Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitenskap

Institutt for datateknologi, elektroteknologi og realfag

Detektering og varsling av hjortevilt ved veibane ved bruk av dyp læring

Detection and warning of game in vicinity of roads with deep learning

Systemdokumentasjon

Versjon <3.0>

.

REVISJONSHISTORIE

Dato	Versjon	Beskrivelse	Forfatter
29/04-22	1.0	Prosjektstruktur	Joachim Leiros
30/04-22	1.0	Klassediagram	Joachim Leiros
1/05-22	1.0	Servertjenester	Joachim Leiros
1/05-22	1.0	Installasjon og kjøring	Joachim Leiros
2/05-22	1.0	Dokumentasjon	Joachim Leiros
3/05-22	1.0	Testing	Joachim Leiros
3/05-22	1.0	Innledning	Joachim Leiros
8/05-22	1.0	Sekvensdiagram	Joachim Leiros
8/05-22	2.0	Klassediagram	Joachim Leiros
8/05-22	2.0	Arkitektur	Joachim Leiros
8/05-22	2.0	Dokumentasjon	Joachim Leiros
9/05-22	2.0	Servertjenester	Joachim Leiros
10/05-22	2.0	Prosjektstruktur	Joachim Leiros
12/05-22	3.0	Servertjenester	Joachim Leiros
12/05-22	3.0	Sekvensdiagram	Joachim Leiros
12/05-22	3.0	Klassediagram	Joachim Leiros
12/05-22	3.0	Arkitektur	Joachim Leiros

INNHOLDSFORTEGNELSE

1. INNLEDNING

Dette dokumentet er skrevet i forbindelse med et bachelorprosjekt hvor det skulle utvikles en dyp maskin læringsmodell med fokus på detektering av hjortevilt, samt et prototype program som tok inn inndata i form av video, prosesserte inndata på modellen og varslet dersom en deteksjon forekom.

Hensikten med dokumentet er å gi leser forståelse i hvordan programmet *Realtime-DWLS* er bygd opp, hvordan programmet fungerer og hvordan benytte seg av programmet.

Kildekoden finnes på https://github.com/jwlei/real-time-object-detection-YOLOv5-cv2

Kapittel **2 Arkitektur** gir en overordnet oversikt i hvilke systemer, komponenter, klienter og datakilder programmet består av. Det gis en grov oversikt over programmets flyt.

Kapittel **3 Prosjektstruktur** gir en oversikt over fil- og katalogstrukturen for prosjektet.

Kapittel **4 Klassediagram** gir en oversikt over klasser som inngår i programmet, hvilke variabler og funksjoner disse har og kobling mellom de forskjellige klassene.

Kapittel **5 Sekvensdiagram** viser programflyten for hoved use-caset *Deteksjon*.

Kapittel **6 Databasemodell** er utelatt da det ikke innehar relevans for prosjektet.

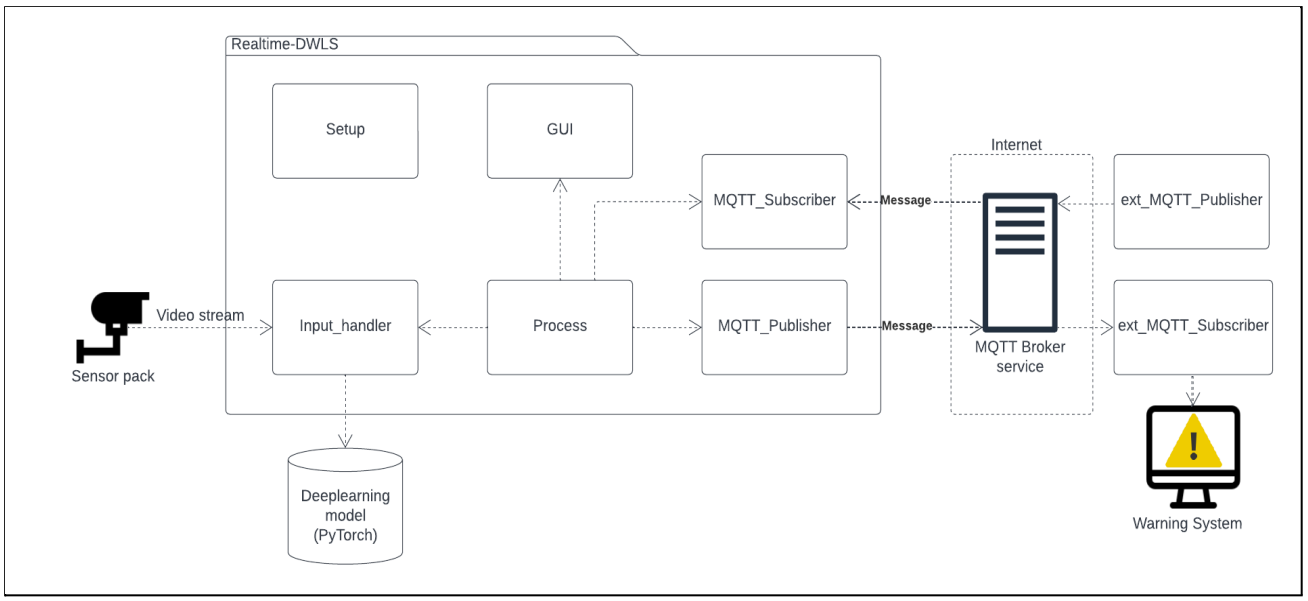
Kapittel **7 Server-tjenester** gir en oversikt og forklaring på hvilken server tjenester programmet benytter seg av for å kunne ta imot og sende informasjon. Det blir også nevnt hvilke tiltak som er iverksatt i forhold til sikkerhet.

Kapittel **8 Installasjon og kjøring** gir grundige instruksjoner i hvordan installere og kjøre programmet. Her finnes og en oversikt over nødvendige avhengigheter med en kort beskrivelse.

Kapittel **9 Dokumentasjon av kildekode** viser til hvor dokumentasjon på kildekoden finnes samt hvordan den leses og en kan generere ny oppdatert kildekode.

Kapittel **10 Kontinuerlig integrasjon og testing** er utelatt da det ikke innehar relevans for prosjektet.

2. ARKITEKTUR



Figur 2.1: Skisse av programmets arkitektur

På figur 2.1 sees en overordnet skisse over Realtime-DWLS programmet som viser arkitekturen samt flyten i programmet.

I sentrum av programmet finnes *Process* klassen som kaller på de andre klassene.

Prosessen starter en instans av *Input_handler*, *GUI*, MQ Telemetry Transport(*MQTT) Subscriber* og *Publisher*.

Input_handler får inn inndata fra en video strøm og kjører dataen gjennom PyTorch rammeverket som benytter seg av den trente YOLOv5 modellen. Prosessen leser så bildet og verdier gitt av Input_handler og sender så ut det nåværende bildet som har blitt predikert på til GUI.

I tillegg legges verdier fra prediksjonen, slik som status deteksjon, hvor høy konfidens, antallet deteksjoner og klokkeslett inn i en melding MQTT Publisheren sender ut som en tekststreng, formatert i JSON format.

Prosessen lytter og på innkommende meldinger via MQTT Subscriber og kan få en ny URL til en oppdatert modell som den vil laste ned og sette som ny standard modell.

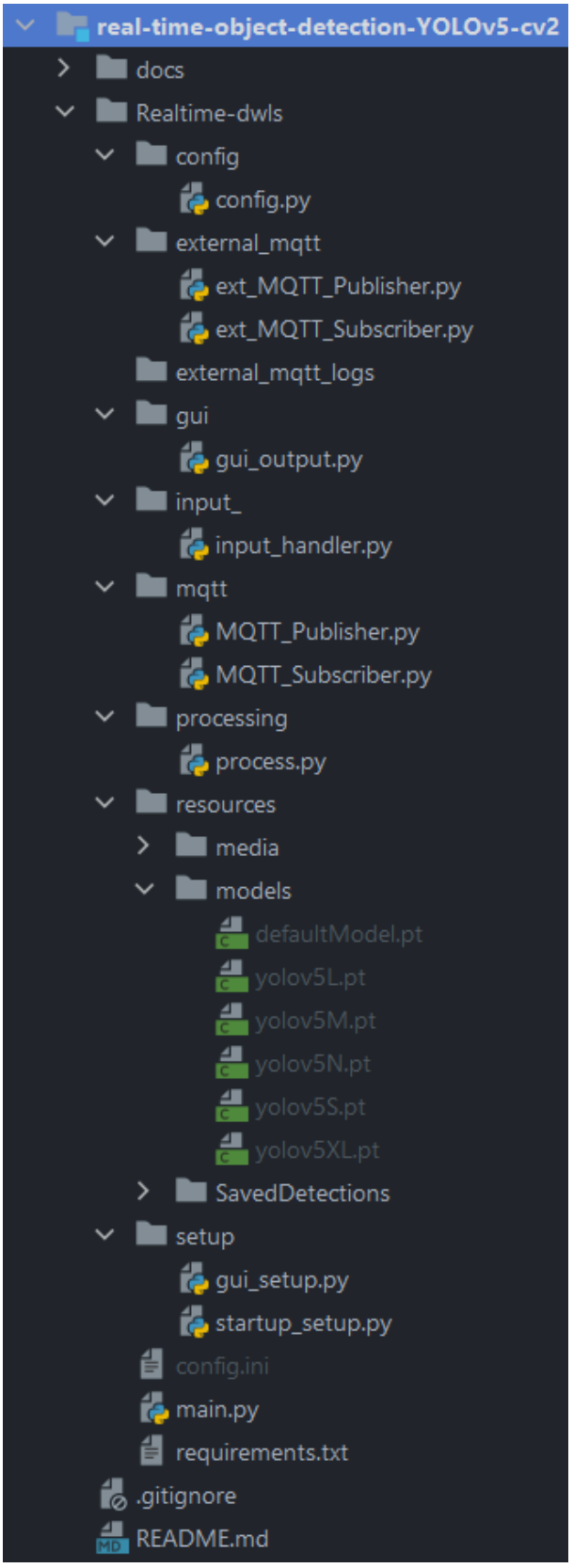
3. PROSJEKTSTRUKTUR

I figur 3.1 til høyre sees prosjektstrukturen i IDE verktøyet IntelliJ. Strukturen viser hvordan prosjektet, utviklet i Python 3.10 er inndelt i forskjellige mapper og moduler.

I prosjektmappen finnes dokumentasjon som forklarer kildekoden til programmet samt en README.md fil.

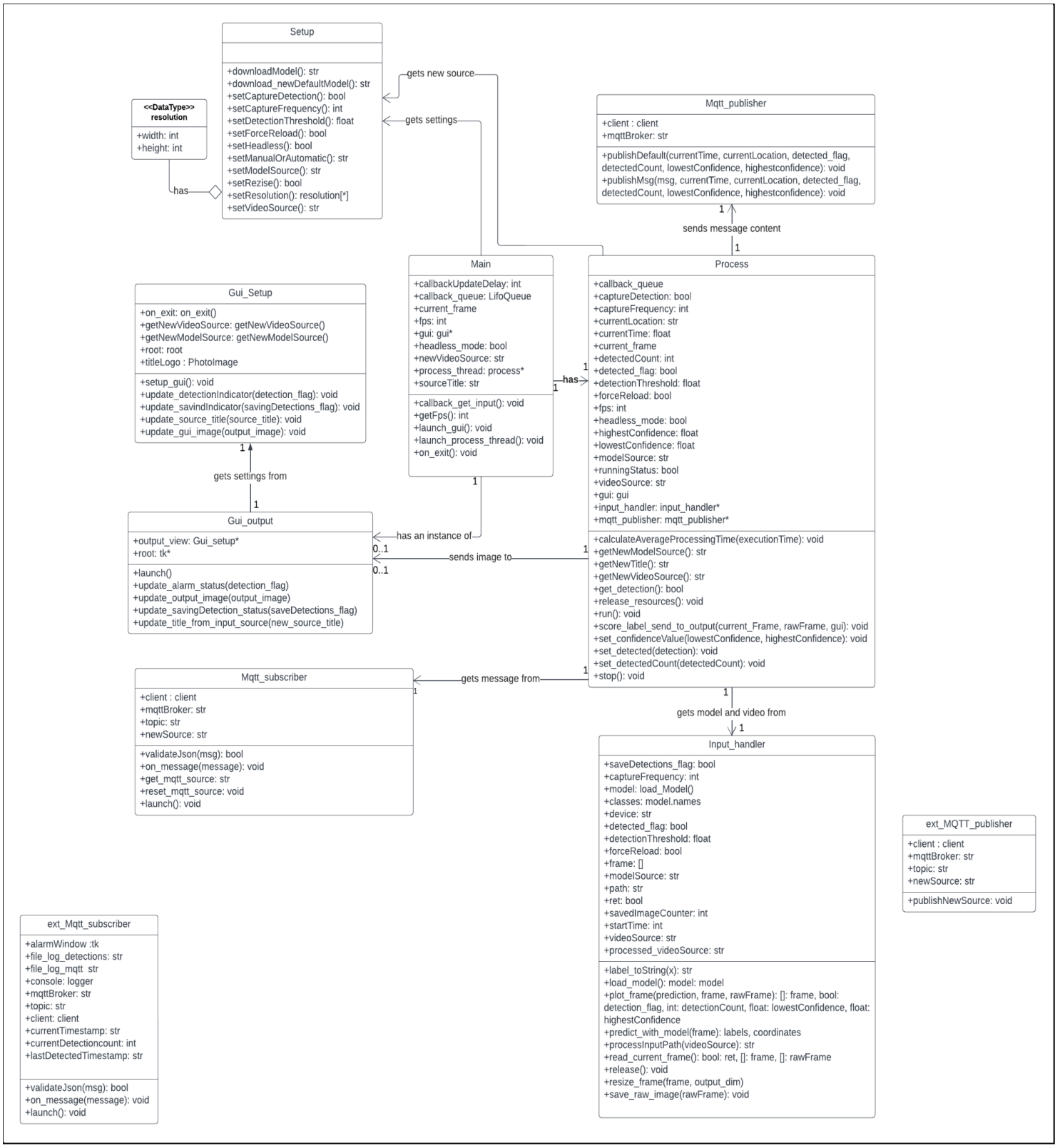
I programmets mappe */Realtime-dwls/* finnes de forskjellige modulene og ressursene til prosjektet samt en konfigurasjonsfil(config.ini) og requirements.txt som inneholder en liste over avhengigheter som må installeres før bruk.

I mappen /resources/ finnes grafiske elementer brukt til presentasjon i det grafiske grensesnittet, en mappe dedikert til nedlastede modeller samt en mappe for bilder som blir lagret ved deteksjoner.

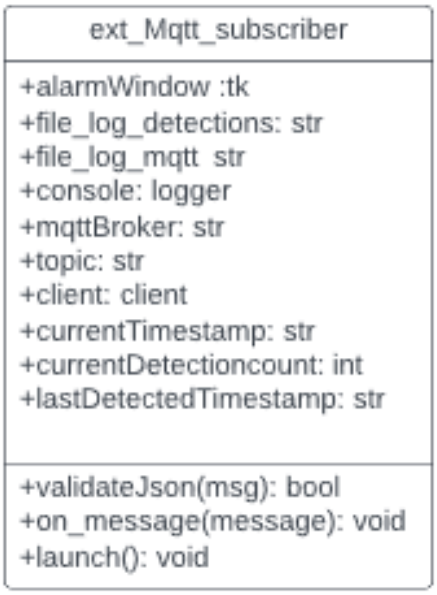


Figur 3.1: Prosjektets struktur

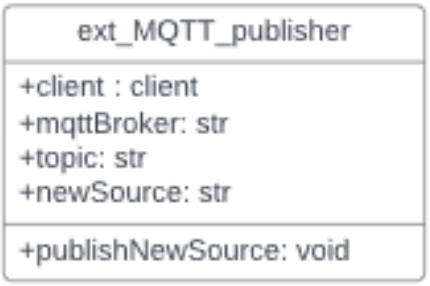
4. KLASSEDIAGRAM



Figur 4.1: Klassediagram av Realtime-DWLS

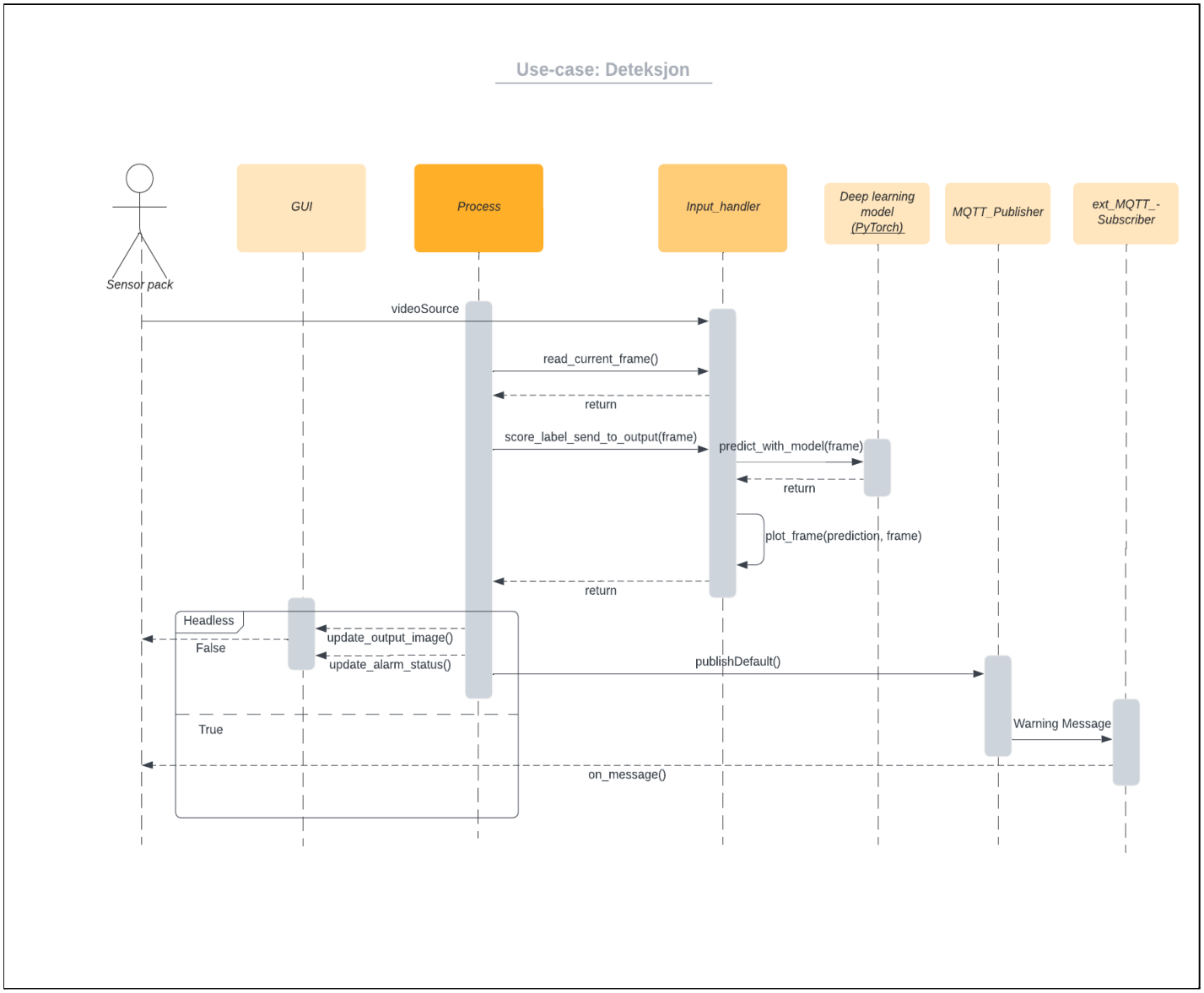


Figur 4.2: Klassediagram av den eksterne MQTT Publisher klienten



Figur 4.3: Klassediagram av den eksterne MQTT Subscriber klienten

5. SEKVENSDIAGRAM



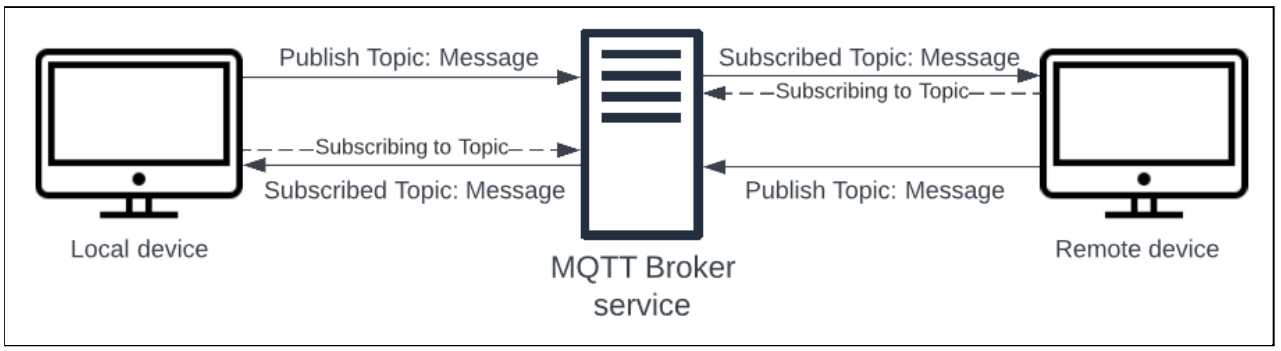
Figur 5.1: Sekvensdiagram for use-caset "Deteksjon"

I figur 5.1 sees programflyten for use-caset deteksjon. Input_handler får inn en videostrøm fra sensorsystemet. Process henter et enkelt bilde fra videostrømmen og kaller på funksjoner for å utføre prediksjon og sende resultatet av prediksjonen til varslingssystemene i GUI og hos ext_MQTT_Subscriber.

6. DATABASEMODELL

Dette kapittelet er utelatt da det ikke har relevans for prosjektet.

7. SERVER-TJENESTER



Figur 7.1: Illustrasjon av MQTT-tjeneste mellom to enheter gjennom en broker.

Meldingstjenesten MQTT fungerer som et knutepunkt hvor *publisher* klienter sender en melding til broker service tilknyttet et emne. *Subscriber* klienter sender beskjed til broker-service om at de abonnerer på et emne. Dersom broker-service får inn en melding tilhørende emne en subscriber abonnerer på, blir denne levert til subscriber neste gang subscriber ber om meldinger tilknyttet emnet.

Gjennom meldingstjenesten tilbys varsling inn- og ut og fjern-oppdatering av modell. Funksjonene i meldingstjenesten er adskilt ved forskjellige *emner*. For funksjonalitet relatert til varsling benyttes emnet *DWLS_DETECTION,* for fjern-oppdatering benyttes emnet *DWLS_REFRESH_MODEL*. Dette sørger for at informasjon holdes adskilt og kun blir sendt eller mottatt der det er behov.

7.1 SIKKERHET

I programmet blir det benyttet en ukryptert Transmission control protocol(TCP) til en offentlig test MQTT broker, *mqtt.eclipseprojects.io*. Data som blir sendt via MQTT klientene i prosjektet bør derfor ikke inneholde sensitiv informasjon.

8. INSTALLASJON OG KJØRING

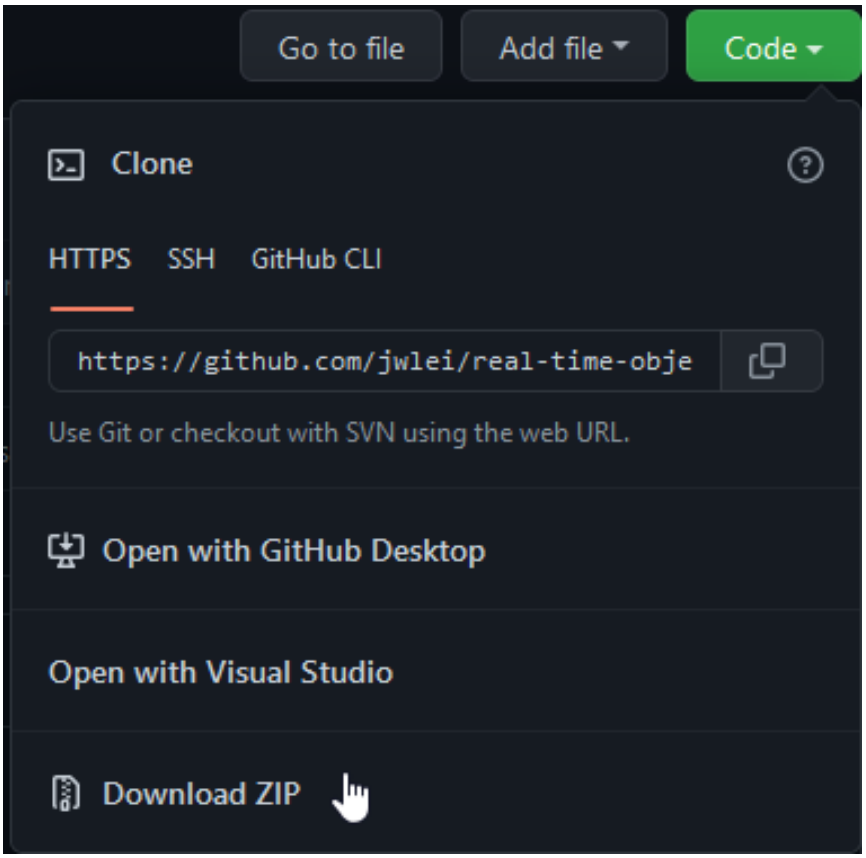
8.1 Installasjon

8.1.1 Hent programmet

For å installere programmet last ned eller *clone* prosjeket fra https://github.com/jwlei/real-time-object-detection-YOLOv5-cv2.

For å klone kjør kommandoen:

```
git clone https://github.com/jwlei/real-time-object-detection-YOLOv5-cv2
```



Figur 7.1: Hvordan hente prosjekt fra github.com

Hvis du velger å laste ned programmet som en **.zip** fil, pakk så ut prosjektet til ønsket sted.

8.1.2 Installer Python

Gå til nettsiden https://www.python.org/downloads/ og last ned Python versjon 3.10, følg installasjonsveilederen men pass på å *legg til Python til Path.*

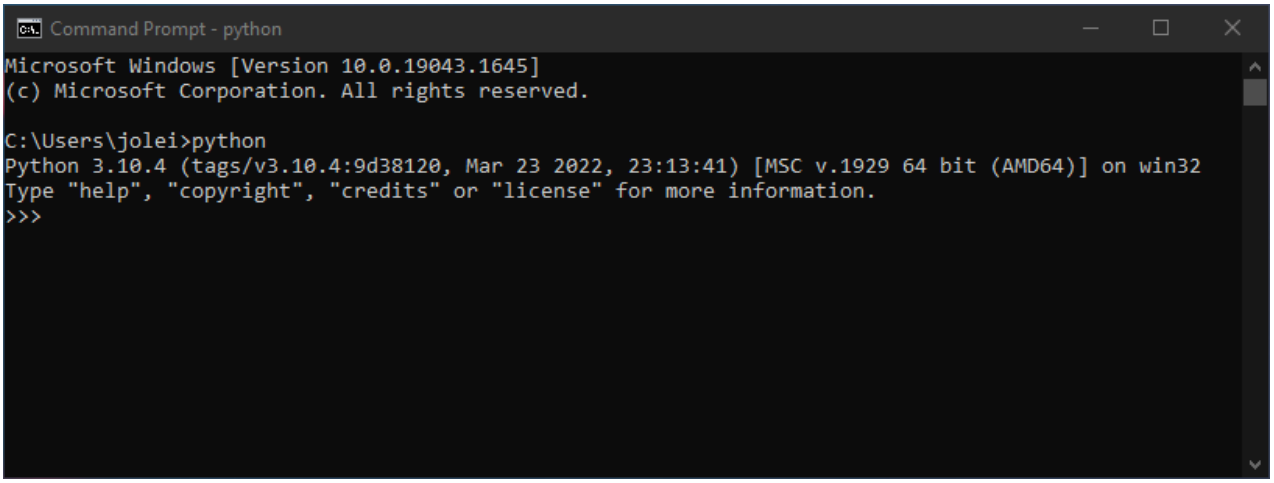
For å sjekke at installasjonen er fullført og Python er satt opp riktig på Path, trykk Windows knappen og åpne ledetekst/Command Prompt skriv så inn *python* og trykk på Enter-knappen.

```
python
```

Dersom du ikke får opp python versjonen din her må du sikre at Python er lagt til Path i Windows innstillingene, høyreklikk på *Min datamaskin* og åpne *egenskaper*, trykk så på Avanserte innstillinger etterfulgt av *miljøvariabler.*

I feltet *Systemvariabler* finn Path, klikk på *rediger*. Legg så inn stien til din Python installasjon og /Python/Script mappe. Du kan legge inn flere variabler ved å skille de ved bruk av ";".

Åpne så ledeteksten på nytt og gjenta steget over.



Figur 7.2: Python er korrekt installert og satt opp i Path som vist i ledetekst

8.1.3 Lag og aktiver et virtuelt miljø

Når Python er installert korrekt åpne en ledetekst-boks og pek den til mappen hvor prosjektet ligger med kommandoen:

```
cd sti/til/prosjekt/Realtime-dwls
```

Etabler så et virtuelt miljø for Python for å forhindre at pakker installert i prosjektet eller andre prosjekt påvirker hverandre. Miljøet lages ved å definere sti/til/miljø/navn_miljø med kommandoen:

```
python -m venv navn_virtuelt_miljø
```

Når miljøet er lagd aktiveres det ved å kjøre activate.bat filen som ligger i navn_virtuelt_miljø/Scripts/ med kommandoen:

```
navn_virtuelt_miljø\Scripts\activate.bat
```

Dersom miljøet er aktivert korrekt vil ledeteksten vise:

```
(navn_virtuelt_miljø) C:\sti\til\prosjekt\Realtime-dwls>
```

8.1.4 Installer avhengigheter

Når det virtuelle miljøet er aktivert, og det er klart for å installere pakker programmet avhengig av. De forskjellige pakkene er nærmere forklart i *kapittel 7.3 Avhengigheter*. Installer avhengighetene fra *requirements.txt* med kommandoen:

```
pip install -r requirements.txt
```

Når avhengighetene fra requirements.txt er installert gå så videre og installer *PyTorch*. Dersom enheten programmet skal kjøres på har en *CUDA-driver*, installer gjerne *PyTorch* med *CUDA* for økt ytelse.

PyTorch uten CUDA med kommandoen:

```
pip install torch torchvision
```

PyTorch med CUDA med kommandoen:

```
pip install torch torchvision --extra-index-url https://download.pytorch.org/whl/cu113
```

Nå er programmet ferdig installert og er klart til å kjøres.

Mange av avhengighetene til YOLOv5 har blitt utelatt da de ikke er nødvendig for drift av Realtime-DWLS programmet, men dersom det skulle være feil med avhengigheter i forbindelse med YOLOv5 kjør kommandoen under for å installere alle avhengigheter forbundet med YOLOv5:

```
pip install -qr https://raw.githubusercontent.com/ultralytics/yolov5/master/requirements.txt
```

8.2 Kjøring og bruk

8.2.1 Konfigurasjon

Første gang programmet blir kjørt vil en konfigureringsfil genereres i programmets mappe (sti/til/prosjekt/Realtime-dwls/) kalt *config.ini*. Denne filen kan åpnes og redigeres i ett tekst-verktøy som *Notepad*.

Ved oppstart tar programmet utgangspunkt i konfigurasjonsfilen, oppstartsinnstillinger kan endres ved å endre på verdiene i konfigurasjonsfilen.

8.2.2 Tilgjengelige innstillinger

Innstilling	Beskrivelse	Godtatte verdier
resizeoutput	Bestemmer om bildet i videostrømmen skal skaleres til ønskede dimensjoner	True/False
width	Bredde på bildet i videostrøm	Tall uten desimaler
height	Høyde på bildet i videostrøm	Tall uten desimaler
skipsetup	Bestemmer om programmet kjører direkte fra konfigurasjonsfil eller om bruker skal definere innstillinger under oppstart av program.	True/False
startmqttsubscriber	Bestemmer om en tilleggs prosess kalt MQTT Subscriber skal startes ved oppstart for å demonstrere fjern-varsling og loggføring.	True/False
videosource	Videostrømmen programmet starter med å lese fra, dette kan være: *URL til en videostrøm* *sti/til/lokal/videofil.mp4* *Ett tal*l som representerer et lokalt kamera på enheten, primært kamera er representert av tallet 0, dersom det er flere kameraer på enheten vil disse være representert av påfølgende tall 1, 2.. osv.	Tekststreng, tall

*Tabell 1: Tilgjengelige innstillinger **del 1**.*

Innstilling	Beskrivelse	Godtatte verdier
modelsource	En tekststreng som representerer stien til hvilken modell programmet skal bruke for å predikere på videostrømmen. *sti/til/modellen.pt* Dersom det blir sendt en ny modell fra skyen vil den bli lagret som: *resources/models/defaultModel.pt*	Tekststreng
remotemodelurl	En URL som til modell fra skyen som blir lastet ned dersom det ikke finnes noen modell under *resources/models/defaultModel.pt.* Sørger for at programmet alltid har en modell.	Tekststreng
capturedetection	Bestemmer om det skal lagres rå-bilder når det blir gjort en detektering. Bilder blir lagret under: *resources/SavedDetections/* Bilder som lagres kan brukes for videre trening av modell.	True/False
capturefrequency	Dersom det skal lagres bilder ved detektering, angir frekvens på hvor ofte de lagres oppgitt i sekunder. Minimum 1 sekund, maksimum 59 sekunder.	Tall uten desimaler
forcereload	Bestemmer om PyTorch skal oppdatere seg i hurtigminne. Dersom enheten allerede har PyTorch i hurtigminne kan denne innstillingen skrus av for raskere oppstart.	True/False
detectionthreshold	Bestemmer hvor sikker modellen skal være før det blir regnet som en deteksjon. Fra: 0.00 tilsvarende 0% sikker. Til: 1.00 tilsvarende 100% sikker.	Tall med desimal Fra 0.0 til 1.0
headless	Bestemmer om programmet skal starte opp uten grafisk brukergrensesnitt for å spare ressurser.	True/False

*Tabell 2: Tilgjengelige innstillinger **del 2**.*

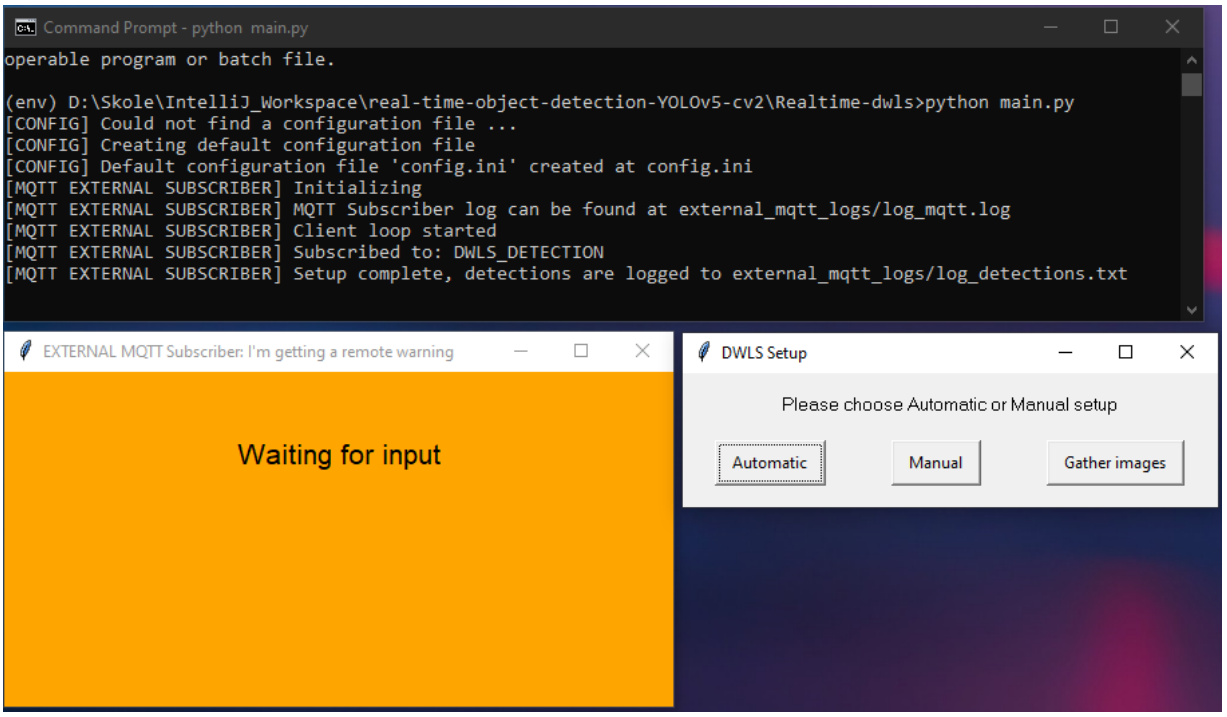
8.2.3 Starte programmet

For instruksene for å kjøre programmet forutsettes det at ledeteksten er åpnet og pekt til programmets mappe og det virtuelle miljøet er aktivert, detaljert i 7.1.3.

Hovedprogrammet kjøres ved å la Python åpne *main.py* filen ved følgende kommando:

```
python main.py
```

Ved oppstart vil Realtime-DWLS programmet se etter en modell under *resources/models/defaultModel.pt*. Dersom den ikke finnes vil en YOLOv5-Medium versjon lastes ned.



Figur 8.3: Skjerm ved første oppstart

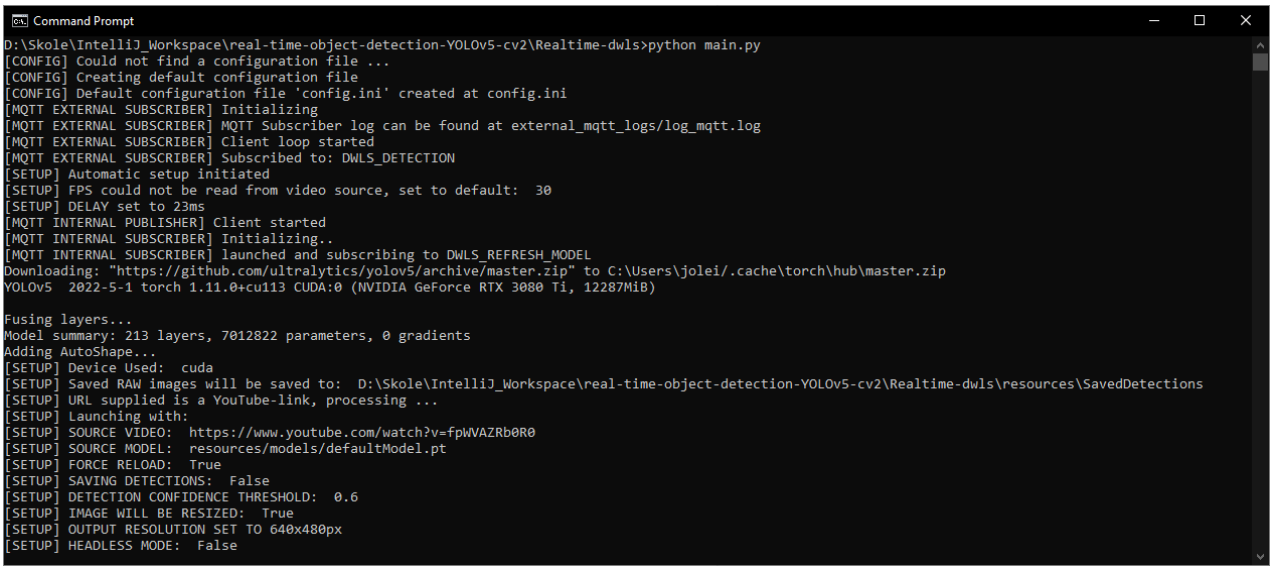
Figur 8.3 viser skjermbildet ved første oppstart, oppe er ledeteksten som viser hvilke steg programmet har utført eller utfører. Nede til venstre er en ekstern MQTT klient som brukes for å demonstrere at en deteksjon blir sendt ut til en annen enhet som kan loggføre eller sette i gang varsling. Nede til høyre blir bruker gitt et oppstartsvalg.

Ved oppstart av programmet kan det velges automatisk oppstart(Automatic), manuelle innstillinger(Manual) eller automatisk oppstart med fokus på innhenting av bilder(Gather Images).

Når type oppstart er valgt vil programmet starte automatisk(*Automatic*) eller etter bruker har definert innstillinger(*Manual*).

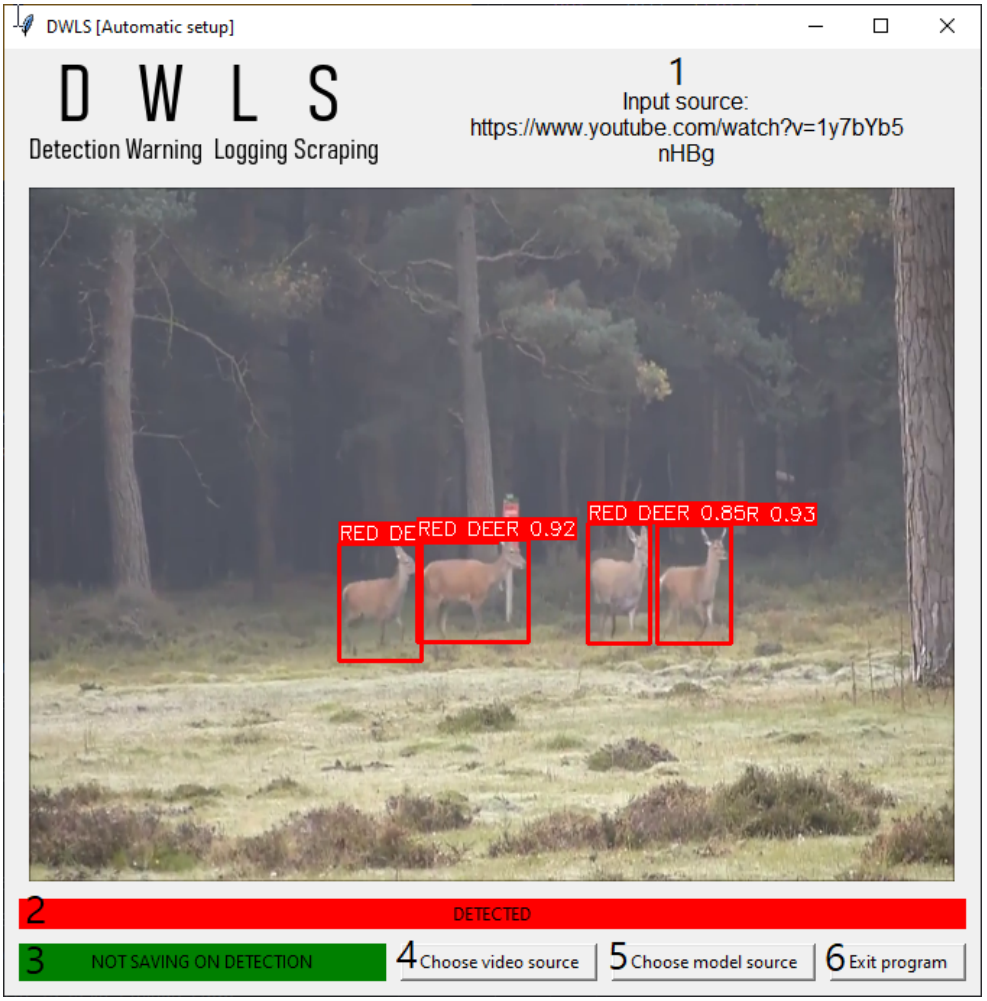
Ved første kjøring anbefales det å velge **automatisk** oppstart. Dersom manuell oppstart blir valgt blir bruker spurt om å svare på innstillinger tilgjengelig i konfigurasjonsfilen og står fritt til å velge hvilken som helst kombinasjon.

Når automatisk oppstart er valgt eller bruker er ferdig med å definere innstillinger starter programmet og ledeteksten vil vise hvilke innstillinger og verdier som blir brukt som illustrert i figur 8.4.



Figur 8.4: Ledetekst ved første oppstart som viser innstillinger og status.

8.2.4 Grafisk grensesnitt



Figur 8.5: Grafisk brukergrensesnitt for program

Beskrivelse av grafisk grensesnitt	
Tall:	**Funksjon**
1	Viser hvilken kilde inndata kommer fra.
2	Indikerer om det er en deteksjon.
3	Indikerer om bilder blir lagret ved deteksjon.
4	Knapp for å velge ny kilde for videostrøm.
5	Knapp for å velge ny kilde for trent modell.
6	Knapp for å avslutte programmet.

Tabell 3: Forklaring av grafisk grensesnitt.

8.2.5 Tilleggsprogram

Vedlagt prosjektet i mappen *sti/til/prosjekt/Realtime-dwls/external_mqtt/* ligger to programmer som representerer en ekstern (ikke tilhørende hovedprogrammet) MQTT Publisher(*ext_mqtt_publisher.py)* og Subscriber(*ext_mqtt_subscriber.py)*.

- Realtime-dwls/external_mqtt/
    - ext_mqtt_publisher.py
    - ext_mqtt_subscriber.py

Hensikten og funksjonen til disse programmene er å simulere og demonstrere kommunikasjon fra hovedprogram til andre enheter og hvordan disse kan interagere.

Program: ext_mqtt_subscriber.py

Funksjon*:* Får meldinger fra hovedprogram og loggfører disse. Dersom det er deteksjon så viser programmet et varsel gjennom et enkelt grafisk brukergrensesnitt.

Program: ext_mqtt_publisher.py

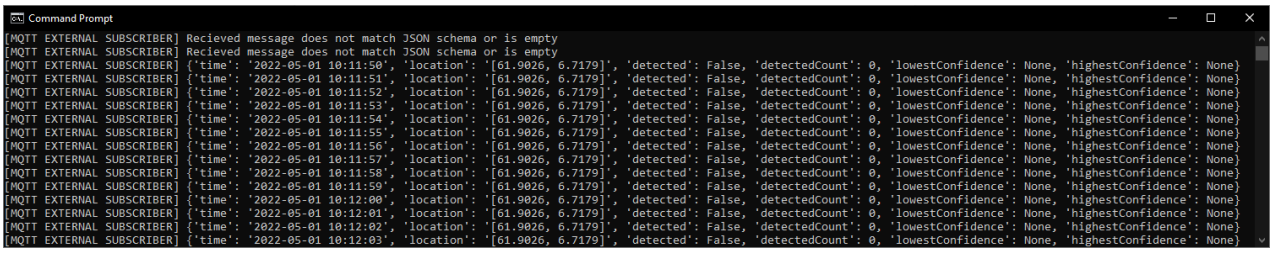
Funksjon*:* Sender en URL til en ny modell til hovedprogrammet, som hovedprogrammet da laster ned og setter som ny hovedmodell.

8.2.6 Ekstern MQTT Subscriber

Ved standardkonfigurasjon vil Realtime-dwls programmet starte den eksterne MQTT Subscriberen automatisk.

For å kjøre den manuelt åpnes et ledetekstvindu som vist i kapittel 7.1.3 etterfulgt av kommandoen:

```
python external_mqtt/ext_mqtt_subscriber.py
```



Figur 8.6: Ledetekst som viser meldinger den eksterne MQTT Subscriberen får fra Realtime-DWLS.

Figur 8.7: Grafisk grensesnitt som viser varsling mottatt fra Realtime-DWLS programmet.

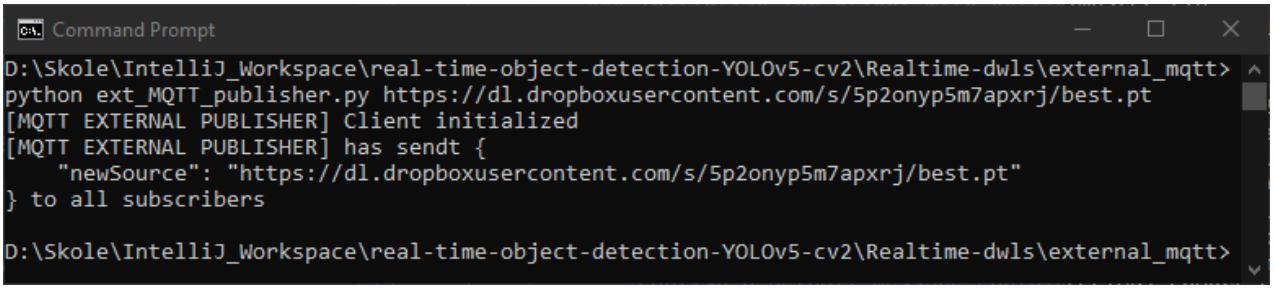
8.2.7 Ekstern MQTT Publisher

For å kjøre den eksterne MQTT Publisheren åpnes et ledetekstvindu som vist i kapittel 7.1.3.

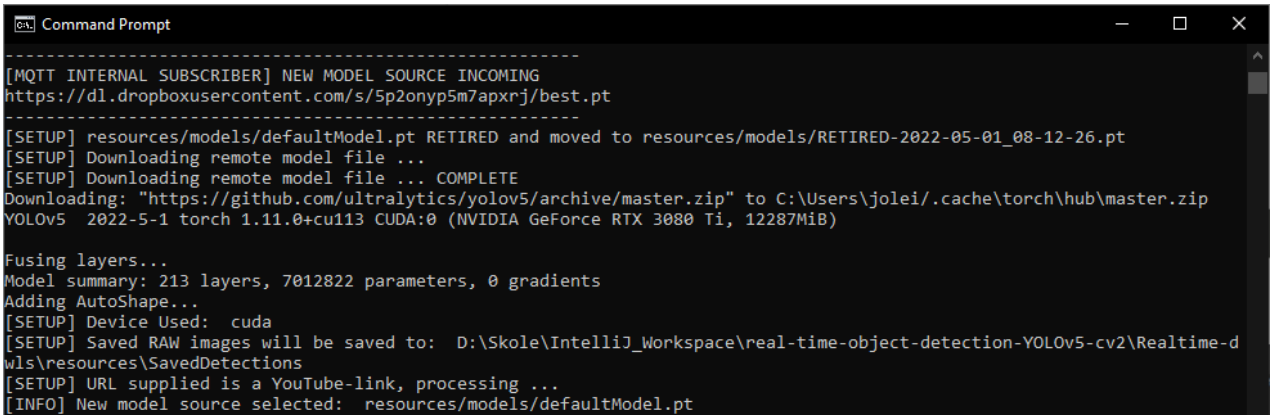
Programmet kjøres ved å kalle på filen *ext_mqtt_publisher.py* etterfulgt av en direkte URL til modellen som Realtime-dwls programmet skal laste ned og benytte som standard modell.

Etterfulgt av kommandoen:

```
python external_mqtt/ext_mqtt_publisher.py http://url_til.modell/modell.pt
```



Figur 8.8: Ledetekst som viser hvilken melding som blir sendt til Realtime-DWLS programmet.



Figur 8.9: Ledetekst for hovedprogrammet som viser at modellen er motatt fra den eksterne MQTT publisheren.

8.3 Avhengigheter

Avhengighet: PyTorch med eller uten CUDA(cu113)
torch versjon 1.11.0+cu113
torchvision versjon 0.11.0+cu113
Beskrivelse: PyTorch maskinlæring rammeverk som benytter seg av et optimisert tensor bibliotek anvendt i dyp læring. (*PyTORCH documentation, u.å.)*

***Avhengighet*:** geocoder versjon 1.38.1 eller nyere.
Beskrivelse: Et bibliotek som gjør det lettere å konvertere IP-adresser til GPS koordinater. *(Simple, consistent, u.å.)*

Avhengighet: jsonschema versjon 4.4.0 eller nyere.
Beskrivelse: En implementering av JSON Schema for Python som implementerer et vokabular fra JSON Schema(*JSON Schema, u.å.)* som gjør det enklere å beskrive og validere data-formatet. *(Schema validation, u.å.)*

Avhengighet: pafy versjon 0.5.5 eller nyere.
Beskrivelse: Et rammeverk for å hente data fra en YouTube lenke.*(PAFY documentation, u.å.)*

Avhengighet: youtube-dl versjon 2020.12.2
Beskrivelse: En del av pafy som bidrar til å hente videoer.

Avhengighet: paho_mqtt versjon 1.6.1 eller nyere.
Beskrivelse: En IoT(Internet of Things) TCP/IP tilkoblings protocol som gjør det enkelt å implementere en- og to-veis kommunikasjon mellom flere enheter. *(Paho-Mqtt, u.å.)*

Avhengighet: Pillow versjon 9.1.0 eller nyere.
Beskrivelse: Et bibliotek som gjør det mulig for Python å prosessere bilder. *(Pillow, u.å.)*

Avhengighet: pymsgbox versjon 1.0.9 eller nyere.
Beskrivelse: Et bibliotek som gjør det enkelt å lage simple meldingsbokser, pymsgbox benytter seg av Pythons innebygde Tkinter GUI verktøy.*(PyMsgBox, n.d)*

Avhengighet: requests versjon 2.27.1 eller nyere.
Beskrivelse: Et bibliotek som gjør det enkelt å sende HTTP/1.1 spørringer.*(HTTP for humans, u.å.)*

Avhengighet: pandas versjon 1.1.4 eller nyere.
Beskrivelse: Bibliotek for verktøy brukt til å analysere eller manipulere data.*(Pandas documentation, u.å.)*

Avhengighet: PyYAMLversjon 5.3.1 eller nyere.
Beskrivelse: Et rammeverk for å håndtere YAML data serialiserings språk.*(Pyyaml, u.å.)*

Avhengighet: tqdm versjon 4.41 eller nyere.
Beskrivelse: Et rammeverk for å lage framdrifts indikatorer.*(Costa-Luis, u.å.)*

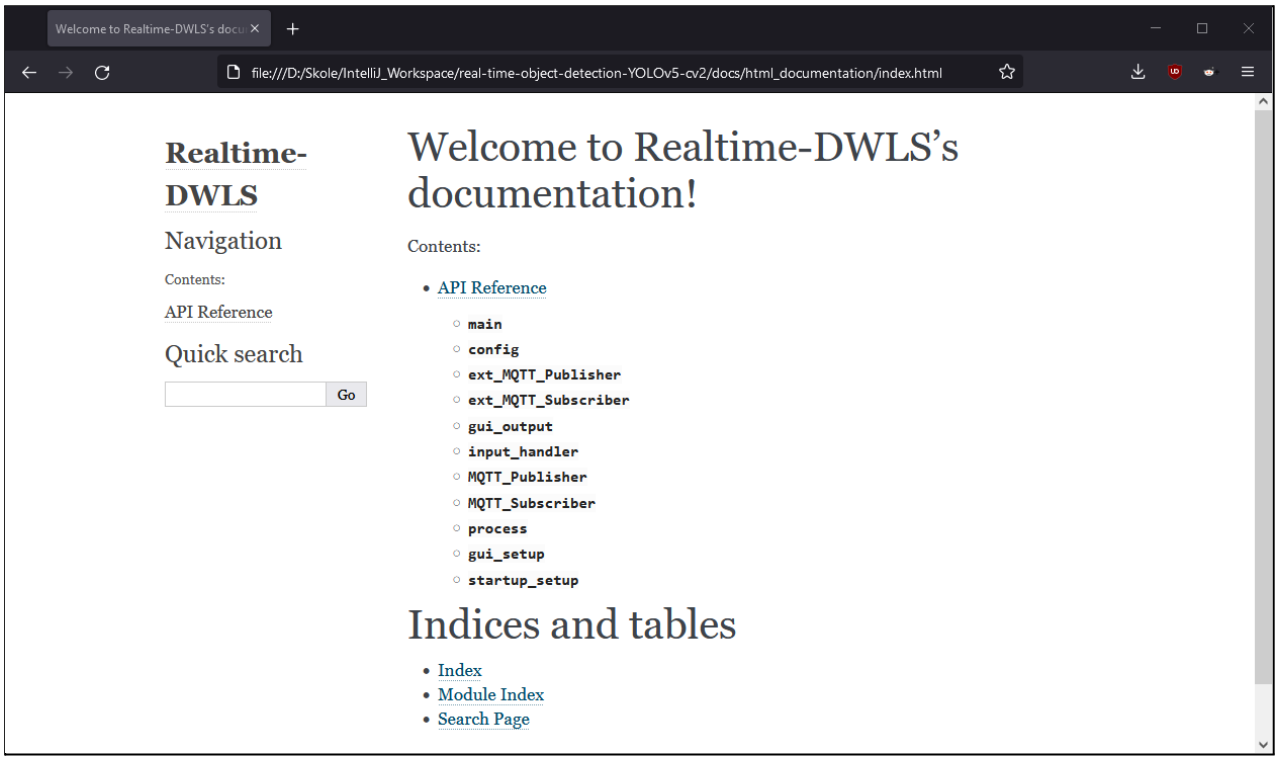
Avhengighet: matplotlib versjon 3.2.2 eller nyere.
Beskrivelse: Bibliotek for visualisering i Python.*(API reference, u.å.)*

Avhengighet: seaborn versjon 0.11.0 eller nyere.
Beskrivelse: Bibliotek for datavisualisering basert på matplotlib.*(Statistical Data Visualization, u.å.)*

9. DOKUMENTASJON AV KILDEKODE

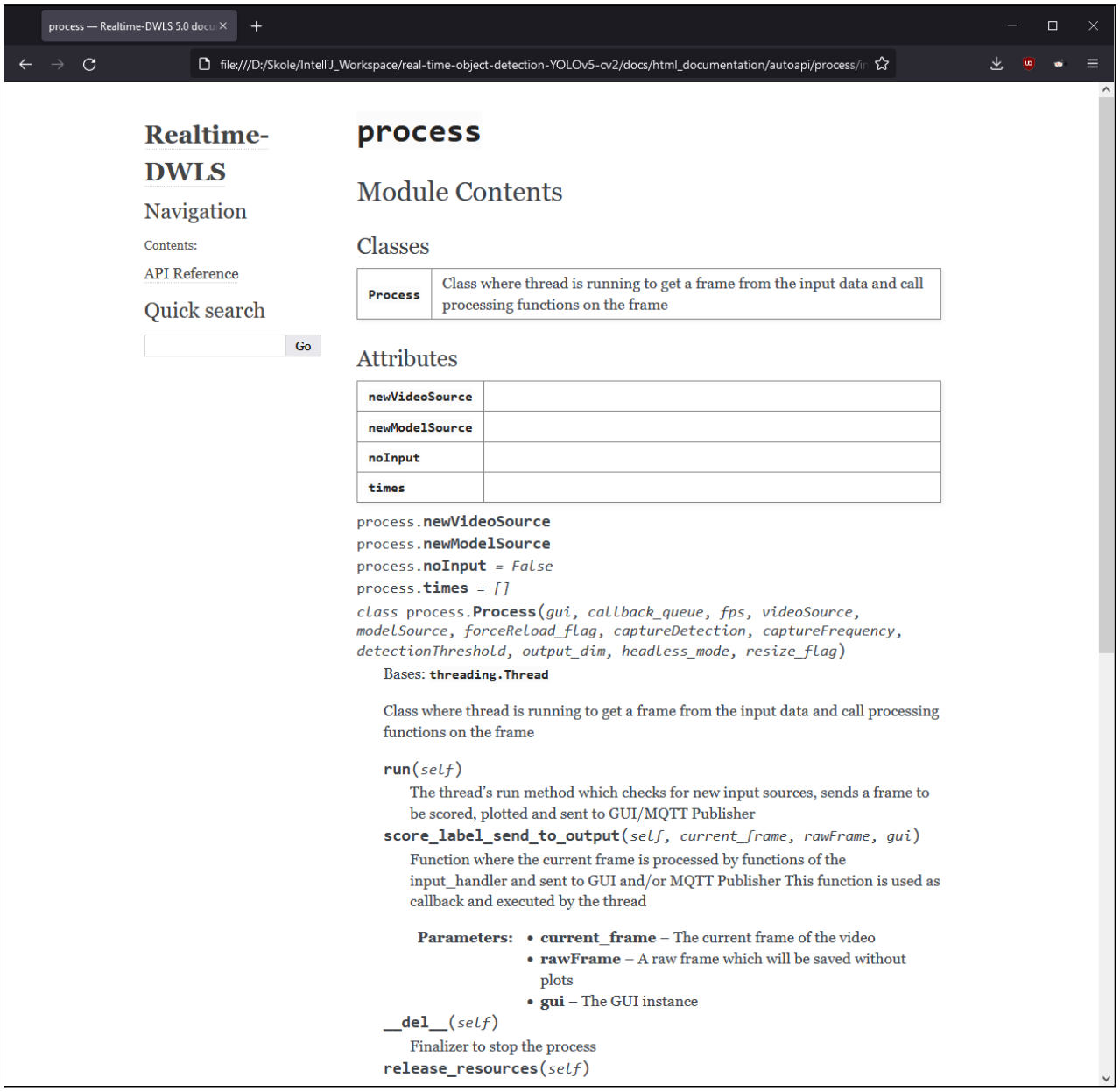
9.1 Hvordan lese dokumentasjon av kildekode

Vedlagt prosjektet under sti/til/prosjekt/docs ligger forhånds generert dokumentasjon av kildekode generert med Sphinx. (Sphinx documentation, u.å.) Denne åpnes ved å gå inn i mappen *prosjektMappe/docs/html_documentation/* og åpne filen index.html.



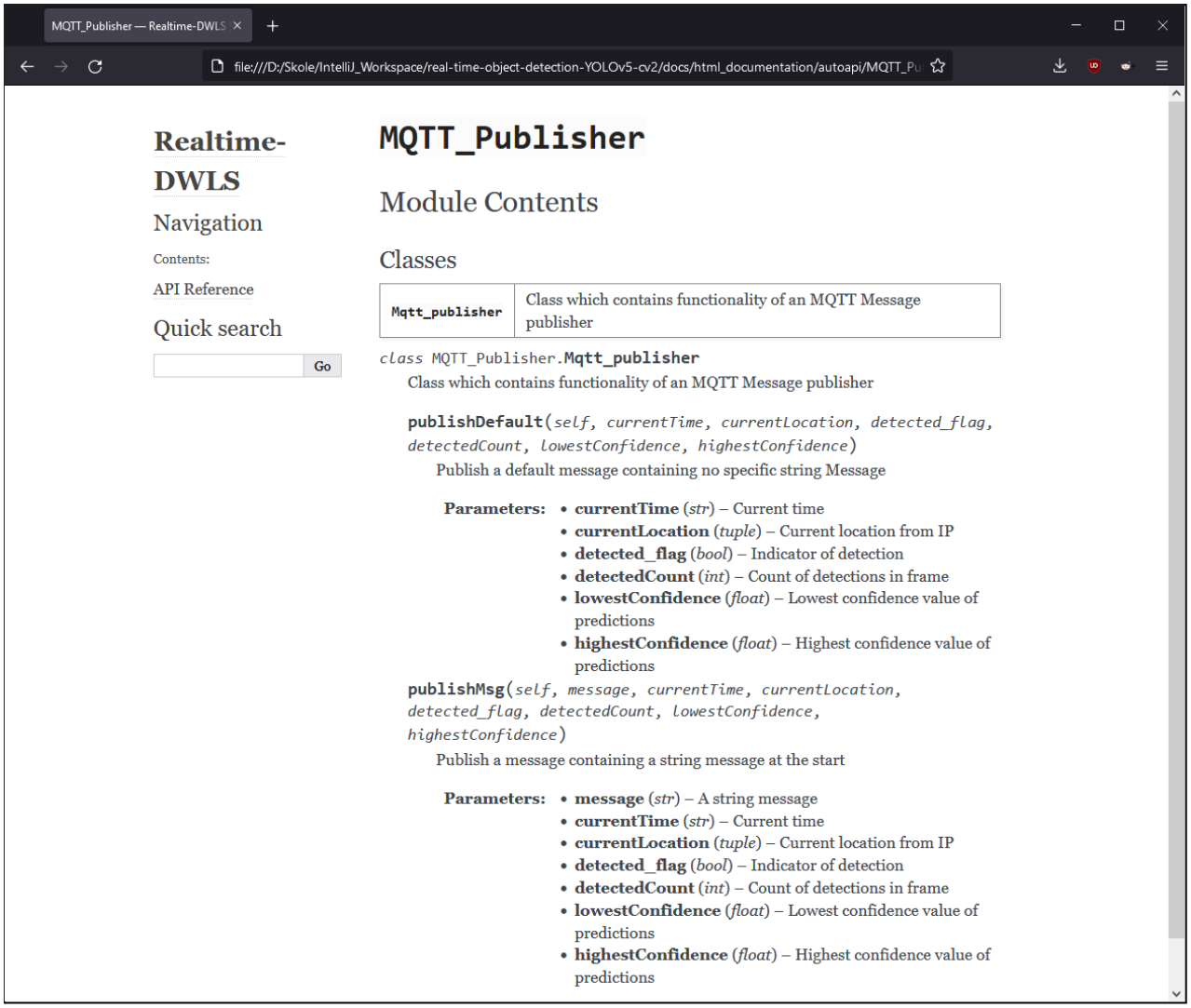
Figur 9.1: Hovedsiden til dokumentasjonen

Når index.html åpnes vises hovedsiden til dokumentasjonen som vist i figur 7.1, her velges det hvilken modul som ønskes lest.



Figur 9.2: Oversikt over modulen "process" som inneholder klassen Process.

På figur 9.2 vises en oversikt over modulen process, her vises det hvilke variabler klasser i modulen tar inn, hvilke funksjoner som finnes i klassen og hvilke parametre disse tar inn. Det gis en beskrivelse for parametre i hver enkelt metode.



Figur 9.3: Oversikt over modulen MQTT_Publisher

Likt som i forrige eksempel vises en oversikt over modulen MQTT_Publisher i figur 7.3. I figuren sees at modulen har en klasse Mqtt_publisher som har metodene:

- *publishDefault()*
- *publishMsg()*

For hver av metodene er det beskrevet hvilke variabler metodene tar inn, hvilken type variablene er og en kort beskrivelse av variabelen.

9.2 Generering av kildekode

For å generere kildekode må det installeres to avhengigheter, sphinx og sphinx-autoapi. Åpne det virtuelle miljøet som ble laget under installasjon av programmet og kjør kommandoene:

```
pip install sphinx
```

```
pip install sphinx-autoapi
```

Når avhengighetene er installert kan disse benyttes for å generere oppdatert kildekode.

For å starte generering av kildekode dokumentasjon, pass på at du er i samme virtuelle miljø, åpne ledetekst og pek til prosjektets mappe:

```
cd sti/til/prosjekt
```

Lag en mappe som heter docs:

```
mkdir docs
```

Pek til docs mappen:

```
cd docs
```

Kjør kommandoen sphinx-quickstart for å generere et oppsett for dokumentasjonen.

```
sphinx-quickstart
```

Når kommandoen kjøres vil du få noen spørsmål:

Spørsmål:	Svar:
Splitte source/build mappe	Nei
Prosjektnavn	Realtime-DWLS
Versjon	Versjonsnummer
Forfatter	Navn navnesen
Språk	en (*engelsk)*

Tabell 4: Spørsmål ved Sphinx hurtigstart

Når Sphinx er ferdig med å generere hurtigstarts oppsettet, gå inn i *docs/conf.py* og legg til innstillingene under "*General configuration*":

```
extensions = ['autoapi.extension']
autoapi_type = 'python'
autoapi_dirs = ['../Realtime-dwls/']
```

I ledeteksten som da peker på /prosjektMappe/docs/:

```
sphinx-build -b html . html_documentation
```

Dokumentasjonen er nå klar til å leses og finnes ved å åpne docs/html_documentation/index.html.

For mer informasjon vedrørende Sphinx, referer til dokumentasjonen: https://www.sphinx-doc.org/en/master/contents.html (*Sphinx documentation, u.å.*)

10. KONTINUERLIG INTEGRASJON OG TESTING

Dette kapittelet er utelatt da det ikke har relevans for prosjektet.

11. REFERANSELISTE

API reference. API Reference - Matplotlib 3.5.1 documentation. (u.å.). Tilgjengelig på: https://matplotlib.org/stable/api/index (Hentet 1. Mai 2022)

Costa-Luis, C. da. (u.å.). *TQDM*. tqdm documentation. Tilgjengelig på: https://tqdm.github.io/ (Hentet 1. Mai 2022)

HTTP for humans. Requests. (u.å.). Tilgjengelig på: https://docs.python-requests.org/en/latest/ (Hentet 1. Mai 2022)

JSON schema. JSON Schema. (u.å.). Tilgjengelig på: https://json-schema.org/ (Hentet 1. Mai 2022)

PAFY documentation. Pafy Documentation - pafy 0.5.1 documentation. (u.å.). Tilgjengelig på: https://pythonhosted.org/pafy/ (Hentet 1. Mai 2022)

Paho-Mqtt. PyPI. (u.å.). Tilgjengelig på: https://pypi.org/project/paho-mqtt/ (Hentet 1. Mai 2022)

Pandas documentation. pandas documentation - pandas 1.4.2 documentation. (u.å.). Tilgjengelig på: https://pandas.pydata.org/docs/ (Hentet 1. Mai 2022)

Pillow. (u.å.). Tilgjengelig på: https://pillow.readthedocs.io/en/stable/ (Hentet 1. Mai 2022)

PyMsgBox. PyMsgBox Basics - PyMsgBox 0.9.0 documentation. (u.å.). Tilgjengelig på: https://pymsgbox.readthedocs.io/en/latest/basics.html (Hentet 1. Mai 2022)

PYTORCH documentation. PyTorch documentation - PyTorch 1.11.0 documentation. (u.å.). Tilgjengelig på: https://pytorch.org/docs/stable/index.html (Hentet 1. Mai 2022)

Pyyaml. pyyaml.org. (u.å.). Tilgjengelig på: https://pyyaml.org/ (Hentet 1. Mai 2022)

Schema validation. Schema Validation - jsonschema 4.4.0 documentation. (u.å.). Tilgjengelig på: https://python-jsonschema.readthedocs.io/en/stable/validate/ (Hentet 1. Mai 2022)

Simple, consistent. Geocoder. (u.å.). Tilgjengelig på: https://geocoder.readthedocs.io/ (Hentet 1. Mai 2022)

Sphinx documentation. Sphinx documentation contents - Sphinx documentation. (u.å.). Tilgjengelig på: https://www.sphinx-doc.org/en/master/contents.html (Hentet 2. Mai 2022)

Statistical Data Visualization. seaborn. (u.å.). Tilgjengelig på: https://seaborn.pydata.org/ (Hentet 1. Mai 2022)