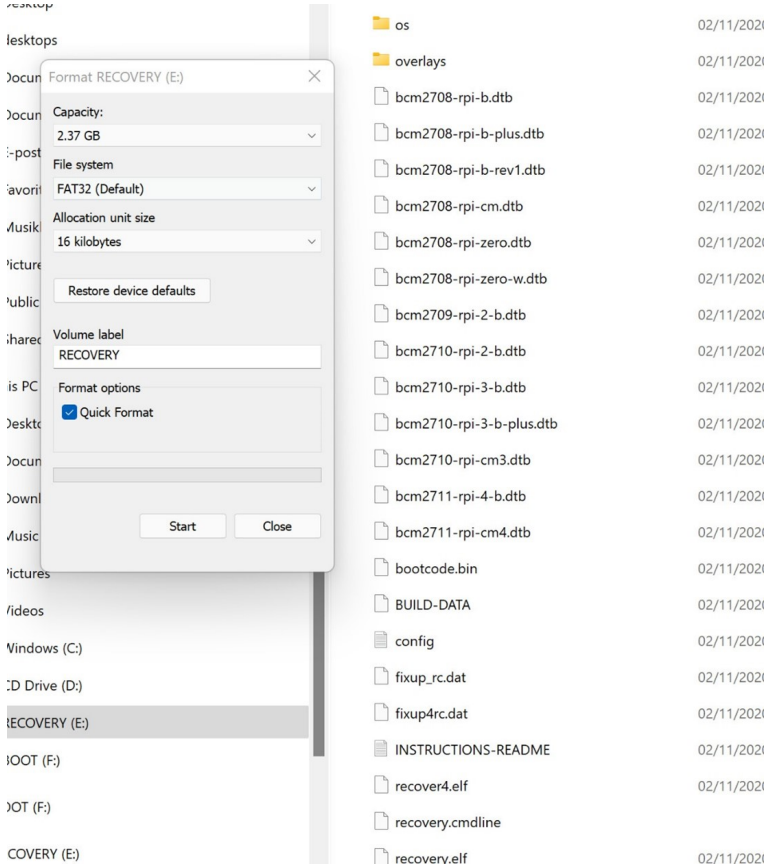
Figur 140: Filstrukturen på slutten.

De filene jeg har merkert, er de vi trenger for å laste opp filer fra pc til google drive via python.

En mer detaljert video ligger på Oppsett av GoogleDrive

D Oppsett av Raspberry pi 64 bits

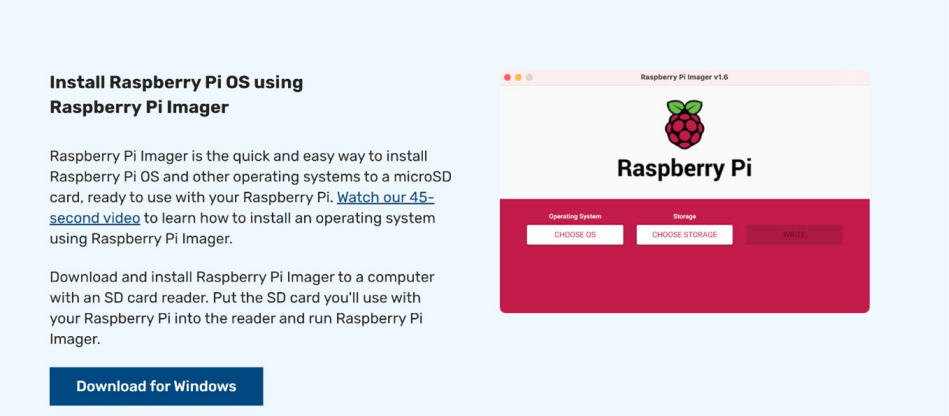
- For å oppgradere OS-en fra 32 bit til 64 bit, vi trenger å slette innholdet av gamle SD kortet som 32 bit var installert. Først, vi må putte SD mikrokortet i kortleseren og da vil vi få en liten skjerm som vises nedenfor i Figur 141



Figur 141: Format av SD kortet.

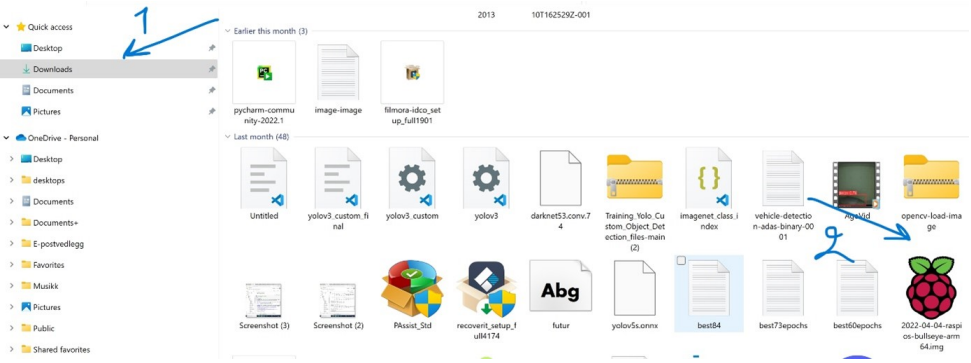
Når vi har fått opp denne siden, neste vi må gjøre er kryss av på Quick format også start, da sletter det alt data-en du har inn i mikrokortet. En ting må gjøres før du sletter, sikre deg at du ikke trenger dataene du har inn i mikrokortet.

- **Laste ned OS 64 bit**
  I denne steg, vi trenger å besøk hovedsiden av raspberry pi for å laste ned OS 64 bits. Med linken under kommer vi til siden som vist på bildet nedenfor. Trykk på download knappen for å laste ned til Windows og den havner i nedlastinger mappen vist i Figur 142. `https://www.raspberrypi.com/software/`



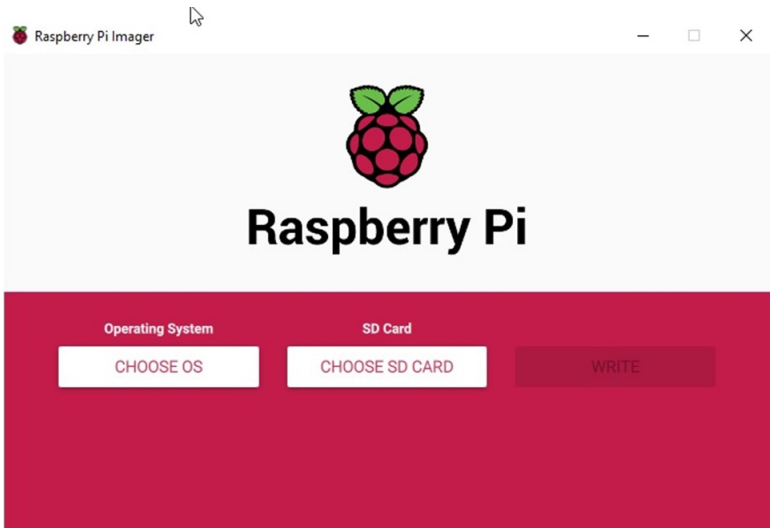
Figur 142: Filen plassering etter lasted ned

- Filen Havnet i nedlastinger mappen. Nå mikrokortet skal settes inn og filen skal overføres til SD kortet vist i Figur 143.



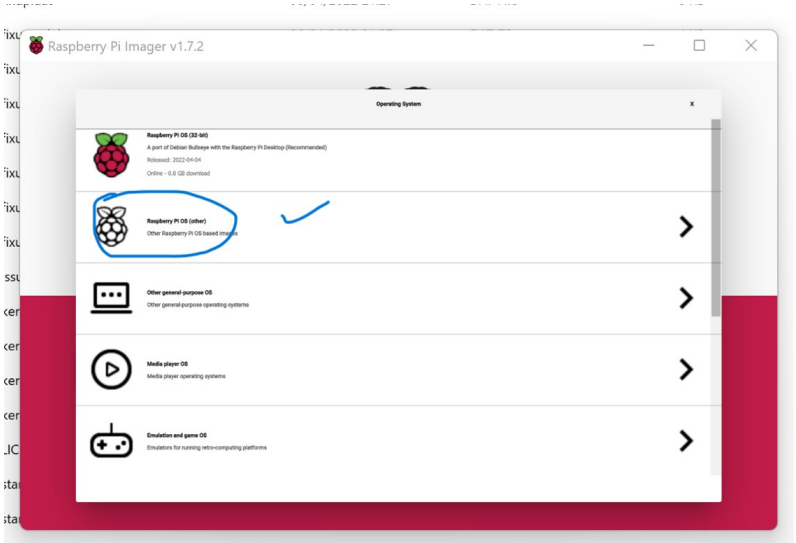
Figur 143: velg av OS type

- Etter at filen ble sendt til mikrokortet. Da er det noen instruksjoner som må gjøres for å installere OS 64bits.
  - Velge første alternativet ‘’Choose OS”144



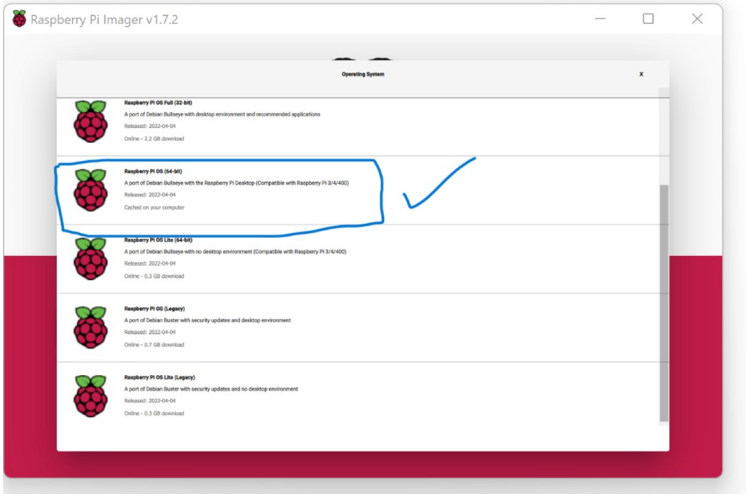
Figur 144: Finne 64 bits OS

  - Nå, vi må velge raspberry pi OS (other) vist i Figur 145.



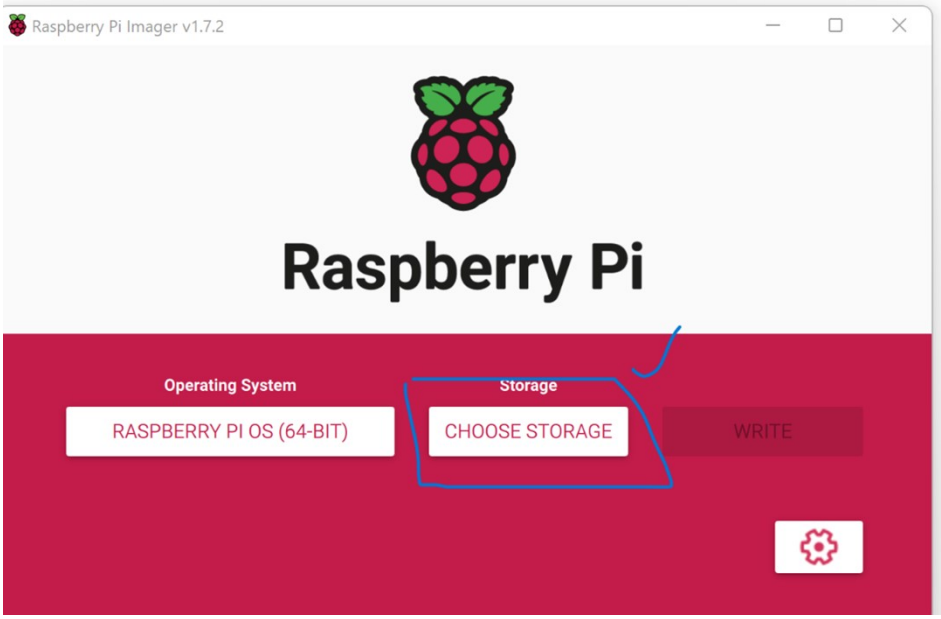
Figur 145: Valg av 64 bits OS

– Etter valget, vi får en ny side hvor vi velger raspberry pi OS (64-bit) vist i Figur 146.



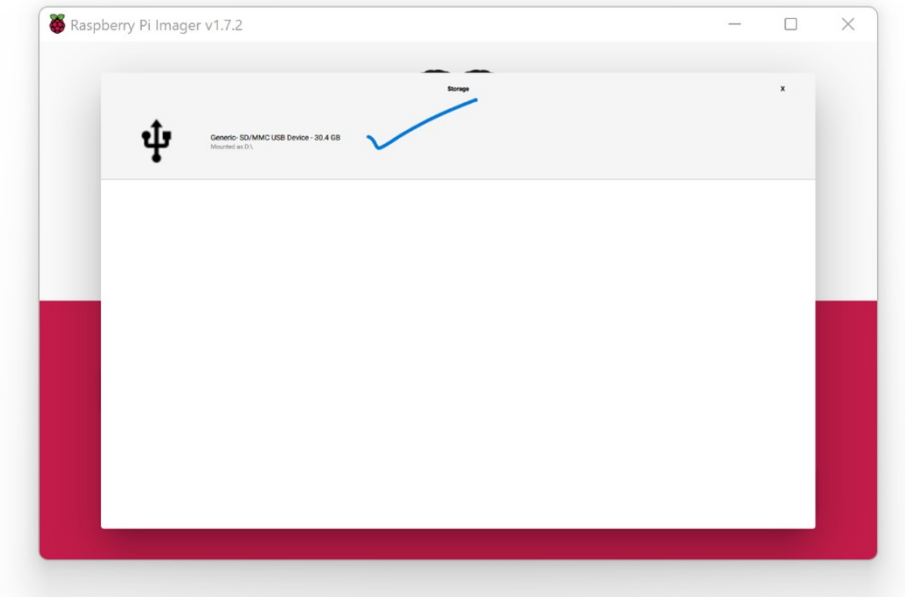
Figur 146: Velge av lagring

– Neste er det å velge ''Choose storage" vist i Figur 147.



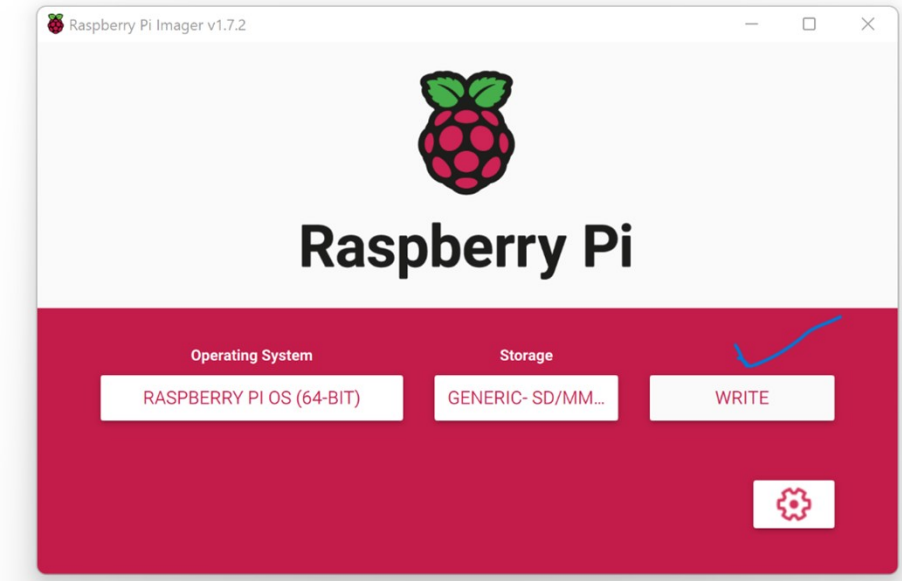
Figur 147: Velge SD kortet

– Velg Generic SD/MMC USB device vist i Figur 148.



Figur 148: Write for å installere OS

– Etter siste valget, så velger vi ‘’write” for å installere OS-en vist i Figur 149.

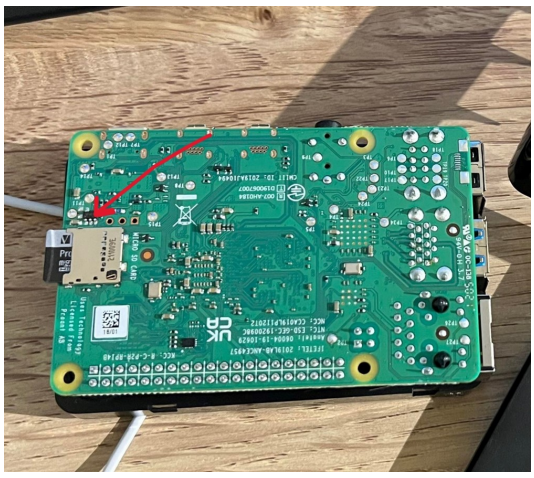


Figur 149: Installing av OS

– Da er det installasjonen av OS 64-bit i gang. Det vil ta litt tid før den er installert i mikrokortet vist i Figur 150.

Figur 150: Sett in av SD kortet

– Nå, OS er installert i mikro kortet. Da er det på tide å konfigurere raspberry pi fysisk. Finn mikrokort kontakten, sett mikrokortet forsiktig inn. Når den sitter godt, da er det bare starte raspberry pi og resten av oppsettet er samme som 32 bits OS som vi viste i Figur .

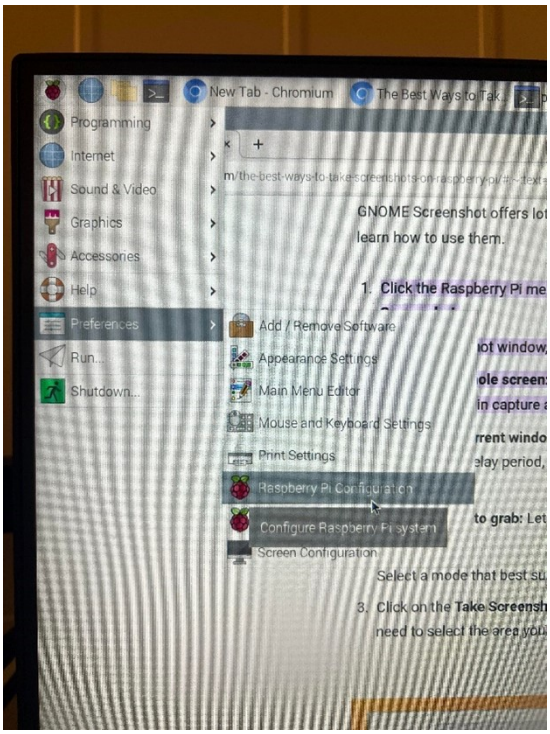


Figur 151: Caption

E Oppsett av VNC:

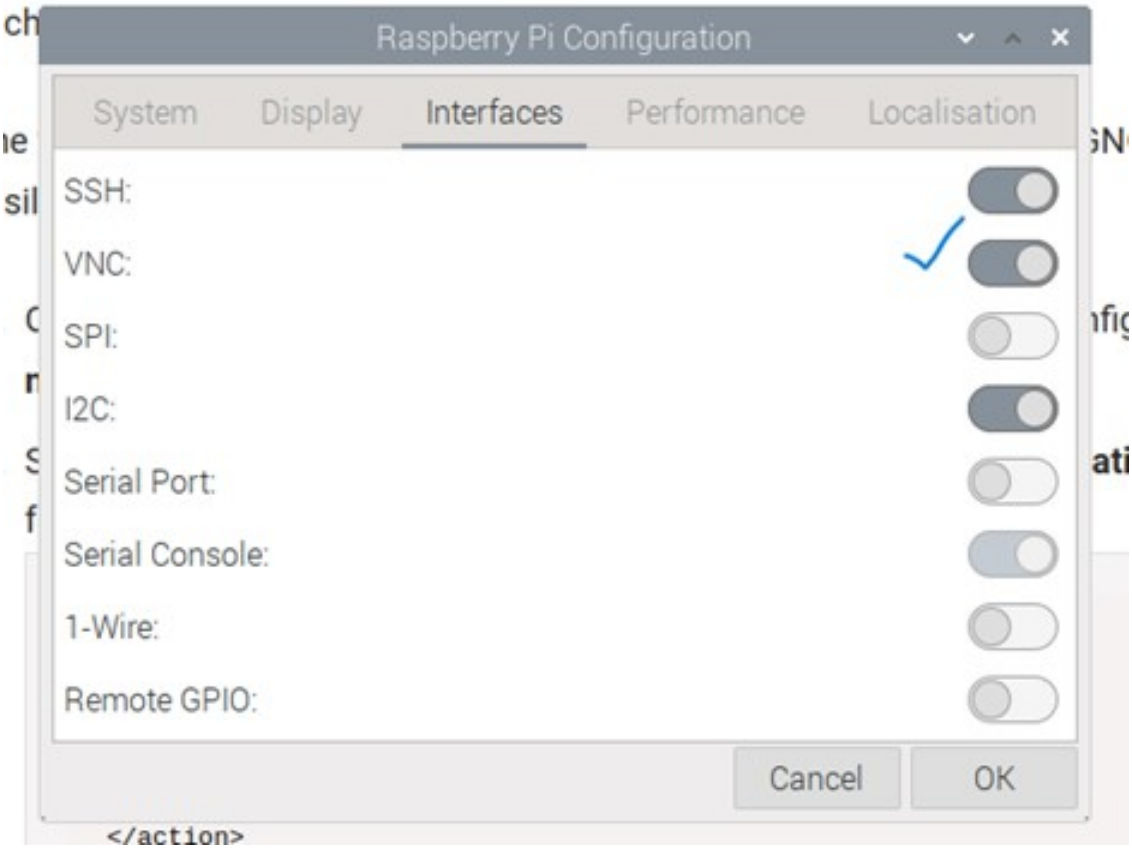
Her går vi gjennom om hvordan å sette opp VNC. Aller første vi trenger er å aktivere VNC i raspberry pi-en og den kan vi gjøre både gjennom terminalen og fysisk fra pc-en.

1. Vi går til pi logoen som er på hjørne av skjermen og trykker vi på det, så velger vi Preferences, også velger vi raspberry pi konfigurasjon vist i Figur 152



Figur 152: Komme fram til VNC

2. Etter at vi valget konfigurasjon, så dukker opp en ny meny. I menyen, aktiverer vi VNC vist i Figur 153

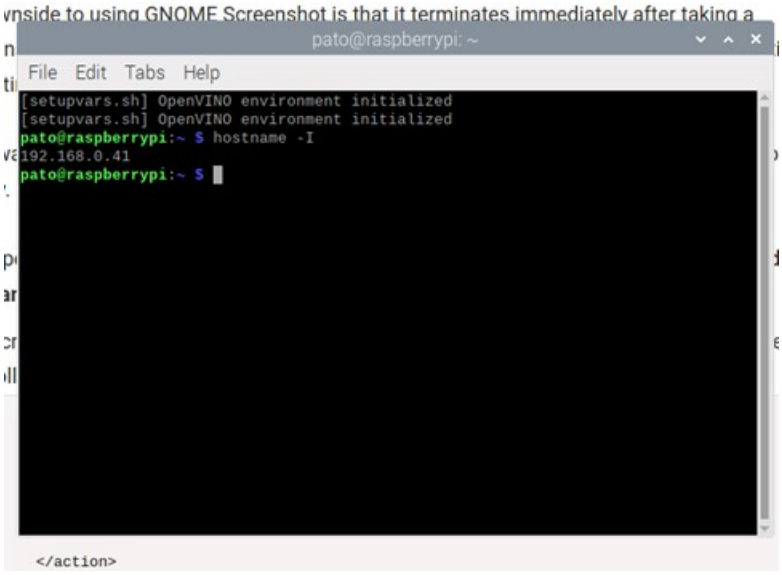


Figur 153: Slå på VNC

3. Nå må vi slå på VNC serveren og det vi trenger er å åpne terminalen i raspberry pi. I terminalen skriver vi vncserver og trykker enter.

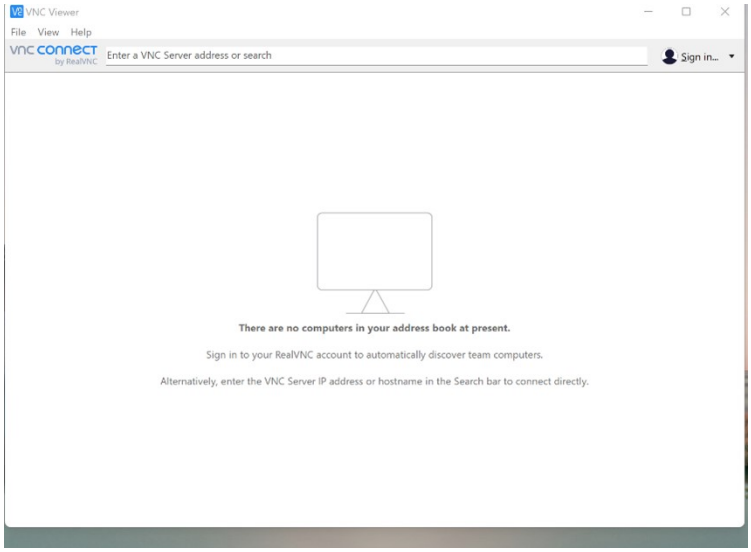
4. Nå som VNC er aktivert på raspberry pi, vi må åpne vnc viewer i pc-en og kravet er at vi må ha VNC i PC-en.

5. Neste er å åpne terminalen i raspberry pi for å finne IP adressen vist i Figur 154.



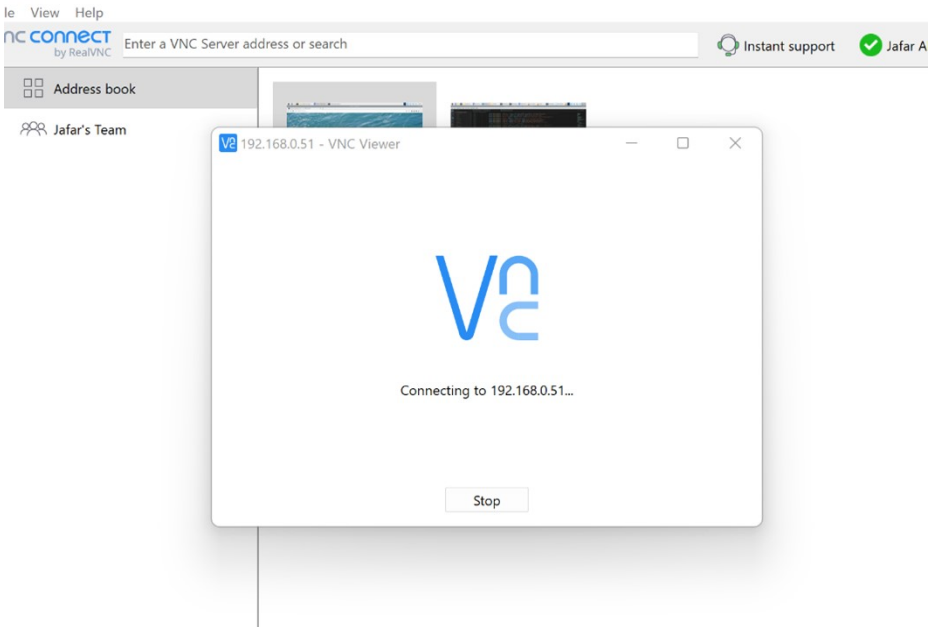
Figur 154: Måten å finne IP adressen i pi

6. Etter at vi har aktivert og funnet ut IP adressen, så går vi til PC-en, der åpner vi VNC vist i Figur 155.



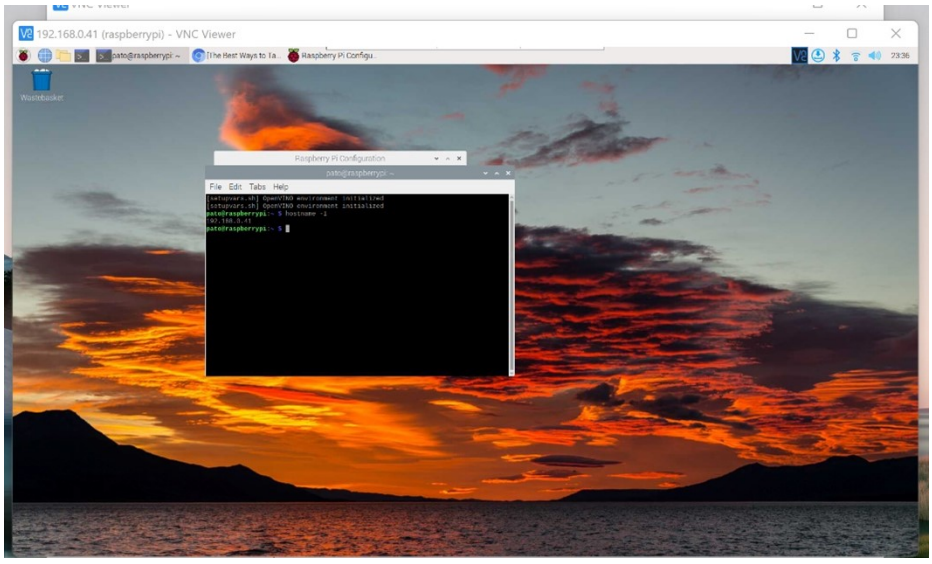
Figur 155: Skjermen som skal IP adress limes

7. Nå, det vi trenger er å kopiere IP adressen og limer vi inn i søkelinje til VNC viewer vist i Figur156.



Figur 156: Tilkobling

8. Når tilkobling er vellykket, så laster inn pi skjerm i VNC Viewer som vi ser nede i bildet på Figur 157. Da har vi full kontroll over pi-en.

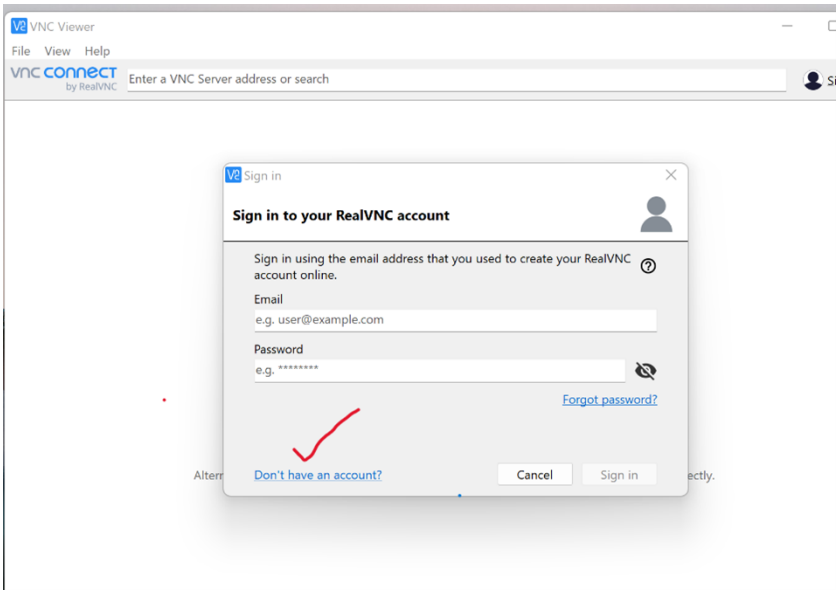


Figur 157: Raspberry pi terminal i PC-en

For å kunne jobbe med raspberry pi fra hvor som helst, vi trenger å lage konto hos

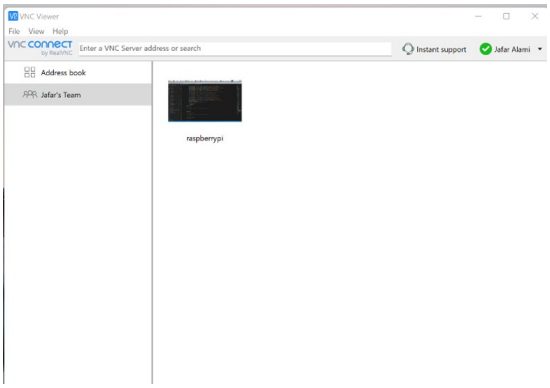
RealVNC.

9. Siden vi har RealVNC konto, så vi logger direkte inn. For deg som ikke har det, så trykk på der jeg viste på bildet nede i Figur 158.



Figur 158: Innloging side av RealVNC viewer.

10. Når det er logget inn, så åpnes en nye side hvor du limer IP adressen som vi viste tidligere i oppsett konfigurasjonen. Det å kopiere IP adressen og limes videre i søkelinje i VNC. Neste gang du vil bruke VNC, så alt er klar og det er bare trykke på liten skjerm som er vist i Figur 159



Figur 159: VNC viewer.

F Oppsett av remoteit

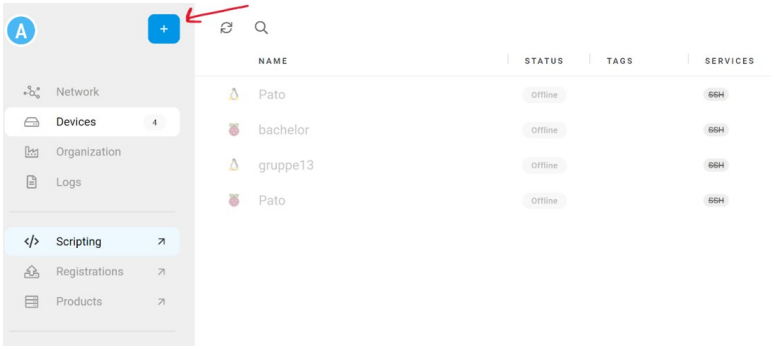
I likhet med VNC, det finnes to metoder å styre raspberry pi med remoteit.

- Første metoden er at raspberry pi og datamaskinen du vil styre raspberry pi med være tilkoblet til samme nettet.
- Andre metoden er at du kan jobbe med raspberry pi-en over internett uten at du må være tilkoblet til samme nettet som Pi-en.

Videre viser vi oppsettet av remoteit over internett.

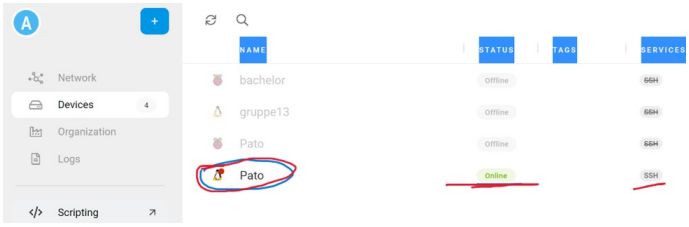
1. Først, vi trenger å opprettes en konto hos remoteit. `https://www.remote.it/`
2. Neste, vi trenger å installere remoteit pakker i OS. Den gjør vi ved hjelp av kommando-en
   Sudo apt-get install remoteit
3. sudo apt install connectd
4. sudo connectd_installer
   - Nå får vi flere alternativ å velge mellom. Vi å velge nummer 1.
   - Skrive inn brukernavn eller E-mail.
   - Skriv inn passordet ditt.
   - Velg alternativ 1 igjen for å kunne få tilgang fra hvor som helst.
   - Velg 1 igjen for SSH on port 22.
   - Velg Y
   - velg et navn for server_ssh

5. Nå, alt som er installert, vi må logge inn i remoteit med brukenavn og passordet. Åpne remoteit konto-en og velg devices eller enheter som vises i Figur 160.



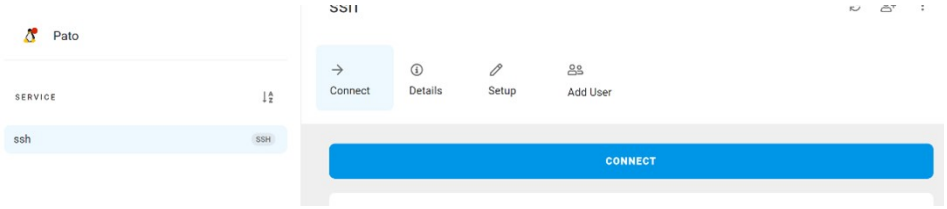
Figur 160: Velge av enheter.

6. For å koble til raspberry pi, vi går til devices eller enheter. Etter at vi trykket på enheter, så vil dukke opp en ny side som vi ser på bildet nedenfor. Hvis pi-en er tilkoblet til nettet, det vil vise grønn farge og viser at enheten er pålogget vist i Figur 161.



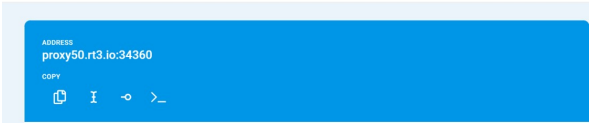
Figur 161: Sjekking av enheter.

7. Da går vi til enheten som er pålogget, trykker vi på den, det vil åpnes nye side. Der går vi til SSH og trykker på connect igjen vist i Figur 162.



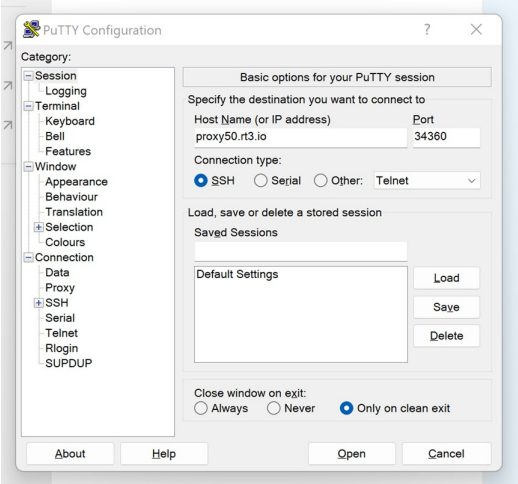
Figur 162: tilkobling av SSH.

8. Når vi fått koden som på bildet nedenfor, kopierer vi den og åpner Putty vist i Figur 163.



Figur 163: SSH kode.

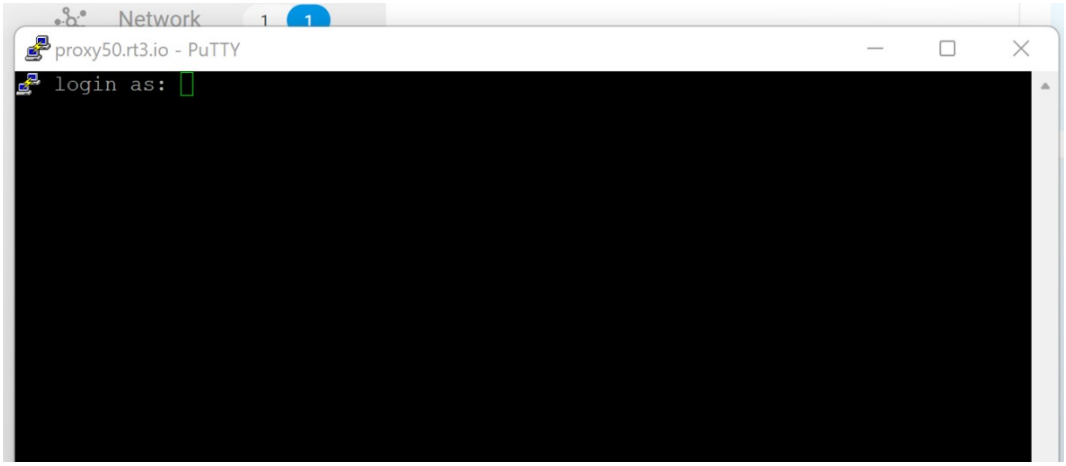
9. Limer inn den kopierte koden og trykker på åpne. Da vil åpne ny side og der vil vi godta for å koble til raspberry pi vist i Figur 164.



Figur 164: Putty.

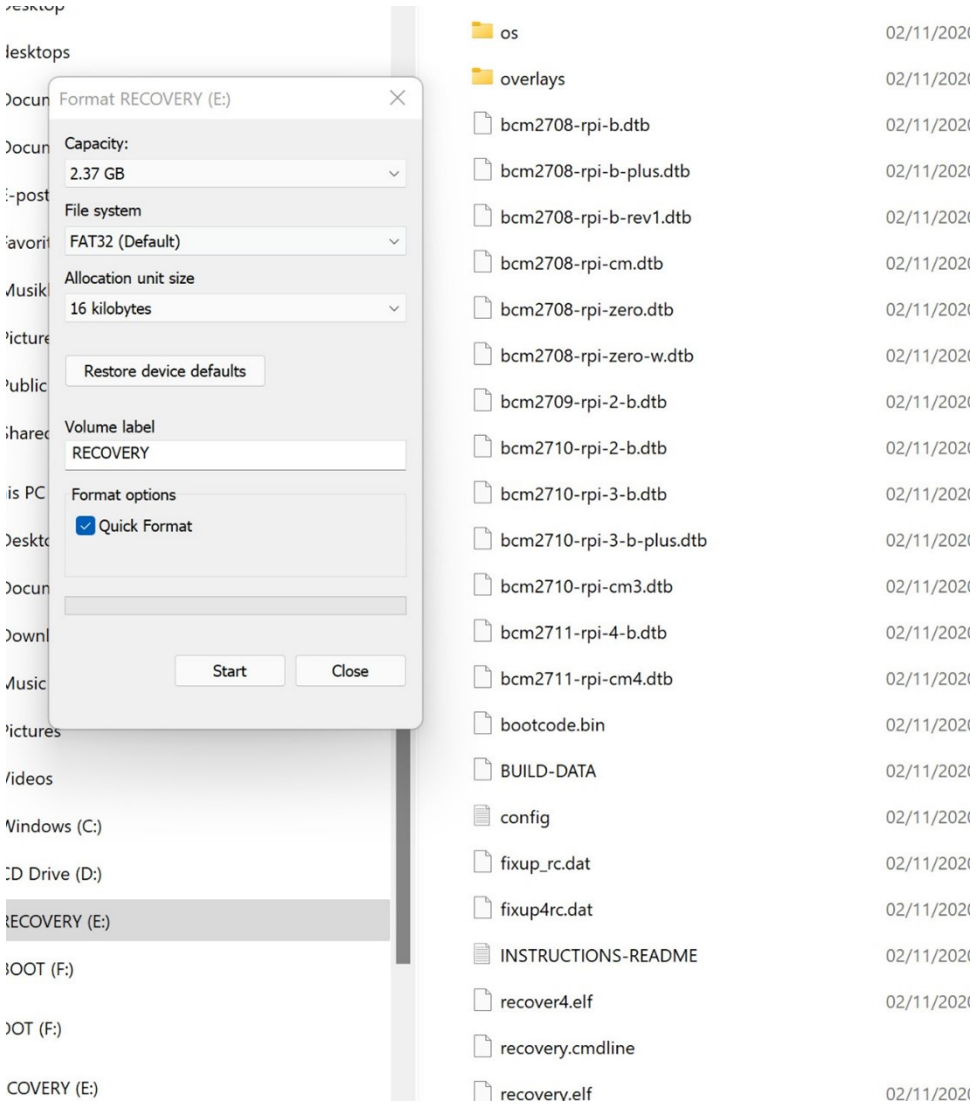
10. Etter godkjenning, får vi tilgang til raspberry pi terminal. Det må oppgis brukenavn og passordet. Remoteit er litt annerledes enn VNC, forskjellen er at i remoteit terminal

vi må gjøre alt gjennom terminal, men VNC fungerer som at vi jobber i selve raspberry pi vist i Figur 165.



Figur 165: Raspberry pi terminal.

For å oppgradere OS-en fra 32 bit til 64 bit, vi trenger å slette innholdet av gamle SD kortet som 32 bit var installert. Først, vi må putte SD mikrokortet i kortleseren og da vil vi få en liten skjerm som vises nedenfor i Figur 166.



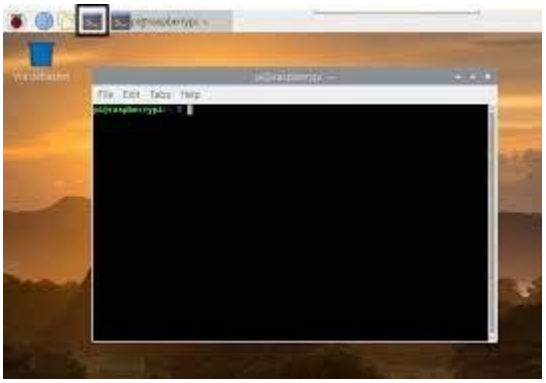
Figur 166: Format av SD kortet.

Når vi har fått opp denne siden, neste vi må gjøre er kryss av på Quick format også start, da sletter det alt data-en du har inn i mikrokortet. En ting må gjøres før du sletter, sikre deg at du ikke trenger dataene du har inne i mikrokortet.

G Oppsettet og kjøring av kode

Når du har fått en ny pi, det første som er viktig er å laste ned en 64-bit os. For å få det gjort har vi skrevet et oppsett (oppsett 64bit)på hvordan man få det til. Med vår 64-bit os pi, trenger vi å laste ned noen dependencies. Åpne opp terminal som vises i Figur 167 og 168:

Figur 167: Raspberry pi terminal symbol.

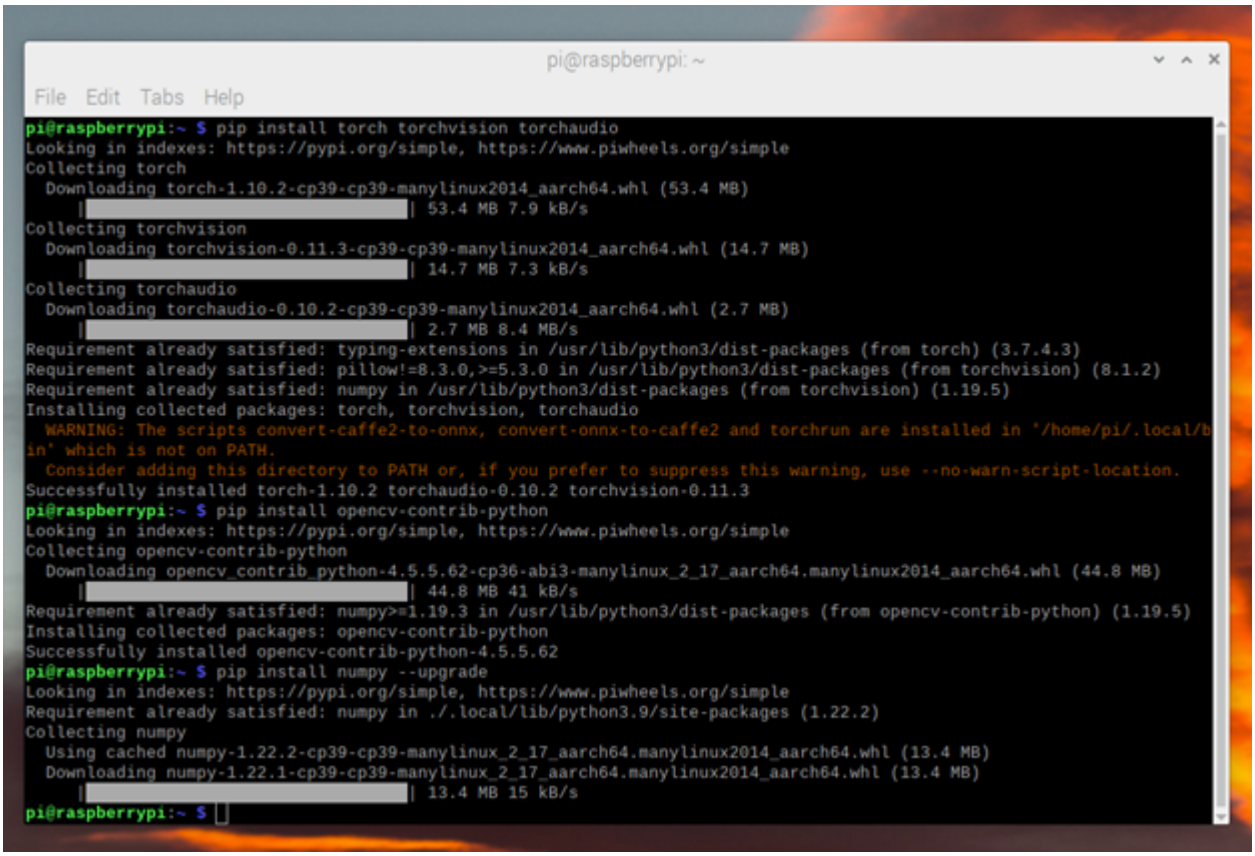


Figur 168: Raspberry pi terminal.

Kopier og lime inn følgenede kommandoer inn på terminal og trykk på enter:

- pip install torch torchvision torchaudio
- pip install opencv-contrib-python
- pip install numpy –upgrade

For å vite forventende output se på Figur 169.



Figur 169: Nedlastingen av Pytorch, Opencv og numpy.

Når den er klar du laste ned koden fra driven til raspberry pi-en:

En detaljert video vises på oppsett av Pi og kode.docx

Vi liker å bruke visual studio code for å kunne åpne folder og se på koden, for laste ned visual studio må du skrive kommandoen til terminal:

sudo apt install code

Du skal finne mappen der du har lastet ned koden, her er en metode å gjøre det på. Først er det å gå inn i filplassering, og ekstrakt det til sitt eget folder. Vi skal da finne det og åpne det i visual studio code:

En detaljert video vises på oppsett av Pi og kode.docx

Nå må vi laste ned alle dependencies:

En detaljert video vises på oppsett av Pi og kode.docx

Etter vi har skrevet den kommandoen:
pip install -r requirements.txt

Skal vi trykke på enter. Her skal alle bibliotekene som trenges lastet ned. Hvis vi får en feilmelding som sier at vi mangler et modul, skal vi kopiere det modulet og skrive det på terminalen slik:

pip install "module".

Et annet problem som kan skje er at opencv sitter fast med nedlastingen. Hvis det går over 15 min uten å bli ferdig med nedlastingen av Building wheel for opencv-python (PEP 517). Du må trykke ctrl + c for å kansellere kjøringen. Løsningen er å oppgradere pip ved: pip install –upgrade pip Da laster vi opencv direkte med: pip install opencv-python==4.5.3.56

Etter den har blitt lastet ned skal vi kjører requirements på nytt ved å skrive: pip install -r requirements.txt
Etter vi har lastet ned alle de riktige bibliotekene, er vi endelig klar for å kjøre programmet. Da skal vi åpne to terminaler for å kjøre vår SenseCam.py og track.py:
En detaljert video vises på oppsett av Pi og kode2.docx