Ole Thomas Norbye Theisen
Torbjørn Trømborg

Bachelor's thesis

Data acquisition system for fault detection of guard rails

May 2021

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

NTNU
Norwegian University of
Science and Technology

Bachelor's thesis

2021

NTNU
Norwegian University of
Science and Technology

Ole Thomas Norbye Theisen
Torbjørn Trømborg

Data acquisition system for fault detection of guard rails

Bachelor's thesis
May 2021

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

NTNU
Norwegian University of
Science and Technology

Preface

About us

We are two students attending the Automation-engineering degree at NTNU in Ålesund, and this thesis is the final product of our bachelor study. We both have different professional backgrounds. One of us has prior technical experience working as an Instrument Technician. However, the other author, has no previous technical background. Therefore, we had different starting points when embarking on our journey writing this bachelor thesis.

What motivated us to engage on the topic was being part of a larger project that has been carried out for two years and that contains many aspects of automation and data engineering like machine learning, web development, and data collection from sensors that could be valuable learning for us. Also, one of us has been working part-time for the last three semesters developing an image classification application for the client. Therefore, had already acquire insight into the project and knew the client. Thus, choosing the topic for our bachelor thesis was a simple task.

About our client

iSi AS was our client for the bachelor thesis. iSi AS is as a small IT-company located in Åndalsnes and is specialising in developing IT solutions for digital collaboration and documentation. They have delivered IT-solutions for the road safety company Arvid Gjerde AS since 2006, which resulted in the emergence of the automating road rail inspections project .

Acknowledgement

We would like to thank our client for such professional collaboration, and for being integrated in the project of solving automated guard railing inspection. During the bachelor period, we have been treated as serious contributors to the project and have received all resources we have asked for. We want to give special thanks to iSi AS employee Basir Sedighi for assisting us on the full scale testing of the system and for always supporting us on technical issues.
We will also thank our supervisors for being supportive throughout the project.

Summary and Conclusions

Guard railing inspection is today a tedious task. The inspection is conducted by visually investigating the guard rail and the person investigating have to be a trained personnel for correctly identifying defects on the guard rails. The client is trying to automate this tedious process by developing a machine learning model to detect defects on guard railings by using images.

The aim of the thesis is to make an application that simplifies the process of getting usable images of guard railings, to be processed by the machine learning model. Another aim is to map each image with a GPS-coordinate, for reporting the defects to the NPRA. Furthermore the application must have a user interface that can be operated by untrained personnel and to be safely used while driving.

The results lefts us confident that we have developed an application with potential to be used by untrained personnel. It is still requiring further work to be a solid solution to the problem, but is now functioning as an application used by the client for capturing images to train their model.

Contents

Terminology

GUI Graphical User Interface, makes it possible to interact with a computer.

API Application Programming Interface, activates functions from a remote software.

TCP Transmission Control Protocol, connection oriented transmission protocol of information.

IP Internet Protocol is a "best effort" delivery protocol.

PTP Presistion Time Protocol.

PWM Pulse With Modulatio.n

RTK Real-time Kinematic Positioning.

SDK Software Development Kit.

FPS Frame Per Second.

NPRA Norwegian Public Roads Administration.

SSD Solid State Drive.

EHS Environment, health and safety.

Notation

ACK Acknowledge Message.

Abbreviations

NVDB Nasjonal Vegdatabank

NMEA National Marine Electronics Association.

List of source codes

List of Figures

Chapter 1

Introduction

"Feil på autovern medvirker til mange dødsulykker. Tre typer feil går igjen"

NRK.no [27]

1.1 Background

Norway is an elongated country with high peaks, deep valleys, and narrow fjords. Thus, it is challenging to build and maintain roads' infrastructure. Often, guard railings get damaged, and incorrect set up can be the cause of fatal accidents for motorists. Loose bolts, wrong height, damage made by the plow truck, and corrosion are some examples. The Norwegian Public Roads Administration (NPRA) is responsible for maintaining the road infrastructure in the country, and they are in charge of carrying out yearly inspection of the guard railings. This procedure consists of examining the fences with unassisted eye, usually from a car at low speed that requires one driver and one passenger. The inspection is, evidently, time-consuming, and makes fault detection a challenging task C.

In 2019, NPRA started the search for new technology for guard rail inspection and encouraged companies to use today's technology for developing a more effective, safe, and less time-consuming method for detecting guard rail faults. Some of these technologies involve image recognition with deep learning, and for detection accuracy need large amount of images for its

training and evaluation. iSi AS, our client, is developing such deep learning model for detecting faults on guard railings; therefore, they need data for training and testing their deep learning algorithm.

1.2 Project introduction

1.3 Problem Formulation

Conventional guard rail inspection is time-consuming. It is performed by splitting up the guard railings in pieces of 100 meters and then by car or foot examine the guard railings for faults. After the 100 meter inspection, the findings are reported. The recommended speed is 0-15kph, and the inspector has to detect if the railings are skewd, have missing bolts, loose bolts, correct height, or any other fault. C.

This report will look at the possibilities of developing an application for data capturing used in fault detection of guard rails. The requirements of the application was that it should consist of a GUI and have the ability to capture and store images from three cameras without loss and to map each image with a GPS-coordinate. The images must also be taken synchronously because two of the cameras are functioning as a stereo-camera and the client is reconstructing the guard railings in 3D. The specification of the GUI was making it as user-friendly that it could be used while driving and be operated by a person with no prior technical understanding.

1.3.1 Scope

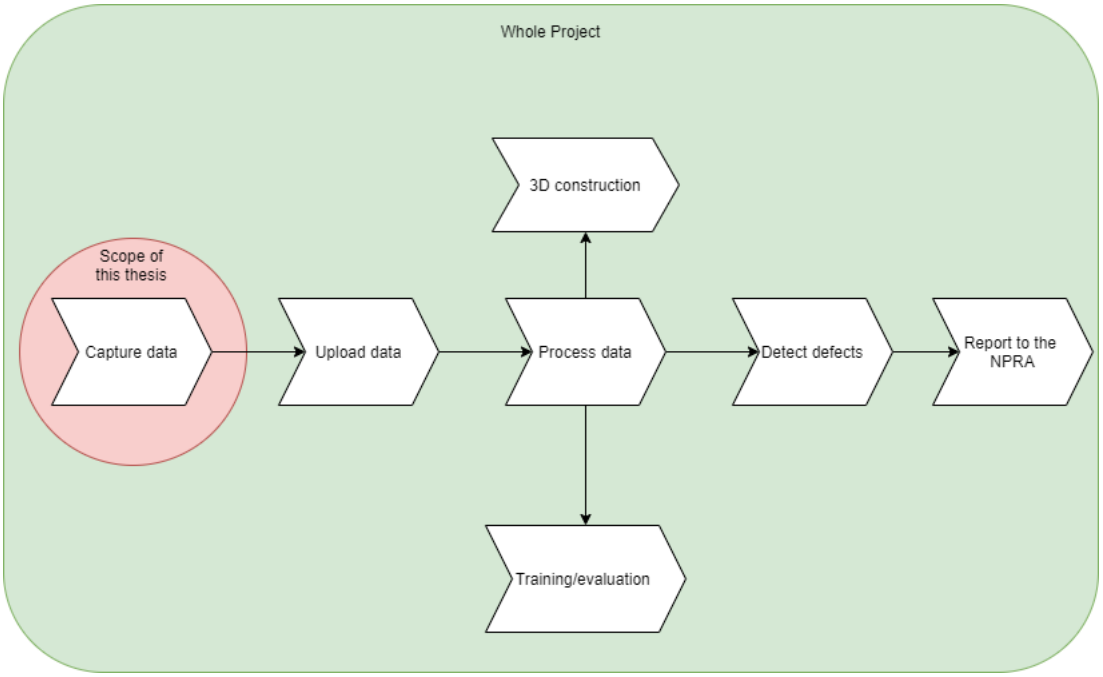


Figure 1.1: Scope

1.4 Aim and Objectives

The aim of this project is to develop an application for capturing data for used for machine learning for automating road railing inspection. The application will be an instrument for road safety companies doing road railing inspection. Core objectives for developing this application are mentioned below.

1. Making the GUI more user friendly and drive-safe.
2. The system could be operated without technical background.
3. Capturing images without loss.
4. Receiving valuable feedback (GPS, video, storage).
5. Mapping GPS position to image.

1.5 Limitations

This project's product is a system that requires special hardware. That hardware is a I/O optimized computer, GPS and three cameras. For testing the software through the development process, it is crucial to use the hardware that will be part of the final product. Because of the limitations of the testing-equipment the debug of the software was not carried out fully until it is was full scale tested.

1.6 Structure of the Report

The rest of the report is structured as follows.

Chapter 2 - Theoretical basis: The theory basis relevant for the subjects presented throughout the report.

Chapter 3 - Method: Shows the methods used to create and test the various elements in the project, leading to the system solution and contains a overview of the materials, information and components used in the system.

Chapter 4 - Testing: Presents all tests conducted during the project.

Chapter 5 - Result: Shows the final implementation of the end result of the project

Chapter 6 - Discussion: Contains the discussion regarding the different results achieved, and some thoughts regarding the choices made.

Chapter 7 - Conclusions: Presents an overall conclusion to the final state of the project.

Chapter 2

Theoretical basis

2.1 Concurrency Python

2.1.1 Threading

The threading module in Python provides an high-level threading interface for executing concurrent tasks. Because of the Global Interpreter Lock, only one thread can execute Python code at once. The multiple threads share the CPU, and execute the code "simultaneous" by executing parts of code when resources are available.[28]

2.1.2 Multiprocessing

Multiprocessing package uses an API similar to the threading module and offers concurrency by running tasks in sub-processes on multiple cores of the computer. [7]The processes can exchange values using a shared information state.

2.1.3 Queue

Queue is a module that is used for safely exchanging information between threads. This module is also implemented in the multiprocessing package. The queue is operating as a stack, were threads can put or get items from the stack. [9]

2.1.4 Pipe

The pipe module comes from the multiprocessing package. The module uses a double duplex pipe for exchanging data between processes. [8]

2.2 Communication protocols

2.2.1 HTTP

Hypertext Transfer Protocol is the foundation for data communication for the World Wide Web since 1990. The HTTP is a request/response protocol based on the client/server architecture where web browsers, robots, etc. act as HTTP clients and the web-server acts as a server. It provides a standardized way for computers to communicate with each other [40].
HTTP is a connection-less communication protocol where the client initiates a request and waits for the response. Then, the server processes the request and sends a response back, after which the client ends the connection. The server and the client is only aware of each other during current requests [33].

2.2.2 Websocket

The term WebSocket is a computer communication protocol used to transfer data over a single TCP connection. As WebSockets provide full-duplex communication, the data can be transferred both ways without blocking. The basic functionality of WebSockets relies on event-driven actions which supports real-time performance between two communicating entities. Since the WebSockets are designed to work over HTTP, most common sessions usually consist of a web client and a web server [52].

2.2.3 Serial

Serial communication is the process of transferring one bit at a time, sequentially, over a communication channel or computer bus. Serial communications cables are used in all long-range

communication and most computer networks due to cables' cost compared to other alternatives [47].

2.2.4 TCP/ip

TCP/IP is a collection of communications protocols to interconnect network devices to the internet. It is also used in private networks as a communication protocol. TCP/IP provides end-to-end communication that identifies how data is broken into packages, addressed, transmitted, routed and received[30].

2.2.5 Bluetooth

Bluetooth is a protocol for wireless transmission of data over short distances. It operates in frequency bands 2.402-2.480GHz, and the range is power-class-dependent, but effective ranges vary in practise and is much lower than the line-of-sight ranges of the Bluetooth products [36].

2.2.6 PTP

Precision Time Protocol is used to synchronise clocks over a local network where microsecond accuracy is required [45].

2.3 GNSS

Global Navigation Satellite Systems refers to all satellites that deliverers global navigation and position service. It consists of four systems; The American GPS, The Russian GLONASS,The Chinese BeiDou and the European GALILEO. [26]

2.3.1 Trilateration

Trilateration is the technique a GPS uses to find its position. Trilateration uses distance to find a point. It uses four satellites to find the location. Two satellites creates an intersecting sphere. The third satellites creates two points were all three satellites intersect. Of the two remaining

points one is outside the earth's atmosphere. The location can be found by using three satellites, however to achieve this the GPS clock needs to be accurate, otherwise it cant calculate the distance properly. The fourth satellite is used to correct the GPS internal clock [3].

2.3.2 NMEA sentences

NMEA is the standard data format and supported by all GNSS manufacturers. The standard makes it easier for developers to write software for a wide variety of GNSS-receivers without customizing interfaces for each receiver. The NMEA data format is structured by many different types of sentences, and the sentences the receiver provides depends on the capabilities of the GNSS receiver. [15]
This a an example of a $GPGGA NMEA sentence:
"$GPGGA,181908.00,3404.7041778,N,07044.3966270, W,4,13,1.00,495.144,M,29.200,M,0.10,0000*40"

- All sentences start with "$".
- "GP" is that it uses GPS-position. GL for GLONASS.
- "GGA" is the message identifier.
- All data is comma separated.

2.3.3 RTK

Real-Time Kinematic is a method for getting real-time position measurements from a GNSS system down to an accuracy of 1 cm by correcting the position using fixed base stations which have a known position. The technique uses the relative position of the GNSS-receiver in relation to a fixed base station that has to be located a maximum of 10km from the GNSS-receiver. The base station computes the position of it self by using the same satellites as the GNSS-receiver and sends the correction data over radio-link.[23]

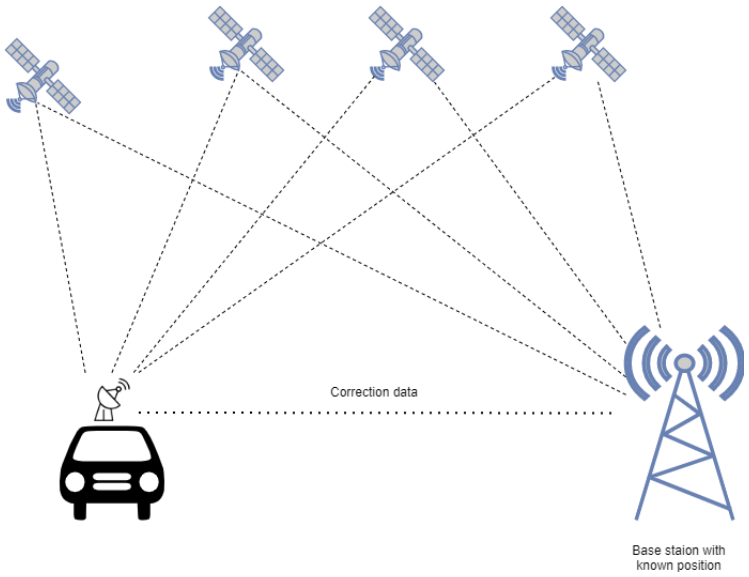


Figure 2.1: RTK

2.3.4 CPOS

CPOS is a service delivered by Kartverket, and gives the GPS-receiver RTK by a connection to its server over GPRS/4G. Kartverket has a fixed base station all across Norway and, by sending the GPS-receivers position to the server, it can compute the position of the closest base station relative to the GNSS-receiver and send correction data. This gives the GNSS-receiver corrected data as long as it is connected to the CPOS-server. [20]

2.4 Misc

2.4.1 Master/Slave

Master/slave is a model where the master controls the slaves. The master dictates what the slaves can do, while the slaves can only sends information to the master, not commands. Inserting data to the system can only be done through the master [42].

2.4.2 Client-Server Model

Client–server model is a distributed application dividing tasks between server and client. They can be in the same system or communicate over a computer network. A server is a physical computer that provide a function or services. The client accesses the function or services. [38].

2.4.3 REST-API

REST-API is a software architecture style that gives clients a standardized and easy way of requesting resources from a server. It is designed to have some limitations that makes it easier for developers to give API-calls. [46]

2.4.4 TIA

International Atomic Time is a time standard that uses the combine output of around 400 atomic clocks in 69 laboratories worldwide [48].

2.4.5 UTC

Coordinated Universal Time is the time standard used to synchronise clocks and time worldwide. In 1963 the UTC was officially formalized. The UTC is based on the TIA. As a result of the changing motion of the earth, the system was adjusted several times until 1972, when leap seconds were implemented to simplify future adjustment [50].

2.4.6 Network Time Protocol

Network time protocol is a network protocol that synchronises time over multiple servers. Its intended use is to synchronise computers within a few milliseconds of the UTC [43].

Road Reference System

The road reference system (NVDB) is containing descriptions of the physical road infrastructure of Norway [34]. Figure 2.2 shows an example of a reference of a road section.

Figure 2.2: Example of road reference [35]

2.4.7 PWM

Pulse-Width Modulation is a technique to control analog devices with digital pulses of microcontroller's. PWM imitates an analog signal by applying voltage in pulses or short bursts of regulated voltage [16].

2.4.8 Haversine formula

The Haversine formula is used for calculating the distance between two points on the earth using latitude and longitude [22].

2.5 Version control

2.5.1 GIT

GIT is a software control system version used for coordinating software projects. It allows people to work on the same source code simultaneously. The main focuses of GIT are speed, data integrity, and support for distributed non linear workflow [39].

Chapter 3

Materials and methods

This section explains how the project was organized, planned and how we ensured progress throughout the semester. It describes the methods used to solve the problems mentioned in the introduction and also which programming languages and tools were used to recreate this project.

3.1 Project Organisation

Client:

iSi AS

Supervisors:

Ottar L. Osen

Robin T. Bye

Students:

Ole Thomas N.Theisen

Torbjørn Trømborg

3.1.1 Project group

The project group consists of two students at NTNU in Ålesund, Ole Thomas N.Theisen and Torbjørn Trømborg.

3.2 Planning

Before we started the bachelor project, we wrote a pre-project report that attempted to identify the scope and requirements of the main project. In that process, we had regular video meetings with the client and a final workshop were we established all necessary requirements of the main scope and nice-to-have features that were outside the main scope. These features were meant to be implemented in the project if we satisfied the client's requirements. In the pre-planning we also agreed on a code of conduct for maintaining a continuous progress throughout the project. For monitoring our progress, we made a GANTT diagram B. The latter was of great use for tracking progress during the thesis and for distributing tasks between group members.

A meeting with the client was attended every two weeks. It was conducted to give a status report and receive the client's suggestions on improvement for our application.

Note; The original project group originally consisted of three students, but because of personal reasons the third member had to quit the project. This caused a increase in workload, but we continued with the original scope of the project.

3.3 Existing software

The existing software 3.1 was designed as a desktop application and developed in the Python GUI framework PyQT. The application was operated with a remote desktop connection from an external computer. The existing application were able to acquire images from two cameras with a fixed FPS.

Figure 3.1: Excising GUI

3.3.1 Existing Hardware

All the existing hardware is used. Additionally, our application has an extra camera in use.

3.4 Software

This section presents the plans for developing the software needed for the application.

3.4.1 GUI

In the planning process, working with the pre-project repport A, we had a workshop with the client discussing different solutions for the GUI. One of the main concerns was making the GUI drive-safe so that the driver could safely operate the data capturing. To make it safe to

use while driving, we had to simplify the GUI so that the operator did not need any technical understanding of the application; in other words, we were making the GUI as intuitive and self-explainable as possible. Other discussions topics were how to make the GUI platform and device-independent, making it possible to be used on any device such as a mobile phone or on any operating system (mac OS,Windows or Linux).

Choosing GUI framework

When choosing the GUI framework, the most critical factor was to make the GUI device and platform-independent. Since every smart device has an internet browser, the best approach was to create a web-GUI, developing in HTML, Javascript, and CSS. Moreover, since one of the authors had experience with web development and the limited time available, it was more advantageous over other frameworks that had required a much steeper learning curve for successfully making a satisfying GUI. Therefore, the GUI will work as a client for a server responsible for executing all business logic.

Making a Web-GUI will translates into easier maintenance and update of the GUI by separating the business logic from the user interface, that is, making it two separate systems. Unlike the existing software, that was created as a desktop application and all code associated with the program was in one file, which made future maintenance/updates of the application more difficult and/or time consuming.

Design

In the workshop, we came up with a sketch 3.2 that was the base of our GUI design. It consisted of four squares. Three squares for where the operator was to get feedback of the condition of the GPS, image-quality, storage, and the fourth one represented the system's overall condition, making it easy for the operator to determine the state of the system.

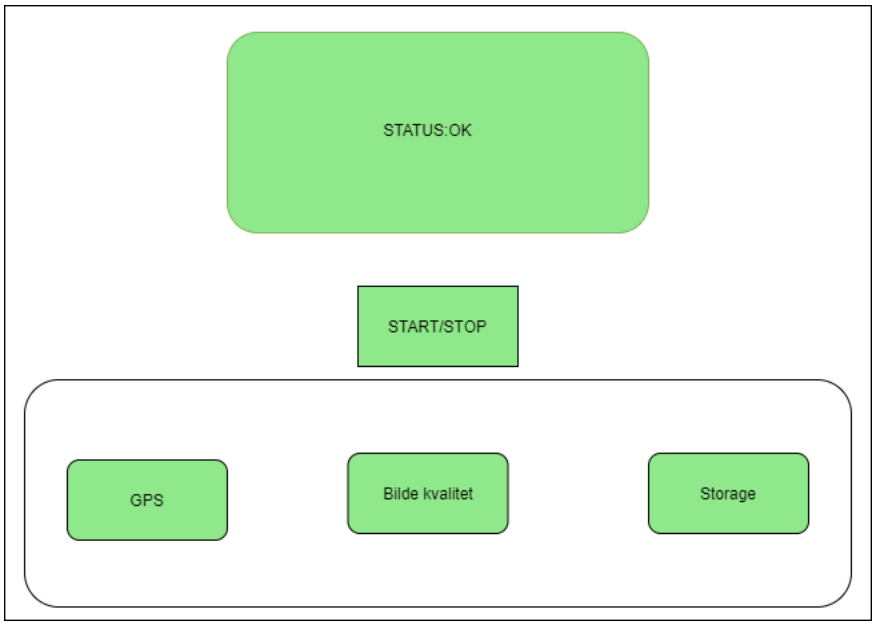


Figure 3.2: GUI sketch

User feedback

The user feedback has to be easy to understand and not distracting, therefore a color-system will be used to indicate the different states of the system. The colors green, yellow and red are commonly used to indicate three different states of a system like a traffic lights, etc. We think by using this three colors, it will be intuitive for any operator of which state the system is in.
The operator will be able to press the GPS and storage button to get more detailed information, this can be used for support.

3.4.2 Server/Backend

Since we developed the GUI as a web application, it is most appropriate to develop a server that uses communication protocols that support the web, like HTTP and Web-socket.

Choosing framework

We wanted a fast and easy-to-implement framework that was written in Python. The choice was FastAPI, which has built-in Web-socket and HTTP integration that enables communication with the GUI. Expressly, FastAPI-framework runs on a ASGI-server that provides asynchronous

executions of I/O blocking tasks.

FastAPI

FastAPI supports asynchronous coding and provides methods for running long-lasting I/O-blocking tasks in threads without the need of initiating a native Python thread. Moreover, it uses "async" and "await" syntax for executing tasks that needs to be awaited, i.e waiting for a response from the client or searching through a online database. When the "async/await" starts running, it tells the program to wait for the task to be finished and to go do something else. If the function is created with a standard "def" declaration, it will run as a separate thread in a thread pool and cleared after it has completed the task. Thus it provides the means to a good framework for executing long-lasting tasks without blocking the server.

The HTTP endpoints of the server is decorated as shown in figure 3.3. With the decoration the user can define the request-type of the endpoint (GET,PUT,POST).

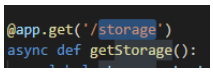


Figure 3.3: Decorating endpoints

3.4.3 Data acquisition

Image capturing

In order to acquire usable images for the machine learning model, the system uses advanced cameras 3.5 that require an SDK called "Common Vision Blox". This SDK provides an API for interacting with the cameras. The API supports multiple programming languages included Python; and, since we are writing the backend code in Python, it was most suitable to use the Python-API.

The system has three cameras. Each one of them are required to capture one image per meter, so at a speed of 80kph, the system has to acquire up to 66 images per second. The images are transferred from the camera over TCP/IP in BMP format and require a bandwidth of 44Mb/s

on each ethernet port. The application must, therefore, be fast enough to get all images without any loss. Furthermore the images have to be captured synchronously, because each image must correspond to the same place and time. The three images must be taken at the exact same moment, because of the post-processing of reconstructing guard railings in 3D and giving each image a GPS-position. The software must also store the images to a SSD/HDD and the image capturing rate(FPS) have to be controlled to reduce abundant images.

The methodology of the mentioned aspects of the image capturing will be explained in the next paragraphs.

Capturing images without loss An essential requirement of the system is to capture all images. The loss of images would probably lead to the image synchronisation to stop. For reconstructing of a 3D model, images from all three cameras are needed. In the case that the image synchronisation didn't stop due to loss of images, we could end up with and incomplete picture of the state of the guard railings.

Camera synchronization

To synchronize the image capturing, the camera's interface provides two different solutions: Controlling by software, using the CVB-API 3.5.3; or controlling by hardware, using the internal I/O of the cameras.

Software

The internal clocks of all three cameras are synced by PTP. One camera is the master and the two other are slaves. Then, the image rate of the master-camera can be controlled by setting it through the CVB API. The slaves are triggered by events from the master-camera, for instance "frame start" [19].

Hardware

The main camera has an external I/O which triggers the cameras by giving the pulse to the input. This gives the user control of the image rate by adjusting the frequency of the pulses.

The solution for this project was the hardware sync, because it gives the user full control of the FPS and the synchronization, by controlling the frequency of the pulses.

Adaptive image capturing rate

For reducing abundant images, the PWM-frequency is being adapted by the current speed of the vehicle. This is done by using a GPS to receive live speed and converting the speed to m/s to trigger the cameras every meter. This way we get an image every meter and reduce the overlap of the guard rails. This solution also prevents the system to capture images when the vehicle stops.

GPS data

The GPS is providing information about the position and speed of the vehicle. The accuracy of the position is crucial for post-processing the data to provide each image with a unique GPS-coordinate for later maintenance of the detected defect. Because the system is specified to operate at high speeds, the update frequency from the GPS should be as high as possible to determine the image's position accurately. For testing, we used a cheap consumer GPS(GNS 2000) with the same communication protocol (Serial) as a more advanced GPS, so we could adapt the same code for the industrial GPS3.6.3 used later on the car system. The testing-GPS had an update rate of 1Hz and outputted the standardized NMEA sentences, and gave us an adequate data foundation for later testing the program with the more advanced GPS.

Accurate positioning

The GPS accuracy depends on the number of satellites in view, line of sight, and the receiver's quality. It needs at least four satellites for Trilateration to determine a location on the earth's surface. The more satellites it uses, the more accurate the position is. For achieving more accurate positioning, a more advanced GPS can rely on a technique called RTK to get the accuracy down to centimeters.

RTK is a technique to enhance the precision of the received data from the GPS. By real-time correction, it provides centimeter-level accuracy. The Norwegian Kartverket has base stations located all across Norway, and by obtaining an Internet connection to Kartverkes servers, the

CPOS system can provide real-time corrections all across Norway.

Determine the precision of received GPS data

To determine the GPS quality, we interpret the GGA sentence received from the GPS and use this to calculate the quality of the data. The 3.4.3 method takes the "hdop" and "quality" fields from the sentence as parameter and returns a readable quality from bad to best. Where "Best" is RTK. The method was provided by the client and it is the method used by road entrepreneurs for ensuring that the GPS is receiving centimeter-level position when reporting guard railing faults to the NPRA.

Listing 1: Method for ensuring GPS quality

```
def getGpsQuality(self,fixQuality, hdop):
      gpsQuality = GPS_QUALITY.BAD.value
      if (fixQuality and hdop > 0):
          if (fixQuality == 4):
              gpsQuality = GPS_QUALITY.BEST.value
          elif ((fixQuality == 5 and hdop < 2) or hdop <= 1):
              gpsQuality = GPS_QUALITY.GOOD.value
          else:

              gpsQuality = GPS_QUALITY.LOW.value


      return gpsQuality
```

Logging

One of the requirements for the application was to give each image an GPS position. For this, we log GPS and image data during the inspection and store them on the server. Logging is also done for storing historical data and help the user troubleshoot the application.

Map each image with GPS-position

When they detect a fault in conventional road rail inspection they have to take the GPS-coordinate. The defect-type and position are then reported back to the NPRA for them to announce a tender for road entrepreneurs to correct the fault. The images captured by this application is being used for detecting defects by using a machine learning model. For knowing were the defect is located, it is crucial that the image can be traced back to position on the map for reporting back to the NPRA.

3.4.4 Data structuring

It is important for this system to store the data in a structured way. All log files and images need to be stored so that every log folder corresponds to each image folder, so it is easier to use images of a specific road section to detect guard railing defects with the machine learning model.

Storage of images

To not overload the computer's internal storage space and reducing its capabilities, we save the images on an external SSD/HDD. This facilitates the post-processing stage, when the operator will have to upload the images to the online database, by disconnecting the HDD/SSD's from the car-system and connect them to an online computer.

3.5 Materials

This section presents what hardware and software was used in developing the system.

3.5.1 Programming Languages

Python

The Python programming language is a high-level general-purpose interpreted language. It can be used for web and software development, mathematics and system scripting among other use cases. Python can be written in a procedural, functional or an object-oriented way. It is based on a dynamically typed style and handles garbage collection automatically. Python has a large amount of different third-party libraries [32].

Typescript

Typescript is a programming language which builds on JavaScript and is mainly used in web development. in addition to JavaScript, it has static type definitions, which gives the user more control by validating their code with the Typescript-compiler or Babel. The compiler checks for errors and transforms the code into JavaScript. JavaScript is included in every browser, and is responsible for making advanced and dynamic web pages [49].

CSS

CSS is the language for styling a HTML-document. It describes how the elements should be displayed [37].

HTML

HTML is a markup language employed for creating web pages and describes how the web page is structured [33].
A markup language is human-readable that consists of standard words instead of typical programming syntax [41].

3.5.2 Tools

Vegdata API

The NPRA is offering a REST-API-API that gives access to information that is located in the Norwegian road data bank (NVDB).

GIT

Git is a software version control system used for coordinating software projects. It allows users to work on the same source code simultaneously. The main focuses of GIT are speed, data integrity, and support for distributed non-linear workflow [39].

3.5.3 External Libraries and frameworks

Angular

Angular is an open source web development framework based on Typescript and is for creating web applications. [4]

NGRX store

NGRX is a global state manager for building reactive applications in Angular.Having a global store makes it a convenient way of sharing states between components and services.[10]

RXjs

RXJS is a library used for making asynchronous and event-based JavaScript programs. In a nutshell, the user can make observable objects which others can subscribe to. Users can emit data through a program observable to all subscribers. The subscribers wait for the emitted data and execute it when it has arrived.[6]

Leaflet

Leaflet is JavaScript library for making interactive maps. [21]

FastAPI

FastAPi is a web framework for building APIs with Python.[14]

OpenCV

OpenCV is a cross-platform library with programming functions for computer vision applications.[44]

CVB

Common Vision Blox is a computer vision library developed by Stemmer Imaging. It provides high performance tools for image processing and SDKs for developing computer vision applications. CVB Camera suit offers an SDK for image acquisition from GigE vision cameras that uses the Genicam standard.[17]

Numpy

Numpy is a library for Python, for working with arrays. It is partially written in Python, but all the code that requires fat computations are written in C++ or C.[31]

Nginx

"NGINX is an open source software for web serving, reverse proxying, caching, load balancing, media streaming, and more. It started out as a web server designed for maximum performance and stability. In addition to its HTTP server capabilities, NGINX can also function as a proxy server for email (IMAP, POP3, and SMTP) and a reverse proxy and load balancer for HTTP, TCP, and UDP servers."[25]

RPi.GPIO

RPi.GPIO is a Python library for accessing the GPIO of the Raspberry pi [5].

CSV

The CSV library is used to read from and write to CSV files.[12]

Bisect

The bisect library is useful for long lists of items with expensive comparison operations.[11]

Haversine

Haversine is a Python library that uses the Haversine formula [22].

Time

The Python library time provides various time related functions [13].

Pynmea2

Pynmea2 is a library for parsing NMEA-data to and dict object [1].

3.5.4 Development tools

Visual studio code

Visual Studio code is a source code editor. It includes features to simplify coding [51].

Common vision blox

Common vision blox is a development tool made by Stemmer imaging. It provides tools for image vision applications.[18]

3.6 Hardware

This section contains a description of the hardware operated during the project. The hardware is composed by the data acquisition-system.

3.6.1 ECX-1420

ECX-1420 is a special I/O optimized computer designed for the purpose of collecting large amount of data. It is equipped with PoE and has multiple high speed USB ports. This computer is used in the production-system and was not used for the initial testing of the software. The initial testing was done using our own computers.

Figure 3.4: ECX-1420 found at https://tinyurl.com/3z5zrde7

3.6.2 Genie Nano

Genie Nano is the camera operated to capture images during the project. The Genie Nano is a small high speed camera that supports a data-transfer rate of 252MB/s, which is equal to a frame rate of 84 FPS. The camera is supplied with an external power supply, or by "Power over Ethernet" if the Ethernet ports on the computer/router supports PoE. The camera also has an internal I/O for setting user specific inputs and outputs[24].
The choosing of the cameras was done by the client, and was provided to us for testing and implementing them in our software.

Figure 3.5: Genicam Nano found at https://tinyurl.com/tdthuwzy

3.6.3 GPS

For accomplishing the task of giving each image a GPS-position and controlling the frame rate of the camera a GPS was needed. The GPS had to be accurate and have a high data-rate for finding the closest position based on the timestamp of the image and the timestamp of the coordinate. Two GPS were provided by the client; one for testing (GNS 2000) and one for production (Altus sp3). Both GPS used serial communication as interface and outputted the same NMEA-format. This was convenient when developing using the testing-GPS, the same code could be used on the production-GPS with minor changes.

GNS 2000

The GNS 2000 was used for testing the software used later on the production-GPS. It connects via Bluetooth and outputs its data with a frequency of 1hz on the serial-port.

Figure 3.6: GNS 2000 found at https://tinyurl.com/ta475m5h

Altus aps3

The Altus aps3 is the GPS used in production and has much better specs than the one used for testing. The Altus aps3 is programmable. By writing commands on the serial port, the user can specify what NMEA-sentences to be outputted, set data-rate (max 20hz), and connect it to a CPOS server to get RTK-corrections. The Altus aps3 connects via Bluetooth or USB.
The GPS was only used at the full scale testing of the system because the GPS was not owned by the client.

Figure 3.7: Altus aps3 found at https://tinyurl.com/4au9nkcj

Raspberry pi 3

The Raspberry pi is used for triggering the cameras. By outputting a PWM-source to the internal I/O of the camera it was possible to control the image capturing rate.
The Raspberry pi is a small computer that contains most of the same components as a normal computer. It also has an internal I/O that can be programmed to send digital outputs and receive inputs [2].

Figure 3.8: Raspberry Pi found at https://tinyurl.com/h58uf388

Chapter 4

Testing

This chapter describes the different tests conducted during the development of the application

4.1 Tests

4.1.1 Consecutively saving of images

In trying to save images consecutively, different programming techniques were tested to ensure the highest speed for accomplishing the capturing of all images at the highest FPS (22) and save them to the SSD. In the testing we fixed the output of the PWM controlling the frame rate and logged the avg FPS in the camera threads. Then comparing real FPS to measured FPS. The testing of these methods are described in this section.

Method 1

In this method the camera-stream thread was plaing the object containing the image in a queue, and the image saving process was reading from the queue 2.1.3 see figure 4.1.

FPS	Measured
10	9,66
15	10,15
20	10,15

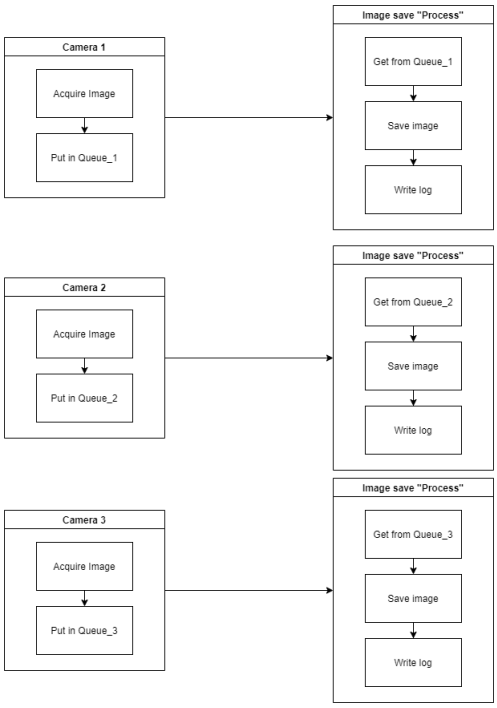


Figure 4.1: Consecutively saving of images method 1

Method 2

In this method, the image saving class was run as a separate process and the communication between the camera-stream thread and process was done by using a pipe. See figure 4.2 .

FPS	Measured
10	8,26
15	8,69
20	8,70

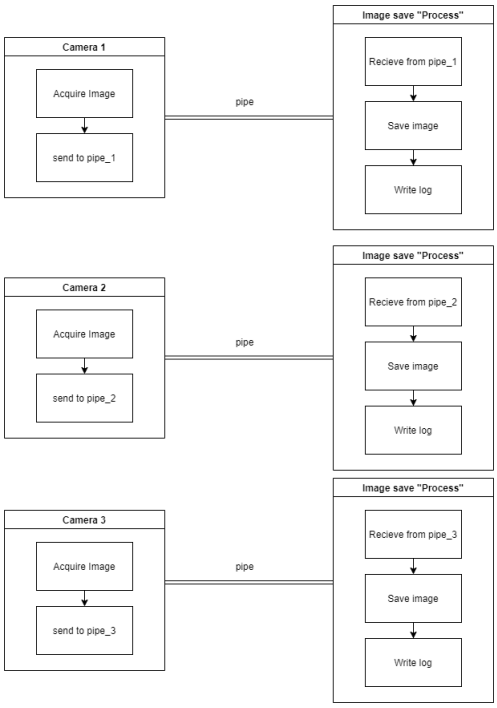


Figure 4.2: Consecutively saving of images method 2

Method 3

Method three ran the camera streams in separate processes and communicate from the server using shared memory mapping 2.1.2.As seen in figure 4.3, the process is responsible for acquiring the image and putting it in a queue. The image storing thread store the image on the SSD/HDD. With this method we managed to save images at a rate of 19 FPS maximum.

FPS	Measured
10	10
15	15
20	19

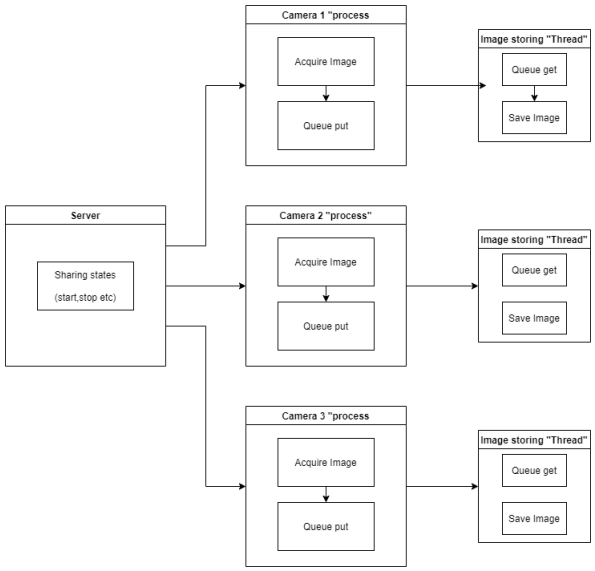


Figure 4.3: Consecutively saving of images method 3

Method 4

In this method the camera-stream thread fills the queue, and after the operator has stopped the image capturing, the image saving class saves the images to the SSD. With this method, the program stores the images in memory before storing it to the SSD. After 1500 images the computer ran out of memory and crashed.

4.1.2 Mapping each image with GPS-position

For easier and more controlled testing, smaller data sets were used in implementing of new features. Functions with calculations were tested by doing the calculation by hand to check if the output matched the desired output. In every part of the testing, the print method was frequently used to check if the functions worked as desired.

Method 1

In this method, we tested if the image was given the correct coordinates by comparing the image timestamp with the GPS timestamp.

Image timestamp	GPS timestamp	Time difference
1618928879942.00	1618928879900.00	42
1618928880438.00	1618928880400.00	38
1618928880942.00	1618928880900.00	42
1618928881446.00	1618928881400.00	46
1618928881960.00	1618928882000.00	-40
1618928882443.00	1618928882400.00	43
1618928882834.00	1618928882800.00	34
1618928882993.00	1618928883000.00	-7
1618928883226.00	1618928883200.00	26
1618928883393.00	1618928883400.00	-7
1618928883594.00	1618928883600.00	-6
1618928883807.00	1618928883800.00	7
1618928884025.00	1618928884000.00	25
1618928884192.00	1618928884200.00	-8
1618928884407.00	1618928884400.00	7
1618928884594.00	1618928884600.00	-6
1618928884808.00	1618928884800.00	8

Figure 4.4: Time difference test

Method 2

In this method, we tested the time difference between comparing every element together using for-loops vs comparing using the Bisect library to find the closest match. We used a timer function to get the time of the two functions.

GPS CSV file size	Bisect method(seconds)	For loop(seconds)	difference (%)
737	0.1421	1.3068	89
5435	0.2601	4.1667	93
14916	1.3284	39.6389	96
18633	2.2925	45.9692	95

Figure 4.5: Speed comparison test

Method 3

Visual test

In this method, the GPS position given to the images are modified based on the time difference between the GPS and image timestamp. The first test we plotted both read and modified coordinates to a website called maps.co. We observed if the modified points were moved in the correct direction. If the image timestamp was before the GPS timestamp, the calculated position should be moved ahead of the GPS coordinate and back if the GPS timestamp was before the image timestamp. The calculation of time difference is shown in figure 4.6

Image timestamp – GPS timestamp = Time difference

Figure 4.6: Calculation of time difference

Figure 4.7: Observation test

Marker test

The second test was a marker test. We marked the railings on six locations and measured the ground truth. To compare the calculated positions to the GPS position, we used a library called Haversine. With Haversine got the accurate distance between two coordinates. In figure 4.8 the distance between the coordinates is shown.

GPS coordnates	Calculated coordinates	
Distance (meter)	Distance (meter)	Differencial (meter)
11.44	11.32	0.12
6.97	7.07	-0.11
7.36	7.24	0.12
7.77	7.76	0.01
9.16	9.11	0.05
9.09	8.91	0.18

Figure 4.8: Comparing calculated and GPS position

4.1.3 Cameras (flash test)

We tested the synchronisation of two cameras by pulsing the blitz on a mobile phone and observed the stream of the cameras. Subsequently, we observed to see if the blitz appeared at the same time on the video streams. Finally, the test ran for 30 minutes to see if the streams would keep being synchronised.

Figure 4.9: Flash test

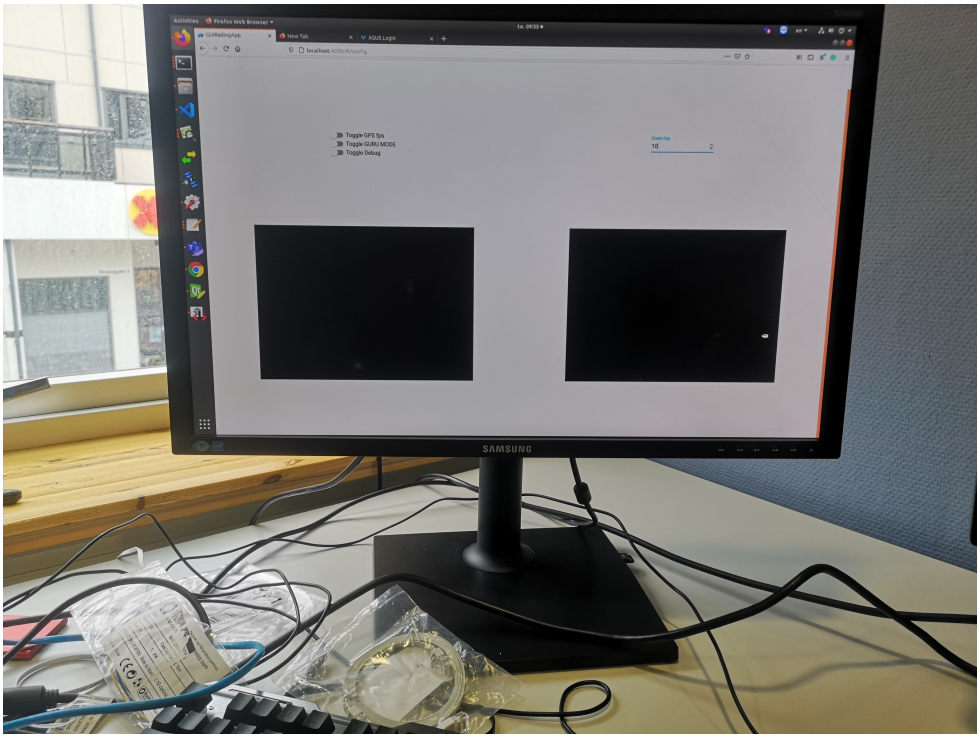


Figure 4.10: Flash test GUI

4.2 Full scale testing

The full scale testing was done at Åndalsnes.

4.2.1 EHS

The full-scale testing was done in a real-life scenario on highly trafficked roads with speed limits that reached up to 80 km/h and required a lot of stops along the road, slow driving, and exiting of the car to place out markers that were being used for ground truth for testing the accuracy of the GPS. Therefore, EHS measures had to be considered to minimize the risk before conducting the tests:

- An employee of iSi with a Certified road working course ("arbeidsvarsling") accompanied us for the tests. It is not only required by law to have someone with the course while working on the roads, but his experience was also very valuable at ensuring regulation being followed.

- The car used was equipped with blinking lights and a sign that says "Registration"
- Every person exiting the vehicle wore a reflex jacket to be more visible to traffic.

4.2.2 Testing rig

The camera rig is designed by iSi AS and is constructed by Arvid Gjerde. It consists of two flashes,one stereo camera pair and one individual camera.
Other hardware provided by the client:

- Special Customized computer.
- Raspberry pi (Transmitting trigger pulse to camera).
- Strobe light driver (Driving the flashes from a trigger input).
- GPS - Had to be initialized before driving to get the best quality.
- WIFI router (For accessing the GUI on the wireless network and connecting raspberry and computer).
- 3 cameras.
- Power converter 12/240V.

Figure 4.11: Camera rig

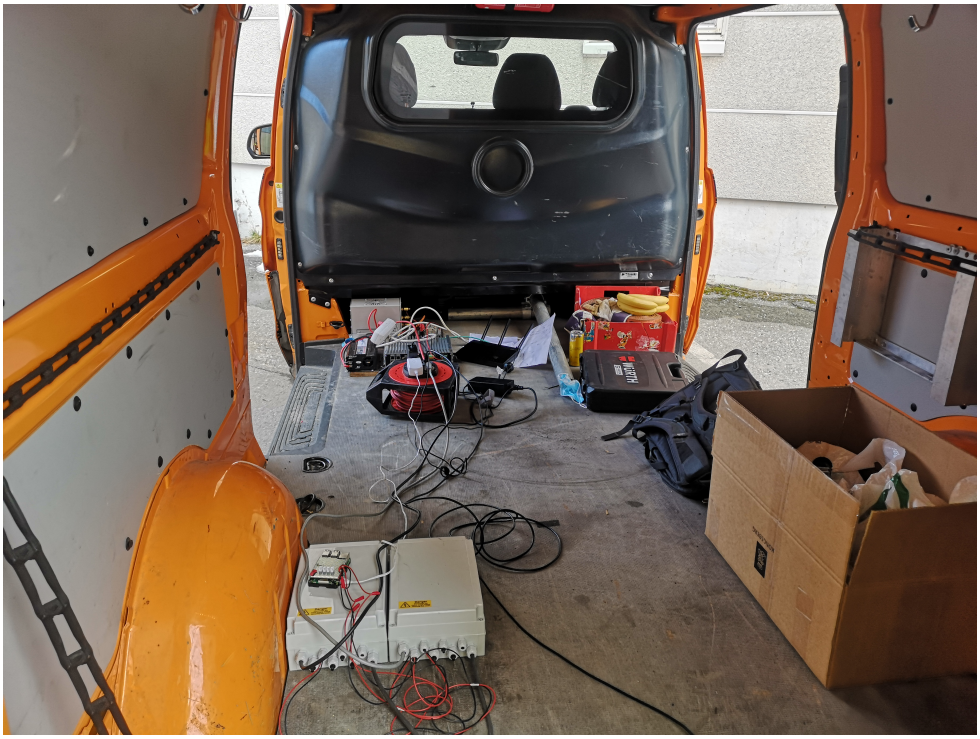


Figure 4.12: Inside of car

4.2.3 Preparations before each test

- Cleaning lenses.
- Made sure everything was fastened and correctly adjusted.

Testing software

- Setting the camera capturing rate to GPS speed.
- Saving of images and storing them correctly.
- Get equal number of images on each camera.
- Start/pause.
- Generation and saving of csv log files.
- Merging image-log and GPS-log to one csv file.

4.2.4 Testing 15.4.2021

This was the first test, and there were performed more thorough tests to check if the basic functions of the hardware and software was working as expected.

Pre check

Series of pre-checks were conducted before the full scale test.

Synchronisation

- Cameras (flash test) - OK @ 10hz 4.1.3
- GPS – runs @10Hz
- Numbering (frame Ids) -OK
- Long time testing (20-30 minutes) -OK

- Camera queuing -OK
- All sensors data should be recorded along with the timestamp in milliseconds -OK
- IR Flashes -OK
- Reliable cable connections -OK
- Third camera synchronization - Not testet

Camera settings

- Settings-OK
- Auto brightness -OK
- Auto expusure time -OK
- Auto whitning -OK
- Adjusting same white balance for both cameras-OK

GPS accuracy

- Could not get fix in office

Storage

- Capacity -OK 0.8 TB

User interface

- Starting the system -OK
- Getting useful information on status of data capturing- GPS and storage
- Pausing capturing -OK
- Ending capturing -OK
- Real-time visualisation -OK

Test Conditions

We chose the "Isfjordvegen" to do the full car system test. The stretch of road was suitable because of low traffic, dry roads, and damaged railings. Low traffic was essential for safe working conditions. Dry roads were needed to minimize the dirt on the camera's lenses, which would have resulted in poor image quality. For testing the system, damaged railings were not necessary, but iSi required stereo images of damaged railings for another project.

4.2.5 Marker test

We marked four spots on the railings with tape and recorded the GPS position on all four spots. Then, we drove with our system on the route and started image capturing. Later, we found the corresponding timestamps in the image-log and the GPS-log, and compared the GPS positions.

Test run setup

Washed the lenses between each run Test 1:

- Drove without pausing
- FPS based on speed from GPS

Test 2:

- Paused the system when the were not railings
- FPS based on speed from GPS

Test 3:

- Marker test
- Drove without pausing
- FPS based on speed from GPS

Test results

Image capturing

The system captured and stored images in correct folders and with no data loss, but the cameras did not capture the same amount of images. Starting and pausing the image acquisition worked well with some minor bugs. Nonetheless, stopping the acquisition worked as expected.

GPS

The GPS got RTK and the highest signal quality. The accuracy was good and the GPS-program worked as expected.

Logging

The logging was working as expected and the two csv files were saved and stored on the server.

Finding position of marker from image

When merging the two csv-files for finding the position to the image containing the marker, the GPS position did not match the ground truth as shown in figures 4.13 and 4.14. The position was approximately 10m off at every marker.

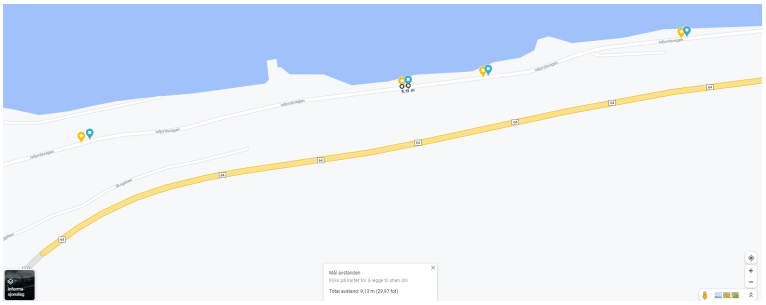


Figure 4.13: Route with marked GPS-positions flag(yellow) and recorded(blue)

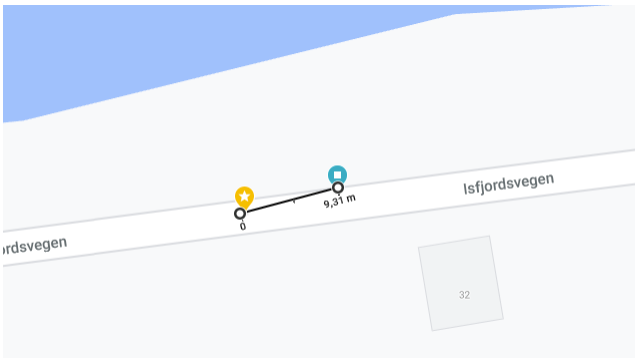


Figure 4.14: Error distance between points

Issues to be resolved

- Test to see if the pulses from the Raspberry pi have sufficient power to drive all the three cameras + flashes.
- Update the GUI to save the current state of the system. Problems of loosing the state of the system when refreshing the GUI.
- Find a better solution for saving the timestamp of the image.

4.2.6 Testing 20.04.2021

In the last test day, there was an issue with acquiring the same number of images from each camera. We found out that the problem lays with the pulses given from the Raspberry pi. It had a voltage output of 3.3V and a maximum current draw at 16mA, which was not sufficient to give pulses to the three cameras and the two blitzes. The solution to the issue was to only give pulses from the Raspberry pi to the cameras, then, use the output from the camera to give pulses to the Strobe lights.

Test 1:

- Paused the image acquiring when the were not railings
- Fps based on speed from GPS

Test 2:

- Drove without pausing
- Fps based on speed from GPS

Test 3:

- Paused the image acquiring when the were not railings
- 20 Fps

Test results

Image capturing

There was still a problem of capturing the same amount of images, but the difference is a lot smaller compared to the first test. With the new arrangement of the wiring, and the Raspberry only outputs its PWM to the cameras we have reduced the problem.

GPS

The GPS got RTK and the highest signal quality. The accuracy was good and the GPS-program worked as expected.

Logging

OK

Finding position of marker from image When comparing the ground truth position of the markers with the position of the image, it was still inaccurate by many meters. After analysing the data, we found out, by moving the timestamp of the images about 1,5 seconds, that the positions corresponded with the ground truth.

Issues to be resolved

- Connect computer to internet and have it sync the clock to UTC while driving.

4.2.7 Testing 27.04.2021

On the prior test day, there was a problem with allocating the correct GPS points to the images. At the time, the system allocated time from the computer clock to the images when they were saved to the system. Before the images were saved to the computer, they were stored in a buffer. This led to the timestamp being inaccurate. In addition to this, the system was not connected to an NTP server, so the computer clock would drift and become more inaccurate over time.
The solution to the synchronization problem was to calculate a new timestamp with the computer timestamp and the camera clock timestamp. In addition to this, we set up a hot-spot on the phone and shared it with the system, synchronizing the computer clock with UTC time.
The synchronization of the camera clock to the computer clock goes as follows: We assume the time between capturing the first image to the image being saved is approximately zero(figure 7). Then, we subtract the new camera timestamp with the first camera timestamp and add it to the first computer timestamp 4.15.

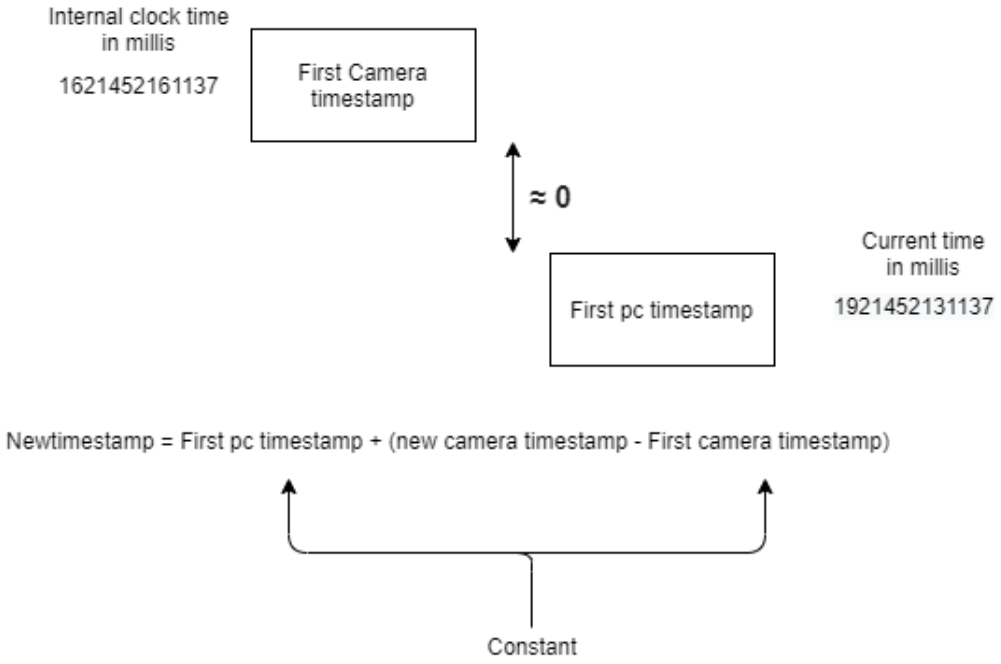


Figure 4.15: Calculating timestamp for images

On previous test days the tests were done on a relative short route with low traffic. This allowed us to make more tests and troubleshoot between each test, as there were still errors in the system. On this day, the test route lasted for 30 minutes, compared to 3-4 minutes on the previous

test days. The speed compared to prior test days was also increased, up to 80 km/h compared to 5-30 km/h on this one.

Test 1/marker test

- Set up a lot of markers
- Fps on camera regulated by speed

Test 2-slow speed

Accuracy test for GPS plotting. Drive slow to get more GPS points per picture

- Drive slow
- Fps on camera regulated by speed

Test results

Image capturing

GPS

Finding position of marker from image

On the first test, we had to take the ground truth position of the markers on the opposite side of the road due to heavy traffic. This will be taken into account as seen in figure 4.17.
For test 2 we had an average speed of 5m/s and could see that the highest error was 1,36m from ground truth. The error varied because of the position of the marker in the image as seen in figure 4.16. The less centered the marker is, the higher the error. The latter is due to the GPS receiver is located on top of the camera.

Figure 4.16: Comparing image position to ground truth

Ground truth	test1	Error m	image	time image	time GPS	lat	lon	speed m/s	Quality
lat	lon								
62.369563166666666	8.0402965	11,44	5373	1619541239753.0	1619541239800.0	62.369605166666666	8.040499	15.793444444444445	3
62.436300333333335	7.857232	6,97	9420	1619542167638.0	1619542167600.0	62.4363605	7.857269666666666	14.147222222222222	3
62.44713333333333	7.836808333333334	7,36	9965	1619542276753.0	1619542276800.0	62.447173	7.8369230000000005	15.896333333333333	3
62.4637855	7.819518	7,77	10521	1619542405798.0	1619542405800.0	62.463835833333334	7.819622833333334	16.050666666666668	2
62.47882883333333	7.767362833333333	9,16	11352	1619542611991.0	1619542612000.0	62.478904666666665	7.767432333333334	15.690555555555557	2
62.53927	7.725302666666667	9,09	12539	1619543122066.0	1619543122100.0	62.53934783333333	7.725357	16.050666666666668	3

Ground truth	test2								
lat	lon	Error m	image	tid bilde	tid gps	lat	lon		
62.323869333333334	8.080515666666667	1,36	1	1619539772930.0	1619539772900.0	62.32386016666667	8.080533	3.4467777777777777	3
62.323767833333335	8.080704333333333	0,50	7	1619539775931.0	1619539775900.0	62.32377	8.080695833333333	5.2987777777777785	3
62.323664666666666	8.0808855	0,60	15	1619539778583.0	1619539778600.0	62.323662666666664	8.080896333333333	6.379111111111111	3
62.30116983333333	8.120154166666667	1,25	3202	1619540025316.0	1619540025300.0	62.30116066666667	8.120140166666667	5.093	3
62.30105616666667	8.120284333333334	1,18	3230	1619540028344.0	1619540028300.0	62.30105266666666	8.120262833333333	4.012666666666667	3
62.3009945	8.120369166666666	0,82	3237	1619540030096.0	1619540030100.0	62.30099633333333	8.120353833333333	5.710333333333334	3
62.2981585	8.1226335	0,31	3559	1619540060231.0	1619540060200.0	62.2981585	8.1226275	5.2987777777777785	3
62.298011	8.122613166666667	0,38	3594	1619540065647.0	1619540065600.0	62.298008	8.1226165	4.527111111111112	3

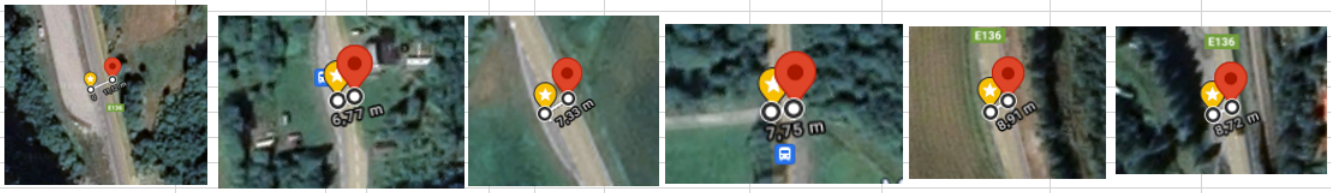


Figure 4.17: Comparing image position to ground truth

Issues to be resolved

Chapter 5

Result

This chapter shows what we have achieved in the project. First there is an overview of the system and how it works. Then, an explanation in detail of all the functionalities and how we developed the different subsystems.

5.1 System overview

Figure 5.1 shows simple overview the final system.

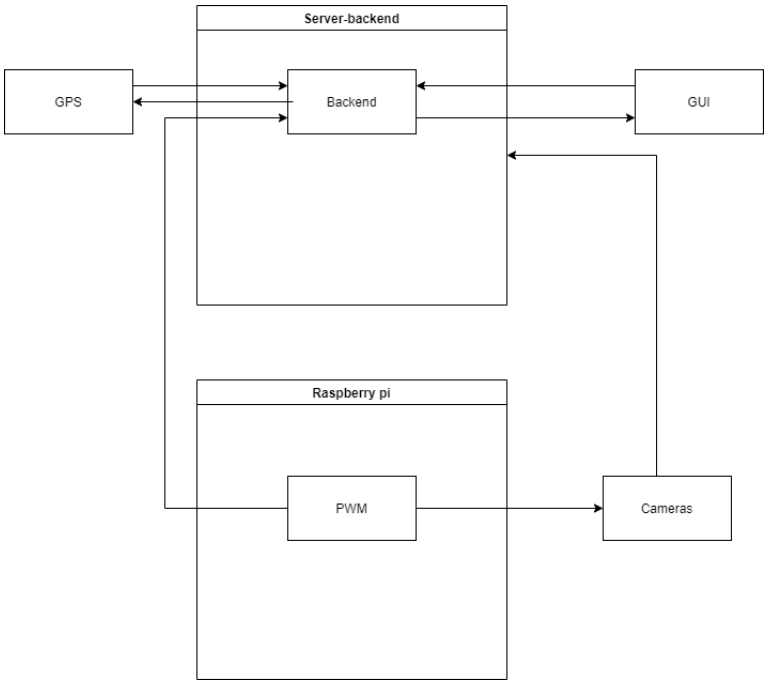


Figure 5.1: System-UML

The application consists of three main systems: the server/backend, the GUI, and the Raspberry Pi. The GUI is displaying information of the system and gives the operator a variety of possibilities like image capturing, mapping and real time video of the cameras. The server is responsible for handling requests from the GUI and the communication between the sub-processes that is GPS and camera streams.

5.2 GUI

This section shows the final results of the GUI and explains all functions developed.

5.2.1 Use-case

Figure 5.2 shows a use-case diagram of the application that describes what the operator can do.

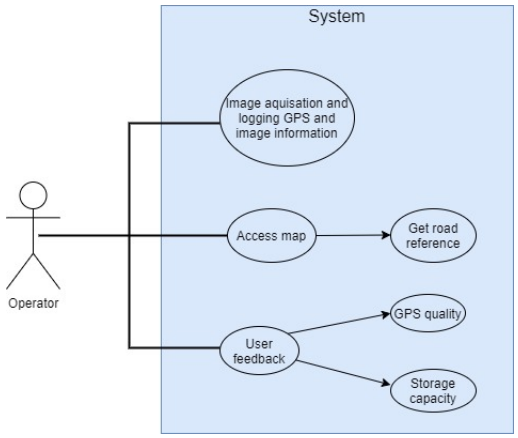


Figure 5.2: Case diagram

5.2.2 Deployment

The GUI is deployed on a Nginx web-server and is hosted on the computers integrated WiFi-card.The operator can access it by connecting to the WiFi-network and going to the computers WiFi-IP and port 3000.

5.2.3 Design

In the design process, it was essential to consider the objectives of making the GUI so simple that it can be operated by a user with no prior technical understanding and giving the operator valuable feedback of the system's current state. In the GUI, the operator can navigate between four pages. Each page has its own function and provides the operator with different tools. The choosing of design of these pages will be explained in this section.

Dashboard

The dashboard is the application's home page; this is where the operator gets the most critical feedback and can choose to start the data acquisition. The dashboard contains four blocks. The big block on top displays the system's overall status and gives feedback on the image capturing. The three other blocks give the operator information about GPS and storage. In addition to showing the different statuses with a text description, it also changes color indicating what status it has. The color status was previously discussed in method 3.4.1.

- Camera connection
- Socket connection to the server
- GPS status
- Storage status
- Image status

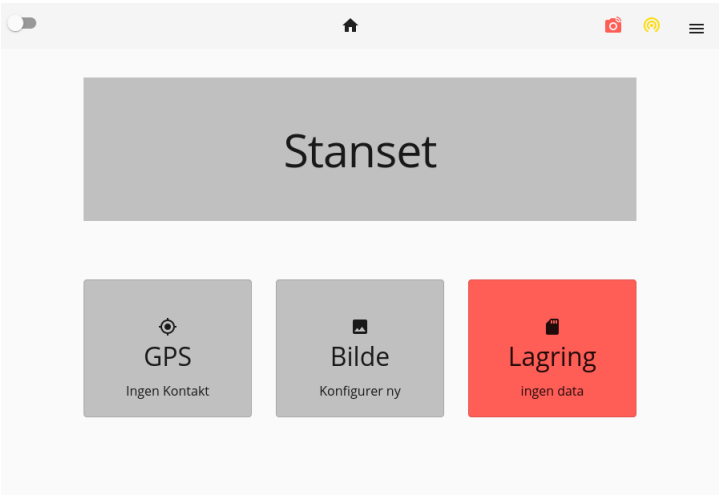


Figure 5.3: GUI not connected

Real time feedback

The operator gets real time feedback of GPS data, Storage and how many images captured on each camera. This gives the operator the opportunity to get more detailed data. When the operator clicks on the respective boxes a dialog will appear as showed in figures 5.4,5.5 and 5.6.

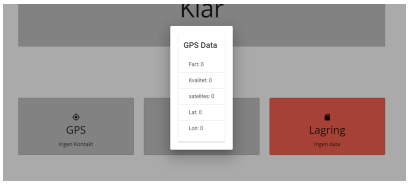


Figure 5.4: Shows GPS-data

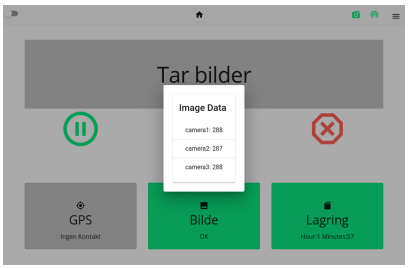


Figure 5.5: Feedback on how many images is acquired

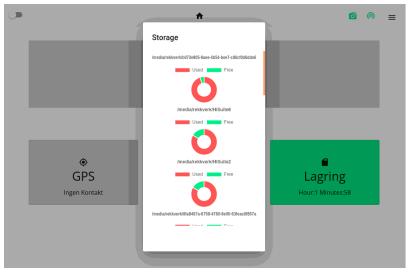


Figure 5.6: Shows an overview of storage

Stepper

When the operator clicks on "Configure new", the operator has to go through three steps before being able to start new data acquisition.

Check camera connection

The first step is ensuring the camera is successfully connected. The GUI sends an event to the Web-socket and the server replies with status of all three cameras. The operator can proceed to the next step if minimum one camera is connected.5.7

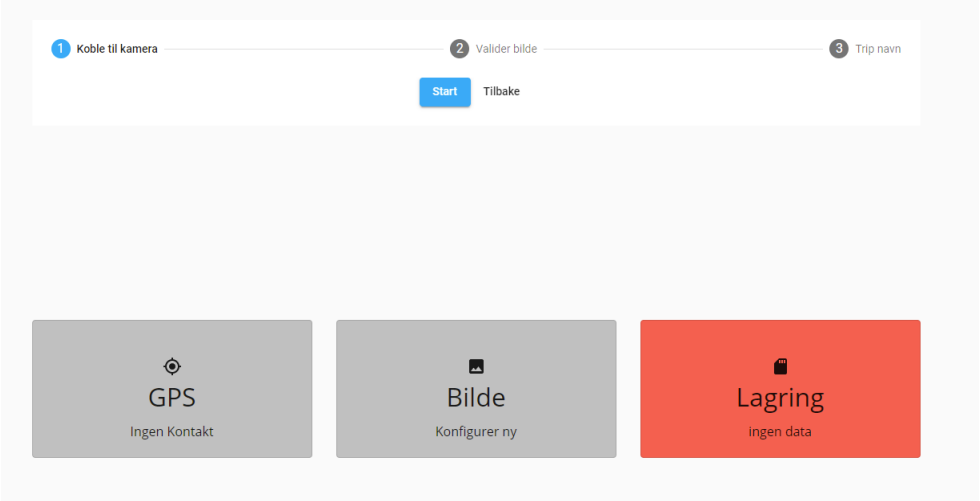


Figure 5.7: Type in name of trip

Validate Images

This is the second step, and last check for the operator to ensure that the images received from the camera have acceptable quality. When the operator has confirmed the quality of the images, it proceeds to the last step.5.8

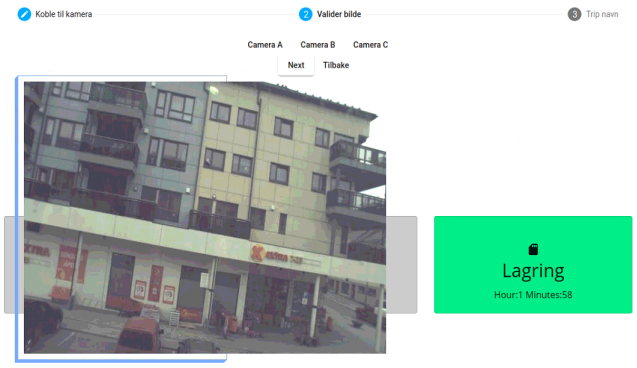


Figure 5.8: Validate image quality

Configure new trip

In the last step, the operator has to define a name for the new configured trip. This will be the name of the folders containing all the images and log files. When a name is defined, the operator can press start and the control panel will appear.

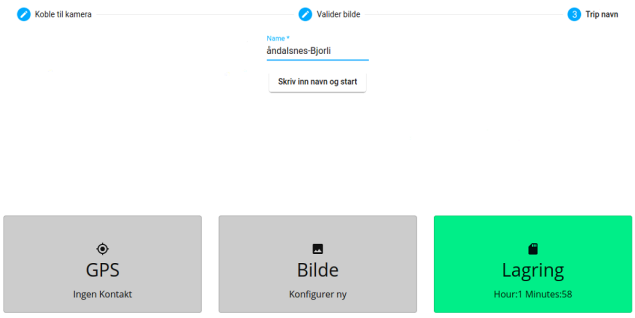


Figure 5.9: Type in name of trip

Data acquisition control panel

The control panel gives the operator the possibility to start and stop the data acquisition. Furthermore, it has an emergency stop button. This button immediately stops the data acquisition, compared to the stop button that is ensuring that all images are captured before safely stopping.

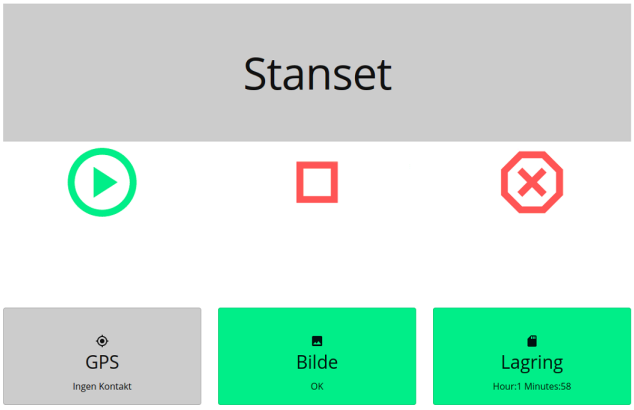


Figure 5.10: Control panel

Config-page

In the Config page, the operator can decide if the system will use a static FPS or use the FPS provided by the speed of the vehicle. When the static FPS is chosen, the operator can set it to a desired value. The page also has debug toggling that is meant for developers for debugging the software. It prints useful messages to the terminal where the program is running.

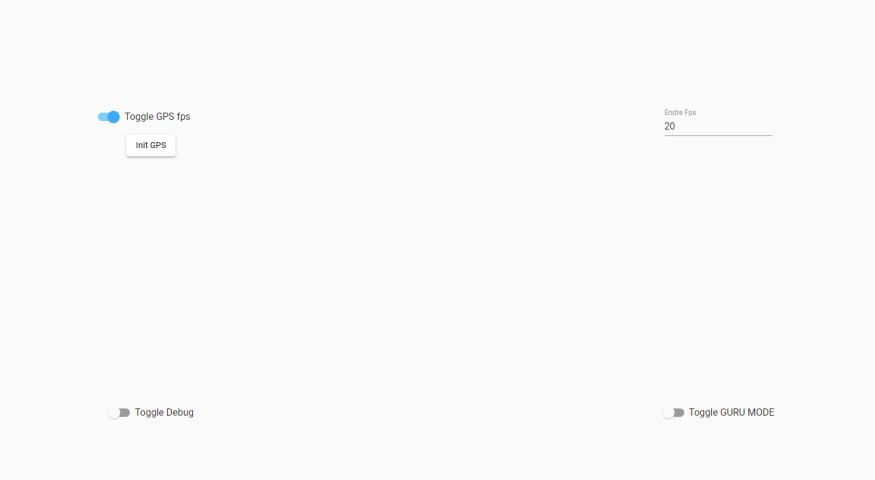


Figure 5.11: Config page

Map-page

The map-page is showing a detailed map to the operator as shown in figure5.12. It is meant as a planning-tool for planning the next route for collection of guard-rail images. The map has also other functionalities that will be explained in the next sections.

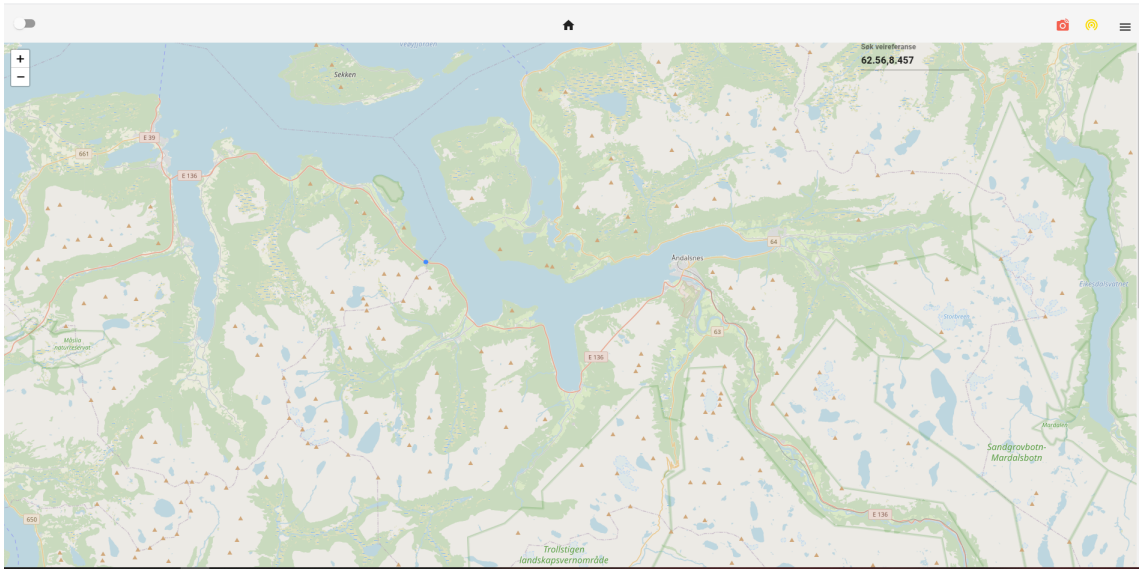


Figure 5.12: Map

Drawing recent trips on map

When the operator navigates the map-page, it will upload GPS-coordinates from the latest trips from the server and draw them on the map.

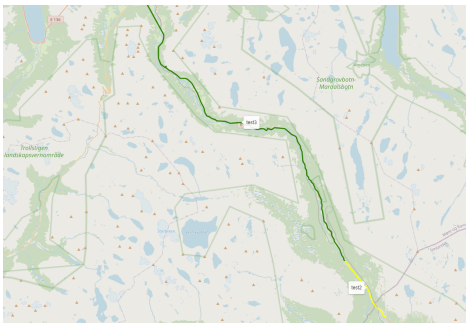


Figure 5.13: Visualisation of recent trips

Road reference

By clicking on a point on a road on the map, the operator can get the road reference.2.4.6. Getting the road reference is useful for naming trips or later mapping a reference to a guard rail fault.

Figure 5.14: Clicking on a road section gives you the reference

Search road reference with coordinates

The map also contains a search field where the operator can type in coordinates (lat,lon) to get a road reference. The map will then fit the map-view to the requested position.

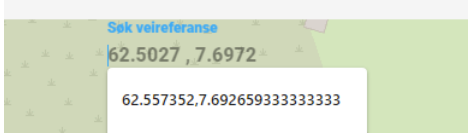


Figure 5.15: Search with coordinates to get road reference

Camera-page

This page shows you real-time video from the three cameras and gives the operator a clear view of the images and can check if the camera-lenses are dirty or need adjustment.

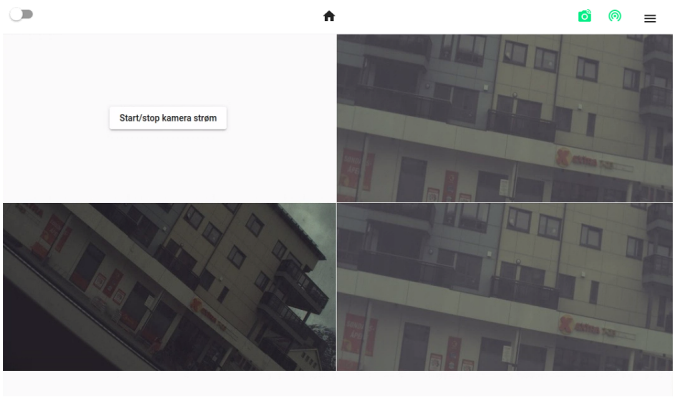


Figure 5.16: Camera view page

5.2.4 Software

The GUI is made as a single-page web application and is developed using Angular as the web framework. This section will further describe how we solved communication with the server and the essential functions needed for the GUI to work. Framework-specific details will not be discussed in this section.

Web-socket service

Is a service/class responsible for establishing a new web-socket connection to the server and reconnecting if the connection is broken. When the page is loaded or refreshed, it will make a new web-socket connection.

The method in figure 5.17 shows how it establishes the web-socket connection and forwards the messages to a Subject 3.5.3 where the subject is subscribed to in other parts in the GUI. It has a parameter reconnect, it can be set to true if it looses connection to the server and will try to reconnect every 2 seconds with the same socket.

```
public connect(cfg: { reconnect: boolean } = { reconnect: false }): void {

  if(cfg.reconnect){
    this.store.dispatch(websocket({ newStatus: 'Reconnecting' }));
  }
 if (!this.connection$ || this.connection$.closed ||this.connection$ == undefined) {


  this.connection$ = this.getNewWebSocket();

    const messages = this.connection$.pipe(
      cfg.reconnect ? this.reconnect  : (o: any) => o,
      tap({

        error: (error) => console.log(error),
      }),
      catchError((_) => EMPTY)
    )       You, 2 months ago • fix
    this.messagesSubject$.next(messages);
  }
}
```

Figure 5.17: Connect to websocket method

The method in figure 5.18 shows how the web-socket is created by connecting to the servers endpoint. For giving the connection a unique ID it uses Date.now() that returns number of milliseconds elapsed since January 1, 1970. The websocket object is configured with the endpoint URL and with actions when the connection is open and when it closes. When the connection closes, it passes the reconnect parameter and will try to reconnect 5.19.

```
  /**
   * Return a custom WebSocket subject which reconnects after failure
   */
  private getNewWebSocket() {
    console.log(WS_ENDPOINT + '/stream/' + Date.now());
    return webSocket({
      url: WS_ENDPOINT + '/stream/' + Date.now(),
      openObserver: {
        next: () => {
          console.log('[DataService]: connection ok');
        },
      },
      closeObserver: {
        next: () => {
          this.store.dispatch(websocket({ newStatus: 'Disconnected' }));
          console.log('[DataService]: connection closed');
          this.connection$ = undefined;

          this.connect({ reconnect: true });
        },
      },
    });
  }
```

Figure 5.18: Get new web-socket

Figure 5.19: reconnect web-socket

HTTP service

Is a service/class for making the HTTP-requests to the server. The HTTP requests are used to execute the longer running server tasks, because of the FastAPIs ability to run the endpoints in separate threads. The service is returning an Observable. The Observable is being subscribed to. And will process the responds when it arrives.

Figure 5.20: Snippet of HTTP-service

Starting image capturing loops

The starting of the image capturing loops is done by sending a GET-request to the server and subscribing for the returned value. When the image-capturing has been stopped by the operator, the server returns the request and the states of the system can be changed accordingly. Web-socket commands is then sent back to the server to change the server-state.

```
startStream(){
  this.http
  .start1()
  .pipe(takeUntil(this.http.stopstream$))
  .subscribe((results)=>{
    let payload = {stream:"stream1",results:results}

    this.websocketService.send({event:"log",data:payload})
  })
this.http
  .start2()
  .pipe(takeUntil(this.http.stopstream$))
  .subscribe((results)=>{

    let payload = {stream:"stream2",results:results}
    this.captureStatus ="Klar"
    this.store.dispatch(running({running:false}))
    this.websocketService.send({event:"reset",data:""})
    this.websocketService.send({event:"log",data:payload})
    this.merge()
  })


    this.http
    .start3()
    .pipe(takeUntil(this.http.stopstream$))
    .subscribe((results)=>{
      console.log(results)
      let payload = {stream:"stream3",results:results}

      this.websocketService.send({event:"log",data:payload})
    })
}
```

Figure 5.21: Starting and subscribing to the image capturing loops

Handling GUI-states

For managing the states of the system and keeping track of variables a global state manager, (NGRX store) is used. This is used because it enables sharing states between components in the software. The store contains the initial state and can be changed or subscribed to by dispatching new states or subscribing to changes of the current states.
the states are:

- Websocket connection
- Camera connection
- Image capturing (started/paused)
- Data acquisition started/stopped
- FPS mode (GPS or static)

- And more internal states

Figure 5.22: Dispatch new state

Figure 5.23: Select state from store

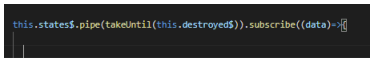


Figure 5.24: Subscribe to state

Every time the page is refreshed, or when the operator is navigating, the states are updated by sending a command to the server. The server then returns the states and the GUI updates its "store".

Map

The map page is developed using an external library Leaflet and is made of two components using the hierarchy from the Angular framework. One acts as a child and other as the parent. The child is responsible of initiating the map, attaching it to the DOM and emitting events 2 to the parent that handles the events.

```
<div
  class="map-container"
  leaflet
  [leafletOptions]="options"
  (leafletMapReady)="onMapReady($event)"
  (leafletMapZoomEnd)="onMapZoomEnd($event)"
  (leafletClick) ="onClick($event)"
></div>
```

Listing 2: DOM events Angular

Road reference

For getting the road reference 2.4.6 of a coordinate, the map is listening for a click event 2. When the user clicks on a desired road section the event is fired and is sending an event-object to the onClick() function 3.

```
onClick(e:any){

  let latlng = e.latlng

  this.coord$.emit(latlng)


}
```

Listing 3: onCLick

The latitude and the longitude is taken out of the event object and emitted through a "EventEmitter" 4 for passing the value to the parent component.

```
@Output() coord$: EventEmitter<any> = new EventEmitter();
```

Listing 4: Eventemitter Angular

The parent component is listening for the event 5. and is passing the received event to the function 6. This function sends a GET-request to the Vegdata API passing along the coordinate. The road reference is then marked on the map as showned in figure 5.14

```
<div class="fs-container">

  <form class="form">

    <mat-form-field class="example-full-width">
      <mat-label>Søk veireferanse</mat-label>
      <input (keydown.enter)="onEnter($event)" type="search"  matInput placeholder="62.5
    </mat-form-field>
  </form>

  <app-map (map$)="receiveMap($event)" (zoom$)="receiveZoom($event)" (coord$)="recieveC
</div>
```

Listing 5: Parent component Angular

5.3 Server/backend

This section shows the final results of the backend software and is explaining important components in the final solution.

5.3.1 Deploying

The server/backed depends on multiple libraries and packages to run. For installing all dependencies a requirements.txt file is created. This file describes what packages and which version to install. Furthermore, the CVB SDK have to be installed on the computer for interfacing the cameras. It is good practice to install all the dependencies in a virtual environment, which is described in the GitHub repository.
By executing app.py, the script will start the server,GPS thread and the three camera processes.

5.3.2 Server

The server is a REST API that clients (GUI) can send HTTP GET, PUT and POST requests for receiving, processing or changing data on the server. Furthermore, it is handling a web socket connections to the client. The REST API is used for executing long running tasks and the websocket is used for doing small fast commandos sent from the client like setting or getting state variables.

```
recieveCoord(latlong:any){


  this.http.getRoadReference(latlong).subscribe((data)=>{


    let section = data[0].vegsystemreferanse.kortform

   let newmarker = circleMarker([latlong.lat,latlong.lng],{radius:2}).bindPopup(sectio

    newmarker.openPopup()
     newmarker.addEventListener("click",function(event){

      newmarker.remove()

     })


  })
```

Listing 6: Get road reference

Communication

Web-socket Handler

The web-socket handler controls connection, disconnection and events from multiple clients. The handler uses a Manager class for storing all connected clients and can choose to broadcast or send messages to specific clients. Figure 5.25 shows a snippet of how we are defining a web-socket endpoint and handling events from the GUI.

```python
@app.websocket("/stream/{client_id}")
async def websocket_endpoint(websocket: WebSocket, client_id: int):
    global started,config_loaded,isConfigured,gps,drive_in_use,gpsControl,valider,tempTrip,guruMode
    await manager.connect(websocket)
    await websocket.send_text(json.dumps({"event": "connected", "data": "connected to server"}))
    try:
        while True:
            data = await websocket.receive_text()
            data = json.loads(data)
            event = data['event']
            msg = data['data']
                You, seconds ago • Uncommitted changes



            if(event == "onConnection"):

                await manager.broadcast(json.dumps({"connection": "connected"}))




            elif(event == 'start'):

                isConfigured = True
                tempTrip = str(msg)
                gps.setTripName(str(msg))
                drive_in_use = await createImageFolder(msg)
                if drive_in_use =="failed":
                    await manager.broadcast(json.dumps({"event": "error","data":"no drives"}))
```

Figure 5.25: Web-socket handler

HTTP

Figure 5.26 shows how we are creating an endpoint for the GUI. The server is handling the request and is returning a payload to the GUI. In this case, storage information.

```python
@app.get('/storage')
async def getStorage():
    global storage,started,drive

    storages =checkStorageAllDrives()
    drive.value = most_free_space()      You, 4 days ago • fdfd



    payload = estimateStorageTime(storages,10)

    total = payload['total']['free']
    if int(total)<5 and started:
        emergencyStop()
        await manager.broadcast(json.dumps({"event":"storage","data":"empty"}))



    return payload
```

Figure 5.26: Example of endpoint

5.3.3 GPS data

The GPS program is running in a separate thread and is developed to be compatible with any GPS that outputs NMEA-sentences on the serial-port. It is also responsible for filtering GGA and RMC-sentences for the server and creating a GPS-log of the trip. This section explains the features of the GPS-program.

Automatic discover and connection of receiver

The program automatically discovers the GPS-receiver connected to the serial-port of the com-

puter. This is done by opening serial-ports one by one and reading the data. If the input of the serial port does not contain NMEA-sentences, it closes the port and try the next port. Subsequently, it checks for NMEA-sentences by parsing the data using the Python library Pynmea2. If the data parsed with PyNMEA2 does not contain NMEA-sentences, it will throw an exception.

```
try:
    while True:
        ports = self.scan_ports()
        if len(ports) == 0:
            if self.debug:
                sys.stderr.write('No ports found, waiting 5 seconds...press Ctrl-C to quit...\n')
            time.sleep(5)
            continue

        for port in ports:
        # try to open serial port
            if self.debug:
                sys.stderr.write('Trying port %s\n' % port)
            try:
        # try to read a line of data from the serial port and parse
                with serial.Serial(port, 115200, timeout=10) as ser:

                    self.serial = ser
                    self.read = io.TextIOWrapper(io.BufferedRWPair(ser, ser))
                      #sends commands to gps
                    if self.init:
                        self.initGPS()
                        self.init = False
        # 'warm up' with reading some input
                    for i in range(10):
                        data = ser.readline()
                        #print(data)
                    # try to parse (will throw an exception if input is not valid NMEA)

                    pynmea2.parse(ser.readline().decode('ascii', errors='replace'))
```

Figure 5.27: Snippet of GPS connection function

```
#https://fishandwhistle.net/post/2016/using-pyserial-pynmea2-and-raspberry-pi-to-log-nmea-output/
def scan_ports(self):

    if sys.platform.startswith('win'):
        ports = ['COM%s' % (i + 1) for i in range(6)]
    elif sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):
        # this excludes your current terminal "/dev/tty"
        patterns = ('/dev/tty[A-Za-z]*', '/dev/ttyUSB*')
        ports = [glob.glob(pattern) for pattern in patterns]
        ports = [item for sublist in ports for item in sublist]  # flatten
    elif sys.platform.startswith('darwin'):
        patterns = ('/dev/*serial*', '/dev/ttyUSB*', '/dev/ttyS*')
        ports = [glob.glob(pattern) for pattern in patterns]
        ports = [item for sublist in ports for item in sublist]  # flatten
    else:
        raise EnvironmentError('Unsupported platform')
    return ports      You, 2 months ago • gps klasse
```

Figure 5.28: Scan ports function

Initializing of the GPS

To initialize the GPS with the correct settings, we wrote a series of commands to the serial-port 5.29. These commands connected the GPS to Kartverkets CPOS-server to receive corrected

position-data and to specify which NMEA-sentences to be outputted trough the serial-port.

```
def initGPS(self):
    self.initStarted =True
    self.send('SetNMEAOutput, Stream1+Stream2+Stream7+Stream8, none, none, off\r\n')
    self.send('SetGPIOFunctionality, GP1, Output, none, LevelLow\r\n')
    self.send('SetGPIOFunctionality, GP2, Output, none, LevelHigh\r\n')
    self.send('SetGPIOFunctionality, GP3, Output, none, LevelHigh\r\n')
    self.send('setDataInOut, COM1,DC1,DC2\r\n')
    self.send('setDataInOut, COM3,DC2,DC1\r\n')
    self.send('#PSPO,1\n')
    self.send('SSSSSSSSSS\r\n')
    self.send('setDataInOut, COM1, CMD, SBF+NMEA\r\n')
    self.send('SSSSSSSSSS\r\n')
    self.send('SetSBFOutput,Stream1+Stream2+Stream3+Stream4+Stream5,none,none,off\r\n')
    self.send('setPVTMode, Rover, all\r\n')
    self.send('sem,+PVT,10\r\n')
    self.send('setDataInOut, COM1, CMD, SBF\r\n')
    self.send('setGeoidUndulation, auto\r\n')
    self.send('setSmoothingInterval,all,100,1\r\n')
    self.send('setRAIMLevels,on,-4,-4,-3\r\n')
    self.send('setPVTMode, Rover, all\r\n')
    self.send('setFixReliability, RTK, 0.2, 4.40\r\n')
    self.send('setDiffCorrUsage,,,auto,0\r\n')
    self.send('setDataInOut, COM2, RTCMv3, SBF+NMEA\r\n')
    self.send('setDataInOut, COM2,DC1,DC2\r\n')
    self.send('setDataInOut, COM3,DC2,DC1\r\n')
    self.send('+++\r\n')
    self.send('AT\r\n')
    self.send('+++\r\n')
    self.send('AT\r\n')
    self.send('+++\r\n')
    self.send('ATE1\r\n')
    self.send('AT+MGEER=2\r\n')
    self.send('AT+CMEE=2\r\n')
    self.send('AT+CBST?\r\n')
    self.send('AT+CPIN?\r\n')
    self.send('AT+CSQ\r\n')
    self.send('AT+MIPCALL?\r\n')
    self.send('AT+MIPCALL=1,"telenor","","\r\n')
    self.send('AT+MIPOPEN?\r\n')
    self.send('AT+MIPODM=1,1200,"159.162.103.14",2101,0\r\n')
    self.send('GET /CPOSHREF HTTP/1.0\r\n')
    self.send('User-Agent: NTRIP Altus\r\n')
    self.send('Authorization: Basic NTgwMDAwNzEzNTMzOkdKRVJERTEzMDU=\r\n')
    self.send('Accept: */*\r\n')
    self.send('Connection: close\r\n')
    self.send('\r\n')
    self.send('\r\n')
    self.send('SSSSSSSSSS\r\n')
    self.send('setDataInOut, COM3, CMD, SBF+NMEA\r\n')
    self.send('setDataInOut, COM2, RTCMv3, SBF+NMEA\r\n')
    self.send('SetNMEAOutput, Stream1, COM2, GGA, sec1\r\n')
    self.send('SetNMEAOutput, Stream8, COM3, GGA+VTG+RMC, msec100\r\n')

    self.initStarted = False
```

Figure 5.29: Serial write commands for initializing the GPS

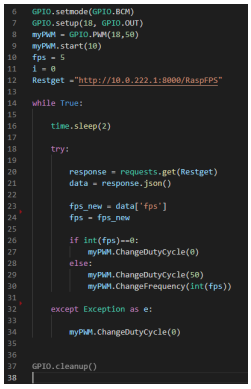
5.3.4 Adaptive image capturing rate

For controlling the FPS of the cameras, two solutions were developed for making the system more versatile.

Raspberry Pi

The Raspberry Pi is connected to the same network as the main computer and is communicating with the server trough HTTP GET-requests and is providing the cameras with PWM for controlling the FPS. The PWM is set to a duty-cycle of 50% and the frequency is controlled by the response of the GET-requests from the server. The response is a Json that contains a integer

value representing the current speed of the vehicle, or a fixed speed. RPi.GPIO library is used for controlling the GPIO of the Raspberry Pi.



```
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
myPWM = GPIO.PWM(18,50)
myPWM.start(10)
fps = 5
i = 0
Restget ="http://10.0.222.1:8000/RaspFPS"

while True:

    time.sleep(2)

    try:

        response = requests.get(Restget)
        data = response.json()

        fps_new = data['fps']
        fps = fps_new

        if int(fps)==0:
            myPWM.ChangeDutyCycle(0)
        else:
            myPWM.ChangeDutyCycle(50)
            myPWM.ChangeFrequency(int(fps))

    except Exception as e:

        myPWM.ChangeDutyCycle(0)


GPIO.cleanup()
```

Figure 5.30: Controlling frequency of pulses

Running program as service

To make the program start when the Raspberry Pi boots, the program is added as a service. The procedure of adding a program as a service is described in the cited article [29]

Arduino

The Arduino is communicating over a serial connection with the computer. The server sends the required FPS to the Arduino and the new frequency is computed. The program 5.32 on the Arduino is using the built in register timer. By default, the timer counts from 0 to 255 in a frequency of 16MHz. By adjusting the top limit of the timer, we can control the duty cycle, and for the frequency, we set the adjust the value of the counter.See figure 5.31.

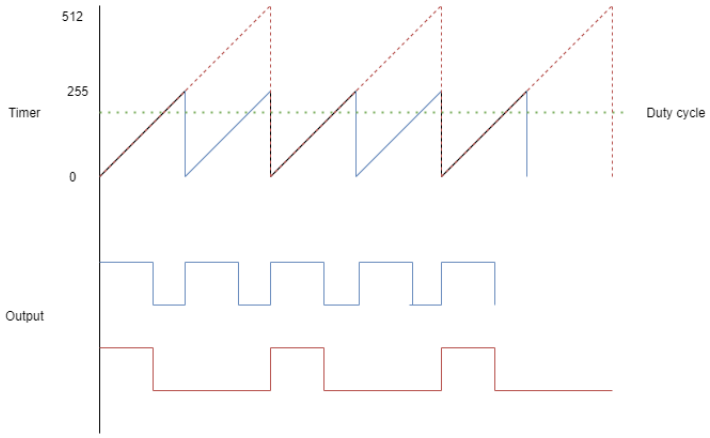


Figure 5.31: Adjusting Register timer values

```
int FPS = 10;
//https://www.arduino.cc/en/Tutorial/SecretsOfArduinoPWM
void setup() {
  pinMode(9, OUTPUT);                          // Set digital pin 9 (D9) to an output
  TCCR1A = _BV(COM1A1) | _BV(WGM11);           // Enable the PWM output OC1A on digital pins 9
  TCCR1B = _BV(WGM13) | _BV(WGM12) | _BV(CS12);  // Set fast PWM and prescaler of 256 on timer 1
  ICR1 = 12499;                                // Set the PWM frequency to 5Hz: 16MHz/(256 * 5Hz) - 1 = 12499
  OCR1A = 1249;                                // Set the duty-cycle to 10%: 62499 / 10 = 1249

  Serial.begin(9600);                          //listen to serial port
}


void loop() {
if(Serial.available() > 0)  {
    int incomingData= Serial.read();
    Serial.print(incomingData);
    int frequency = incomingData;
    int FPS =  16000000/(256*frequency)-1; // calculate new FPS
    ICR1 = FPS;
    OCR1A = FPS/10;

}

}
```

Figure 5.32: Controlling the frequency of pulses

5.3.5 Data acquisition

Image capturing without loss

In order to acquire images from the camera, we made the "CameraStream" class that runs in a separate process. See testing results 4.3. The program has, therefore, three separate processes that are responsible of acquire an image from the camera and put in a queue2.1.3. Each process also has its own thread responsible for getting images from the queue and storing it to the SSD/HDD A. The camera stream class is made as a subclass of Process, see figure 5.33. And by overriding the run() method, we can start the process by calling start() on the camera stream object.

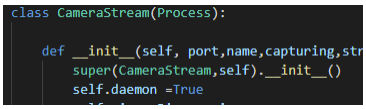


Figure 5.33: Snippet of the camera stream class

Storing Images

The images are automatically being stored on the external hard drive with the most free space. Before each start of a trip, the software is checking all the hard drives connected to the computer, and then, choosing the hard drive with the most free space. This is done because we wanted to give the operator less options to care about.
Every two seconds, the application is checking the total external storage of the computer. A function is finding the drive with the most free storage space and then, updating the image-storing directory7.

Mapping each image with GPS-position

The GPS data and image information are logged to get the image location. For the image, the timestamp and image number are logged. For the GPS timestamp, fix, latitude, and longitude are logged. The timestamp of the image and GPS are used for finding the coordinates of the images. The image number is used for labeling the images. The fix is logged to see how accurate the GPS coordinates are.

Getting correct image timestamp

Synchronising the camera clock and the GPS clock is essential for mapping the GPS coordinate to the correct image. The GPS gives time in UTC, and the camera have an internal clock that represents the time since last reset.

We then assume that the time from when the first image is taken to when it is acquired by the software is approximately zero, seen in figure 4.15. The rest of the timestamps of the images are

calculated by using the first computer timestamp, the first camera timestamp and the current camera timestamp. Figure 4.15 shows the formula used to calculate new image timestamps. For ensuring that the time difference between GPS and computer is close to zero, we had the computer connected to the internet so it synchronises its clock to UTC time.

Finding closest corresponding coordinate to image

The system is designed to acquire thousands of images each run. Comparing every element together in for loops to find the closest matching timestamp would not be a solution. We chose to use the bisect library for comparing. Using the bisect library to compare significantly decrease the time shown in 4.5.

The GPS gives its position every 100 ms; therefore, difference between the image and its closest matching GPS timestamp should have a -50ms ≥ time difference ≤ 50ms. The comparing method worked as intended, as shown in 4.4

The GPS provides the location every 100 ms, which can lead to a time difference of up to 50 ms, see figure 5.34.

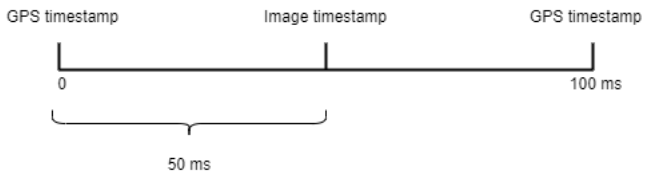


Figure 5.34: Time difference

Calculating new GPS-position of image

The calculation of the image's position happens between two GPS points, where we only have access to start and end speed, and constant acceleration. The only value for calculation from the images is the time difference between the image timestamp and the closest GPS timestamp. Compensating for the time difference, the image position is calculated. Figure 5.35 shows the calculation of the images location.

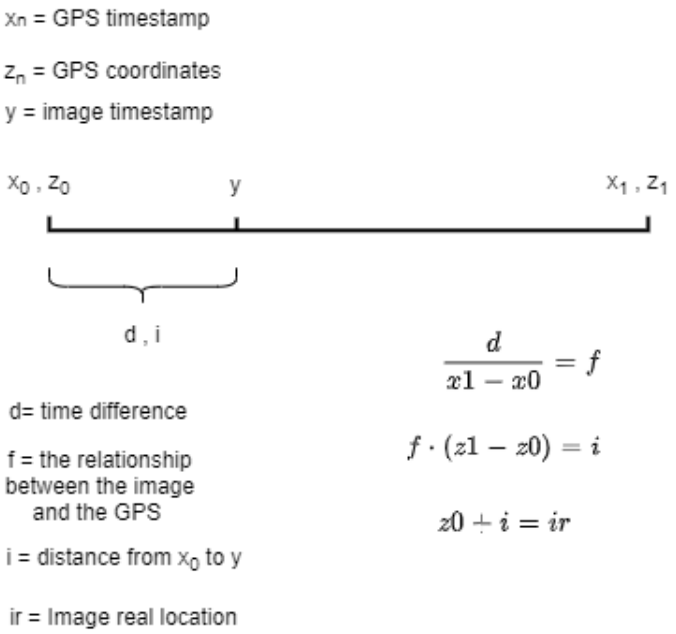


Figure 5.35: Calculating position of images

.

```
def checkStorageAllDrives():
    available_drives = []
    nested_storage= {'drives':{} }


    if sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):

        username = getpass.getuser()

        path = "/media/"+username
        directory_contents = os.listdir(path)

        for item in directory_contents:
            available_drives.append(path+"/"+str(item))



    elif sys.platform.startswith('win'):
        for d in string.ascii_uppercase: # Iterating through the english alphabet
            path = '%s:' % d
            if os.path.exists(path): # checks if path exists
                available_drives.append(path)  #  append the path from a drive to a list




    try:

        number= 1 # Variable for
        for i in available_drives:  #  Iterating through all drives

            total, used, free = shutil.disk_usage(i) # Return disk usage statistics abou

            dictname = "hd"+ str(number)
            nested_storage['drives'].update ({dictname:{ "name": i,"total": "%d" % (tota
            , "used": "%d" % (used // (2**30)), "free": "%d" % (free // (2**30))}})  #
            number += 1



    except:
        pass

    finally:

        return nested_storage
```

Listing 7: Check storage space

Chapter 6

Discussion

In this chapter we address different perspectives of the work and tests we have done during this project. We also provide our thoughts regarding the results and the development of the components that makes up the final system.

6.1 GUI

This section tackles different aspects regarding the design and functionality of the GUI.

6.1.1 Design

The impression we had with the GUI was that it was to be used on a tablet mounted to the dashboard of the vehicle. The operator driving the car could then gather images of guard railings and collect data with the help of a few finger-touches on the tablet. So, we kept this vision in our minds throughout the developing process of the GUI. One of the requirements from the client was that it could be used while driving and operated by a person without any technical background. Hence, it had to have a simple interface without any unnecessary elements that could distract the operator.

The user experience of GUI was not sufficient tested because of limited time used on full scale testing the system. Thus, it is hard to tell if a person without any technical understanding of the system could successfully operate the GUI. The main priorities during the full scale testing

was to get all basic functionality working, and there was no time trying to get an person outside the project to operate the GUI.The only feedback we got was from the client. Although they had been using the system as much as we had, they were happy with the result.

6.1.2 Functionality

When developing the functionalities of GUI, we tried to give the operator the tools needed for simplifying guard rail inspection and creating a reliable communication with the server. We developed the GUI as a web page and could, therefore, mainly focus on the design of the GUI, since the important logic is happening on the server. The only complication we had with GUI was handling all states in a efficient way. This was repaired by using a global state manager NGRX store. We did not had much experience with the tool, so there are still a few bugs, such as the operator not getting the required information or the GUI is freezing in a certain state. The solution to the problem is to refresh the page and hope the server-logic works and provides the actual state.
For the other functionalities like the map and getting the road reference it requires a internet connection. Connecting the computer to the internet and forwarding it to the GUI is not implemented in the system and is, therefore, a tool for future development.

6.2 Server/Backend

This section discusses the solutions we made and the process of making the final result of the server/backend software.

6.2.1 Image capturing without loss

One concurring problem to be solved during the project was acquiring images from multiple cameras without loss. The main issue leading to the problem is the lack of speed in the chosen programming language. We attempted to optimize the speed of the image acquisition as mentioned in the testing 4.1.1. And, in the final results we accomplished to acquire images from the

three cameras at a rate of 20 images per second. This means that the guard rail inspection can be done in a speed of 70kph without loosing images. The CVB-API offered developing in a more low-level programming language like in C++, this may have resolved our speed issues, but due to the lack of time and experience, we chose Python.

When trying to optimize the image capturing we used new techniques that went at the expense of loosing some of the functionalities of the GUI. The functionality that was lost due to different methods of acquiring an image was the validation step in the configure stepper 5.2.3.This was because we did not count with enough time of the project period for changing the method sending and image to the GUI.

6.3 Mapping each image with a GPS-position

The main concern from capturing images we had was to map each image with a GPS-position. This was improved by capturing GPS-data every tens of a second and the timestamp of the captured image. This solution, relied on a accurate timestamp from the camera, for correctly mapping the coordinate to the image. Issues and experience with this task are discussed in this section.

6.3.1 Getting correct image timestamp

The camera clock needs to be synchronised with the GPS clock to get the correct image timestamp. The current solution relies on the assumption that the difference of time when the first image is captured and the time when the image is saved in the system is approximately zero, as shown in figure 4.15. Even a slight delay would lead to a decrease in the accuracy of the image positioning. Another disadvantage with using the computer clock for synchronisation is that it is reliant on internet connection to update its clock to UTC. To solve the two problems and get the most accurate image timestamp possible, one could synchronise the camera clock directly with the GPS clock.

6.3.2 Finding closest corresponding coordinate to image

In the project, we logged the GPS data and image information to get the locations of the images. When finding the closest corresponding timestamp, we used the bisect library as it is much faster than comparing every element together, as shown in Figure 4.1.2.
in the Method 2, the result would vary slightly depending on the computer's CPU power where the test was conducted. This is because for loops are heavy on the CPU while the bisect method is not. The latter is shown in figure 6.1, where the same test was conducted on two computers with different processing power. The bisect method time stayed nearly the same, while the for-loops decreased with a better CPU.

	CPU processing power (seconds)	> CPU processing power (seconds)
Bisect method	2.9	2.925
For loops	35	45

Figure 6.1: CPU computation speed difference

The test was conducted on one of the author's personal computer, which has an inferior CPU compared to the system's CPU. If tested on the system's computer, the time difference would have decreased, as discussed in the paragraph above. But the time difference would still be significant.

6.3.3 Calculating new GPS-position of image

In the Marker test the accuracy of the calculation of image position was tested. The ground truth was measured on the opposite side to the markers on the road. Although we cannot test if the images are given its real location, we can still see if the calculated images position is an improvement to the GPS position. We used six markers in the test. If we had increased the number of markers, we would have a better result for verifying the method.

The accuracy of the image location depends on the fix quality and the clock synchronisations. With an improved GPS, we would consistently get a better fix, as it would not be as dependent on good weather conditions. An improved GPS would also give coordinates more frequently,

leading to better accuracy.

Chapter 7

Conclusions

The aim of the report was to develop an application for data capturing to be used in fault detection of guard rails. The requirements of the application was that it should consist of a graphical user interface, as well as have the ability to capture and store images without loss and to map each image with an GPS-coordinate. The specification of the GUI was making it as user-friendly that it could be employed and manipulated while driving and be operated by a person with no prior technical understanding. Furthermore, it had to give the operator valuable feedback.

We attempted to design a GUI with a simple interface and without any unnecessary elements that could distract the operator. We also seeked to give the operator some nice-to-have features that could assist the user when doing road rail inspection. When testing the application, and therefor the GUI, it was not used by people outside the project; hence, it could not conclude that the GUI is fulfilling one of the specifications.

One of the requirements of the application was acquiring and storing images from three cameras without loss. The application is capable of doing so at a maximum rate of 57 images/s or a speed of 70kph. This was due to limitations in computation speed of the software. Using a low-level programming language like C++ may have increased the speed and removed the limitation of 57 images per second. However, the client was happy with the result of the speed with which the data acquisition can be done.
One additional requirement of the application was to map a GPS-position to each image. This

was achieved, but the accuracy of the image positioning could be improved with a better GPS and with direct synchronisation of the camera and the GPS clock.

Finally, even though there is still room for improvement, the group deem the project a success. Our system is not a finished product as there is still some work and testing to do before it is ready for commercial use. Considering the project was intended for a group size of three members, both the client and us are satisfied with what we managed to achieve.

Bibliography

[1] pynmea2. URL https://github.com/Knio/pynmea2.

[2] Gpio. URL https://www.raspberrypi.org/documentation/usage/gpio/.

[3] Trilateration: How distance measurements help to work out location. URL https://www.oxts.com/trilateration/.

[4] Angular. What is angular. URL https://angular.io/guide/what-is-angular.

[5] Ben Croston. A module to control raspberry pi gpio channels. URL https://pypi.org/project/RPi.GPIO/.

[6] RXJS dev. Overview. URL https://rxjs.dev/guide/overview/.

[7] Python docs. multiprocessing — process-based parallelism, . URL https://docs.python.org/3/library/multiprocessing.html.

[8] Python docs. Multiprocessing.pipe, . URL https://docs.python.org/3/library/multiprocessing.html?highlight=pipe#multiprocessing.Pipe.

[9] Python docs. queue — a synchronized queue class, . URL https://docs.python.org/3/library/queue.html.

[10] docs NGRX. Ngrx introduction. URL https://ngrx.io/docs.

[11] docs.python. bisect — array bisection algorithm, . URL https://docs.python.org/3/library/bisect.html.

[12] docs.python. csv — csv file reading and writing, . URL https://docs.python.org/3/library/csv.html.

[13] docs.python. time — time access and conversions, . URL https://docs.python.org/3/library/time.html.

[14] FastAPI.com. Overview. URL https://fastapi.tiangolo.com/.

[15] Eric Gakstatter. what is nema-data. URL https://www.gpsworld.com/what-exactly-is-gps-nmea-data/.

[16] JANET HEATH. Pwm: Pulse width modulation: What is it and how does it work?

[17] Stemmer Imaging. Cvb camerasuit, . URL https://www.stemmer-imaging.com/en/products/series/cvb-camerasuite/.

[18] Stemmer Imaging. Common vision blox - the machine vision operating system, . URL https://www.stemmer-imaging.com/en/products/category/common-vision-blox-the-machine-vision-operating-system/.

[19] Stemmer imaging. Ptp gigecam. URL https://help.commonvisionblox.com/GenICam-User-Guide/html_english_cvb_genicam_usecases_ptp_for_gige.htm.

[20] Kartverket. Cpos. URL https://www.kartverket.no/til-lands/posisjon/hva-er-cpos.

[21] Leaflet. Leaflet docs. URL https://leafletjs.com/reference-1.7.1.html.

[22] Prof. Nitin R. Chopde Mr. Mangesh K. Nichat. Landmark based shortest path detection by using a* and haversine formula. URL https://www.researchgate.net/profile/Mangesh-Nichat-2/publication/282314348_Landmark_based_shortest_path_detection_by_using_A_Algorithm_and_Haversine_Formula/links/56389bb708ae4bde5021b0f5/Landmark-based-shortest-path-detection-by-using-A-Algorithm-and-Haversine-Formula.pdf.

[23] Lars Mæhlum. Rtk (real time kinematic). URL https://snl.no/RTK_-_Real_Time_Kinematic.

[24] Genie Nano. Camera user's manual. URL https://www.stemmer-imaging.com/media/uploads/cameras/dalsa/12/122239-Teledyne-DALSA-Genie-Nano-Series-Manual.pdf.

[25] Nginx.com. What is nginx. URL https://www.nginx.com/resources/glossary/nginx/.

[26] Store norske leksikon. Gnss. URL https://snl.no/GNSS.

[27] NRK.no. Her er autovern-feilene som rammer igjen og igjen. URL https://www.nrk.no/norge/xl/her-er-autovern-feilene-som-rammer-igjen-og-igjen-1.14463706.

[28] python docs. threading — thread-based parallelism. URL https://docs.python.org/3/library/threading.html.

[29] raspberrypi.org. Creating a service. URL https://www.raspberrypi.org/documentation/linux/usage/systemd.md.

[30] Linda Rosencrance. Tcp/ip (transmission control protocol/internet protocol). URL https://searchnetworking.techtarget.com/definition/TCP-IP.

[31] W3 school. Numpy intro. URL https://www.w3schools.com/python/numpy/numpy_intro.asp.

[32] Ethan Scully. Python vs c++: Dynamic and static. URL https://careerkarma.com/blog/python-vs-c-plus-plus/.

[33] Tutorialspoint. Hypertext transfer protocol. URL https://www.tutorialspoint.com/http/http_overview.

[34] Statens Vegvesen. Referansesystemet i nvdb, . URL https://www.vegvesen.no/fag/teknologi/nasjonal+vegdatabank/vegreferansesystem.

[35] Statens Vegvesen. *Referansesystemet i NVDB fra 2020*, . URL https://www.vegvesen.no/fag/teknologi/nasjonal+vegdatabank/vegreferansesystem/_attachment/2912425?_ts=17859577498&fast_title=faktaarket.

[36] wikipedia. Websocket. URL https://en.wikipedia.org/wiki/Bluetooth.

[37] Wikipedia. Css, . URL https://en.wikipedia.org/wiki/CSS.

[38] Wikipedia. Client–server model, . URL https://en.wikipedia.org/wiki/Client%E2%80%93server_model.

[39] Wikipedia. Git, . URL https://en.wikipedia.org/wiki/Git.

[40] wikipedia. Hypertext transfer protocol. URL https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol.

[41] Wikipedia. Markup language, . URL https://en.wikipedia.org/wiki/Markup_language.

[42] Wikipedia. Master/slave (technology), . URL https://en.wikipedia.org/wiki/Master/slave_(technology).

[43] Wikipedia. Network time protocol, . URL https://en.wikipedia.org/wiki/Network_Time_Protocol.

[44] Wikipedia. Opencv, . URL https://en.wikipedia.org/wiki/OpenCV.

[45] Wikipedia. Precision time protocol, . URL https://en.wikipedia.org/wiki/Precision_Time_Protocol.

[46] WikiPedia. Representational state transfer. URL https://en.wikipedia.org/wiki/Representational_state_transfer.

[47] wikipedia. Serial communication. URL https://en.wikipedia.org/wiki/Serial_communication#:~:text=In%20telecommunication%20and%20data%20transmission,link%20with%20several%20parallel%20channels.

[48] Wikipedia. International atomic time, . URL https://en.wikipedia.org/wiki/International_Atomic_Time.

[49] Wikipedia. Typescript, . URL https://en.wikipedia.org/wiki/TypeScript.

[50] wikipedia. Coordinated universal time. URL https://en.wikipedia.org/wiki/Coordinated_Universal_Time.

[51] Wikipedia. Visual studio code. URL https://en.wikipedia.org/wiki/Visual_Studio_Code.

[52] wikipedia. Websocket. URL https://en.wikipedia.org/wiki/WebSocket.

Appendix A

Appendices

A Preproject report

FORPROSJEKT - RAPPORT

FOR BACHELOROPPGAVE

NTNU
Kunnskap for en bedre verden

TITTEL:
Automatisk brukervennlig datainnsamling fra bil i bevegelse for inspeksjon av rekkverk

KANDIDATNUMMER(E):			
DATO: **14.1.2021**	EMNEKODE: **IE303612**	EMNE: **Bacheloroppgave**	DOKUMENT TILGANG: - Åpen
STUDIUM: **AUTOMATISERINGSTEKNIKK**		ANT SIDER/VEDLEGG: /	BIBL. NR: - Ikke i bruk -

OPPDRAGSGIVER(E)/VEILEDER(E):
Isi as/GL,OLO

OPPGAVE/SAMMENDRAG:
Sette sammen/forbedre eksisterende systemer for kontinuerlig innsamling av data fra bil i bevegelse for opplæring og analyse av maskinlæringsmodell.

Postadresse
Høgskolen i Ålesund
N-6025 Ålesund
Norway

Besøksadresse
Larsgårdsvegen 2
Internett
www.hials.no

Telefon
70 16 12 00
Epostadresse
postmottak@hials.no

Telefax
70 16 13 00

Bankkonto
7694 05 00636
Foretaksregisteret
NO 971 572 140

Denne oppgaven er en eksamensbesvarelse utført av student(er) ved NTNU i Ålesund.

INNHOLD

1 INNLEDNING

Isi as har på oppdrag fra Arvid Gjerde AS laget et system for automatisk gjenkjenning av feil på autovern ved hjelp av maskinlæring. Med dette trengs det et system for innsamling av data for videre trening og til analyse av maskinlæringsmodell. Det skal kontinuerlig samles inn data fra bil i fart, derfor må systemet rundt være presist og justerbart for å sikre god data for videre analyse/trening.

Formålet med oppgaven er å forbedre og integrere de eksiterende systemene og gjøre brukergrensesnittet for operatør mer brukervennlig.

2 BEGREPER

3 PROSJEKTORGANISASJON

3.1 Prosjektgruppe

Studentnummer(e)
Ole Thomas Norbye Theisen Torbjørn Trømborg Michal Leikanger

Tabell: Studentnummer(e) for alle i gruppen som leverer oppgaven for bedømmelse i faget ID 302906

3.1.1 Oppgaver for prosjektgruppen - organisering

3.1.2 Oppgaver for prosjektleder

Ansvarsområde

- Overholde frister
- Fordele oppgaver
- Samle gruppa
- Stille krav til gruppemedlemmer

3.1.3 Oppgaver for sekretær

Ansvarsområde

- Skrive møtereferat
- Få med alt vi gjør

3.1.4 Oppgaver for øvrige medlem(mer)

--

3.2 Styringsgruppe (veileder og kontaktperson oppdragsgiver)

Isi AS

- Bård Indredavik
- Nils Tarjej Hjelme
- Basir Sedighi

4 AVTALER

4.1 Avtale med oppdragsgiver

4.2 Arbeidssted og ressurser

Vi har ingen formelle avtaler med oppdragsgiver, men de har gitt oss tilgang på bedriftens Teams-side der vi kan kommunisere og utveksle informasjon om prosjektet.

Har også fått utdelt testutstyr:

- Kamera
- Switch
- GPS

4.3 Gruppenormer – samarbeidsregler – holdninger

Gruppenormer:

- Støttende
- Inkluderende
- Arbeidsom
- Lærevillig og søke hjelp hos andre
- Åpen

Gruppen skal ha en leder som skal sørge for at alle har en stemme i temaer og diskusjoner om avgjørelser i prosjektet. Leder har ikke autoritet til å overstyre avgjørelser, det skal være en enighet i gruppa før en avgjørelse blir tatt. Ellers skal alt i gruppa bli tatt opp i plenum, slik at alle i gruppa

får anledning til å ta del i diskusjonen. Dette vil sikre at ingen blir utelatt og alle får en helhetlig forståelse av prosjektet.

Man burde ha grunnleggende gode holdninger for å kunne samarbeide og utrykke seg i en jobbsetting/gruppe. Tilegne seg kunnskap, være ydmyk og lære av andre er viktig for å utvikle seg som ingeniør.
Det å tilegne seg kunnskap og søke hjelp hos andre er viktig for oss, som er uerfarne utøvere av vår profesjon, dette er kanskje det viktigste vi kan ta med oss inn i gruppearbeidet. Det er også viktig å ha selvtillit til å ta egne beslutninger og stå for disse.

Spesielt i en gruppe der folk er forskjellige og har andre utgangspunkt enn deg selv er det å være åpen og imøtekommende viktig for at dynamikken i gruppa skal bli best mulig. Derfor har vi laget en egen «brain storm» forum der alle på gruppa kan komme med ideer, som man kan drøfte og diskutere videre på.

5 PROSJEKTBESKRIVELSE

5.1 Problemstilling - målsetting - hensikt

Problemstilling:
Dagens system for innhenting av data er en ganske så provisorisk løsning som fortløpende har blitt satt opp ettersom flere ønskende spesifikasjoner har blitt lagt til systemet. Som en konsekvens av dette kreves det stor systemforståelse av operatøren som skal bruke systemet.

5.1.1 Effektmål

- Gjøre grensesnitt mer brukervennlig
- Forbedre ytelsen på systemet
- Sikre god datainnsamling
- Det skal kunne brukes/betjenes av en person uten inngående teknisk forståelse

5.1.2 Resultatmål

- Lage et grensesnitt som skal kunne brukes av sjåføren av bilen på en trafikksikker måte
- Lage et mer solid system som enkelt kan gjenbrukes
- Validere bilde før lagring for å sikre god datainnsamling.

5.1.3 Prosessmål

- Deltagere av prosjektet skal få mer forståelse av det å jobbe i prosjekt

- Deltagere av prosjektet skal få mer kunnskap om profesjonen

5.2 Krav til løsning eller prosjektresultat – spesifikasjon

Krav til brukergrensesnitt:

- Skal være enkel å bruke og ikke kreve å ha noen form for inngående teknisk forståelse.
- Start og stopp av datainnsamling
- Automatisk kvalitetssikring av data(bilde). Gir bruker beskjed
- Bilde skal kunne vises i GUI for inspeksjon av operatør
- Betjenes på en trafikksikker måte
- Skal kunne sjekke bildestrøm fra kameraer. Når bilen står stille

Krav til system:

- Lett å vedlikeholde
- Rask og sikker dataflyt
- Redusere støy på data

Vi har ikke diskutert det økonomiske med Isi as. Vi har foreløpig det utstyret vi trenger, men vi vil lage prototyper med utstyr vi har tilgjengelig på skolen og komme med forslag til utstyr som trengs videre til ferdig produkt.

5.3 Planlagt framgangsmåte(r) for utviklingsarbeidet – metode(r)

Metoder for prosjektstyring gjennom hele prosjektfasen vil bli GANTT-diagram for planlegging og sette frister for hovedmål og deloppgaver for hele prosjektet fram i tid.

Det vil også bli brukt Kanban-brett for få ned oppgaver som jobbes på og få oversikt over hva som er planlagt og hva som jobbes med her og nå.

GANTT-diagram

Gantt-diagram er en av de mest populære verktøyene for å måle og vise aktiviteter i et prosjekt. Man får oversikt over hva de ulike aktivitetene er, når hver aktivitet starter og slutter og hvem som har ansvar for aktiviteten. Gir deg en totaloversikt over prosjektet.
Svakheter:

- Kan bli uoversiktlig

Kanban-brett

Verktøy for å visualisere stadier i en prosess/arbeidsoppgave.

5.4 Informasjonsinnsamling – utført og planlagt

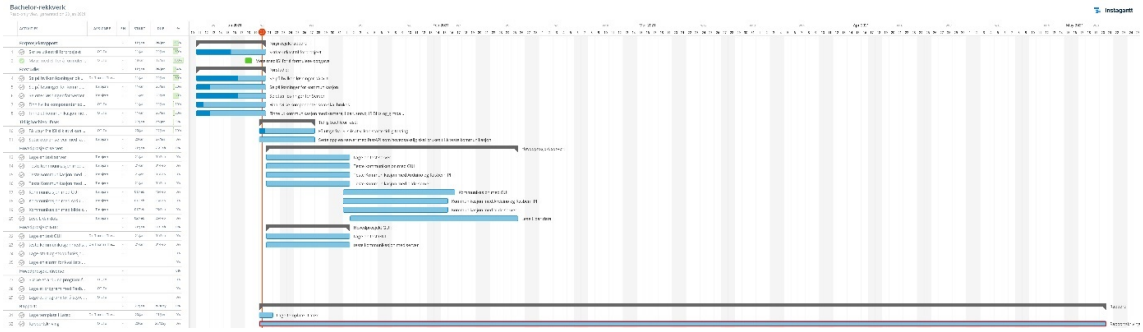
Informasjonsinnsamlingen vil skje i dialog med oppdragsgiver og de har forsikret oss å gi oss den informasjonen vi trenger for å løse oppgaven. Informasjon vi har hentet inn er skisse over eksisterende løsning og vil videre få informasjon om løsningen i detalj. For at arbeidet med hovedprosjektet skal begynne trenger vi kildekode og mer informasjon om systemet på detaljnivå.

5.5 Vurdering – analyse av risiko

Vi har fått en god ramme rundt prosjektet med klare krav som kreves av systemet vi skal designe.

Det viktigste med oppgaven er å lage et system som er enkelt å bruke og som kan ta gode data. Brukergrensesnittet skal være så enkelt at det skal kunne brukes av sjåfør. Dette innebærer en sikkerhetsrisiko for sjåføren, så dette må tas i betraktning om dette er en mulig løsning.

5.6 Hovedaktiviteter i videre arbeid



5.7 Framdriftsplan – styring av prosjektet

5.7.1

5.7.2 Hovedplan

Hovedplanen vist ovenfor er et utkast til endelig framdriftsplan og vil være i kontinuerlig endring ettersom alle premisser for oppgaven ikke er satt.

Hovedoppgaven vil bli delt opp i flere deloppgaver for å holde kontroll på hva som har blitt gjort og vi skal fordele arbeidsoppgavene etter gruppemedlemmers kunnskaper og arbeidskapasitet.

5.7.3 Styringshjelpemidler

Notion: Er en applikasjon som tilbyr komponenter som notater, kanban-brett, lister og påminnelser.

Instagantt: Et online Gantt-diagram der alle gruppemedlemmer kan endre og legge til i sanntid.

Github: Gjennom prosjektet skal vi lagre all kildekoden på Github. På den måten har vi en god oversikt over de forskjellige versjonene av programmene og kan feil søke med å bruke gamle versjoner som fungerer.

Discord: Brukes til kommunikasjon mellom gruppemedlemmer.

5.7.4 Utviklingshjelpemidler

-For øyeblikket har vi ikke behov for noe ekstra hjelpemidler.

5.7.5 Intern kontroll – evaluering

Internkontrollen i prosjektet skal følges opp av gruppeleder. Dette er oppfølging av framdrift, kvalitetssikring av arbeidet vi gjør. Ukentlige statusrapporter og møter skal sikre at alle får framdrift i prosjektet og kvaliteten på det vi gjør er på sitt beste.

5.8 Beslutninger – beslutningsprosess

I møte mellom veiledere og oppdragsgiver har vi blitt enige om presisering av oppgaven og beslutninger som har blitt tatt i forprosjektet. Vi vil ha flere kommende møter med arbeidsgiver om oppgaven for å bli enige om hvilke problemstillinger/områder som skal angripes først. Vi har blitt enige om at omfanget av oppgaven skal reguleres etter hvor mye vi klarer å få gjort.

Viktige beslutninger skal først diskuteres innad i gruppen og videre i samråd med oppdragsgiver. Men vi skal lage en prototyp/løsning til problemstillingen så vi har i grunn full frihet.

6 DOKUMENTASJON

6.1 Rapporter og tekniske dokumenter

- Hva slags dokumentasjon skal utarbeides – utforming, innhold
- Rutiner
- Godkjennelse
- Distribusjon / kopiering
- Oppbevaring
- Vedlikehold

7 PLANLAGTE MØTER OG RAPPORTER

7.1 Møter

7.1.1 Møter med styringsgruppen

Vi har planlagt møter med prosjektgiver hver onsdag annenhver uke. Første møte er planlagt onsdag 03.02.2021.

7.1.2 Prosjektmøter

- Hver uke

7.2 Periodiske rapporter

7.2.1 Framdriftsrapporter (inkl. milepæl)

Hvert gruppemedlem oppdaterer Gantt-diagram med kommentarer i «tasken».

8 PLANLAGT AVVIKSBEHANDLING

Prosjektet har klare hovedmål og delmål. Om framdriften på delmål ikke går som planlagt vil man justere mål etter det. Ha flere milepæler i prosjektet slik at man fullfører det man har kapasitet til gjennom prosjektfasen.

9 UTSTYRSBEHOV/FORUTSETNINGER FOR GJENNOMFØRING

10 REFERANSER

VEDLEGG

[Materiell som er utarbeidet eller innsamlet i tilknytning til rapporten, men som det ikke er naturlig eller hensiktsmessig å ta inn i hoveddelen, skal tas inn som vedlegg.

Vedleggene skal være nummererte og ha en identifiserende tekst.]

Vedlegg 1	Kort identifiserende tekst
Vedlegg 2	… osv.

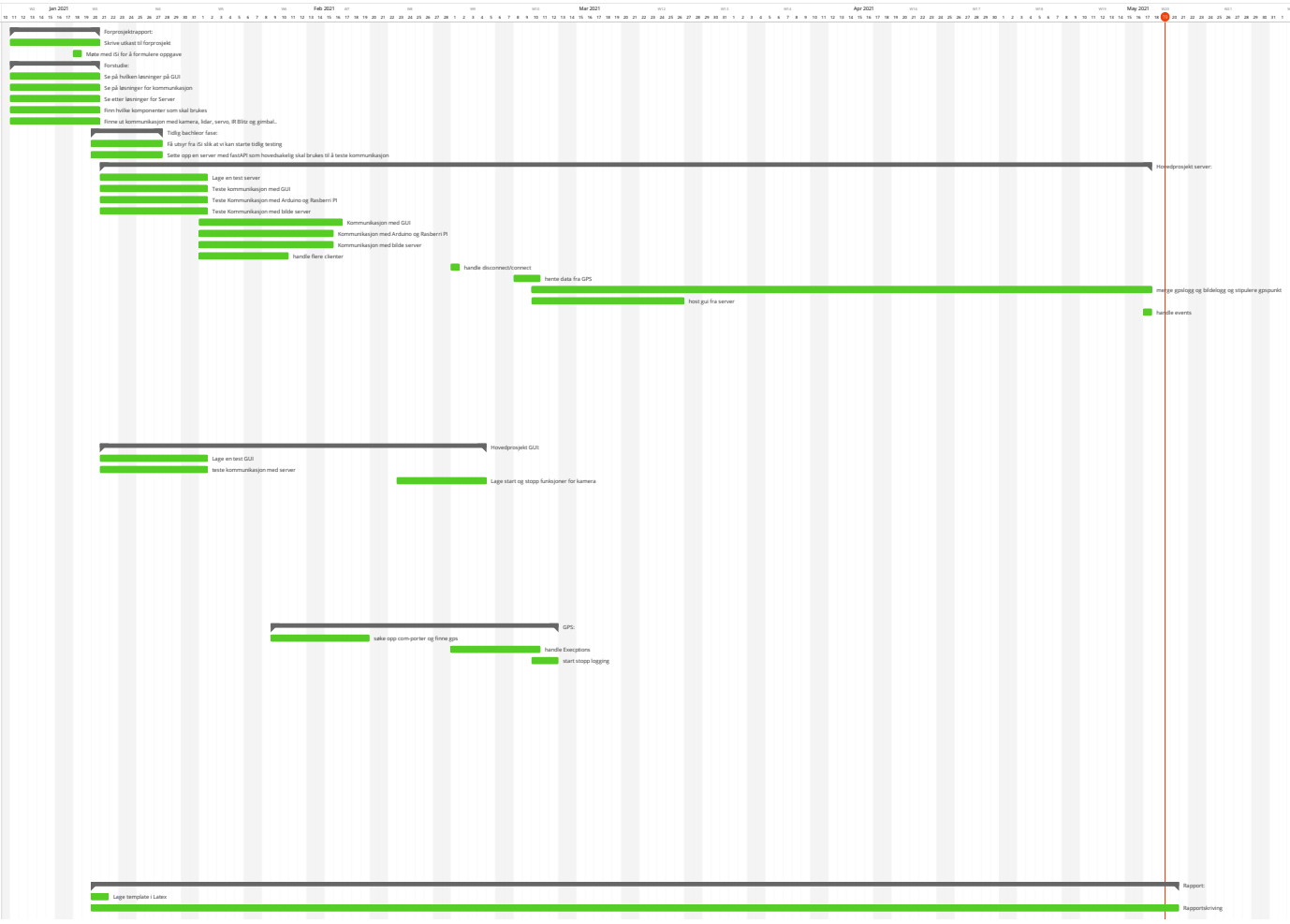
B Gantt diagram

Bachelor-rekkverk

Read-only view, generated on 19 May 2021

Instagantt

	ACTIVITIES	ASSIGNEE	EH	START	DUE	%
	Forprosjektrapport:		-	11/Jan	20/Jan	100%
1	Skrive utkast til forprosjekt	OT, To	-	11/Jan	20/Jan	100%
2	Møte med ISI for å formuler...	OT, To	-	18/Jan	18/Jan	100%
	Forstudie:		-	11/Jan	20/Jan	100%
4	Se på hvilken løsninger på ...	Ine Hinnæs Tho...	-	11/Jan	20/Jan	100%
5	Se på løsninger for kommu...	Torbjørn	-	11/Jan	20/Jan	100%
6	Se etter løsninger for Server	Torbjørn	-	11/Jan	20/Jan	100%
7	Finn hvilke komponenter so...	OT, To	-	11/Jan	20/Jan	100%
8	Finne ut kommunikasjon me...	OT, To	-	11/Jan	20/Jan	100%
	Tidlig bachleor fase:		-	20/Jan	27/Jan	100%
10	Få utsyr fra ISI slik at vi kan ...	OT, To	-	20/Jan	27/Jan	100%
11	Sette opp en server med fa...	Torbjørn	-	20/Jan	27/Jan	100%
	Hovedprosjekt server:		-	27/Jan	17/May	100%
13	Lage en test server	Torbjørn	-	27/Jan	01/Feb	100%
14	Teste kommunikasjon med ...	Torbjørn	-	27/Jan	01/Feb	100%
15	Teste Kommunikasjon med ...	Torbjørn	-	27/Jan	01/Feb	100%
16	Teste Kommunikasjon med ...	Torbjørn	-	27/Jan	01/Feb	100%
17	Kommunikasjon med GUI	Torbjørn	-	01/Feb	10/Feb	100%
18	Kommunikasjon med Ardui...	Torbjørn	-	01/Feb	10/Feb	100%
19	Kommunikasjon med bilde s...	Torbjørn	-	01/Feb	10/Feb	100%
20	handle flere clienter	Ine Hinnæs Tho...	-	01/Feb	10/Feb	100%
21	handle disconnect/connect	Ine Hinnæs Tho...	-	01/Mar	01/Mar	100%
22	hente data fra GPS	Ine Hinnæs Tho...	-	08/Mar	10/Mar	100%
	merge gpslogg og bildelogg...	To, OT	-	10/Mar	17/May	100%
26	host gui fra server	Ine Hinnæs Tho...	-	10/Mar	26/Mar	100%
27	handle events	Ine Hinnæs Tho...	-	17/May	17/May	100%
28	load kamera config ved opp...	Ine Hinnæs Tho...	-			100%
29	start kamera-stream med e...	Ine Hinnæs Tho...	-			100%
30	lagre bilder på ekstern hard...	OT, To	-			100%
31	handle expetions bildetaking	Ine Hinnæs Tho...	-			100%
32	lagring av gpslogg	OT, To	-			100%
33	last kamerakonfig bare ved ...	Ine Hinnæs Tho...	-			100%
34	Hente lagringsplass fra eksb...		-			100%
35	sjekke om program er platf...		-			100%
	Lagring av bilder:		-			100%
37	finne eksterne harddisker og ...	OT, To	-			100%
38	lage en watchdog på harddi...	Ine Hinnæs Tho...	-			100%
	Hovedprosjekt GUI:		-	27/Jan	06/Mar	100%
40	Lage en test GUI	Ine Hinnæs Tho...	-	27/Jan	01/Feb	100%
41	teste kommunikasjon med s...	Ine Hinnæs Tho...	-	27/Jan	01/Feb	100%
42	Lage start og stopp funksjo...		-	26/Feb	06/Mar	100%
43	Lage en alarm for kvalitets ...		-			100%
44	hente gpsdata fra server	Ine Hinnæs Tho...	-			100%
45	connecte med socket til ser...	Ine Hinnæs Tho...	-			100%
46	visning for kamera	Ine Hinnæs Tho...	-			100%
47	Hente lagrings-data fra ser...		-			100%
48	pause bildetaking	Ine Hinnæs Tho...	-			100%
49	vise resterende plass igjen ...	Ine Hinnæs Tho...	-			100%
50	live stream kamera konfig s...		-			100%
51	verifisere bilder		-			100%
	config page	Ine Hinnæs Tho...	-			100%
58	plotte gps-punkter i kart	Ine Hinnæs Tho...	-			100%
59	Give more feedback when t...		-			100%
	GPS:		-	09/Feb	12/Mar	100%
61	søke opp com-porter og fin...		-	09/Feb	19/Feb	100%
62	handle Exeptions		-	01/Mar	10/Mar	100%
63	start stopp logging	Ine Hinnæs Tho...	-	10/Mar	12/Mar	100%
64	Bluetooth til com port	OT, To	-			100%
65	Kommunikasjon med server		-			100%
66	Thread safety		-			100%
67	GPS log	Torbjørn	-			100%
68	teste med produksjon.gps	OT, To	-			100%
	Kamera styring trigging puls:		-			100%
70	Raspberry vs arduino		-			100%
71	Stying PWM kamera trigger		-			100%
72	Kommunikasjon med server		-			100%
73	Behandle feil		-			100%
	Lagring av bilder:		-			100%
75	mappestrukturering	Ine Hinnæs Tho...	-			100%
76	Sjekke om bilde er valid		-			100%
77	lage csv logging, timestamp...	OT, To	-			100%
	Bildeinnhenting:		-			100%
79	lage kamera class		-			100%
80	Sjekke bildekvalitet	OT, To	-			100%
	Hovedprosjekt diverse:		-			100%
82	Lage et program med Rasb...	OT, To	-			100%
	Rapport:		-	20/Jan	20/May	100%
84	Lage template i Latex	Ine Hinnæs Tho...	-	20/Jan	21/Jan	100%
85	Rapportskriving	OT, To	-	20/Jan	20/May	100%



C Inspeksjon av rekkverk

Statens vegvesen

Notat

Til:	Kontaktpersone r i regionene	**Saksbehandler:**	Bård Nonstad
Fra:		**Tlf saksbeh.**	97654306
Kopi til:		**Vår dato:**	10.05.2021

Inspeksjon av rekkverk på riksveger- Instruks

Vegtilsynet har gjennomført tilsyn av rekkverk i Region øst i 2017 (http://www.vegtilsynet.com/tilsyn/tilsynsrapporter/inspeksjoner-som-grunnlag-for-drift-og-vedlikehold) og ett tilsyn av systemet for oppfølging av rekkverk hos Vegdirektoratet i desember i fjor. (http://www.vegtilsynet.com/tilsyn/tilsynsrapporter/system-for-oppfolging-av-avvik-pa-rekkverk). I tillegg er det gjennomført nye tilsyn av rekkverk i Vegavdeling Hedmark og Telemark: https://vegtilsynet.com/tilsyn/tilsynsrapporter/inspeksjon-av-rekkverk.statens-vegvesen-vegavdeling-telemark og https://vegtilsynet.com/tilsyn/tilsynsrapporter/inspeksjon-av-rekkverk

Et av strakstiltakene som må gjøres i løpet av sommeren/høsten er inspeksjon av rekkverk på **riksveg** med et spesielt fokus på manglende bolter. Dersom det registreres manglende bolter, skal disse monteres fortløpende eller så snart som mulig.

Kontaktpersoner i regionene:
- Region øst: Inge Martin Haugen og Kjersti Røislien
- Region sør: Sven Håkon Jørgensen
- Region vest: Jan Bremer Remø, Rune Henning Skår og Torgeir Strand
- Region midt: Audun Vognild og Tom Tverli
- Region nord: Jan-Erik Stenmark

På hvilke veger skal registreringen foregå?
Riksveger

Hvilke objekter skal registreres?
Det er tilstand på rekkverk langs kjøreveg som skal registreres. Dette gjøres på det nye objektet *Tilstandsgrad, rekkverk (VT 947)* i NVDB. Ingen registrering skal foregå på det gamle *tilstand/skade, rekkverk*.
Det er kun tilstand på rekkverk som skal registreres. Nyregistrering eller endring av rekkverksobjekt inngår ikke i oppdraget, men når man først er ute er det selvsagt fornuftig å oppdatere dette dersom det er feil i dagens registeringer.

Hovedfokus er registrering av tilstand på stålrekkverk og wirerekkverk, og disse rekkverkene skal prioriteres først.

På betongrekkverk hvor tilstand er registrert og dokumentert i løpet de siste 5 år er det ikke nødvendig å gjennomføre en ny registrering av tilstand.

Rekkverk for gang- og sykkelveg skal ikke registreres i denne omgang. Dette gjelder rekkverk på ytterside av gang-sykkelveg og rekkverk for gående og syklende mot kjøreveg.

Hvordan skal registreringen/inspeksjonen foregå?

- *Type inspeksjon*: Enkel inspeksjon slik det er beskrevet i R610. «Det vil si å registrere tilstand, skader og forhold som kan påvirke funksjon og sikkerhetsforhold, trafikksikkerhet, framkommelighet, framtidig drift og vedlikehold, miljø, estetikk og universell utforming».
- *Inspektører:* Entreprenør i driftskontrakt evt. rekkverkskontrakt/Egne ansatte/Sommerstudenter (Opp til regionen). Det er viktig å gi opplæring til de som ikke har noe kjennskap til rekkverkstyper/skader.

- *Utførelse/registreringsmetode:* Visuell kontroll fra bil evt. til fots ved behov. Sakte kjøring (0-15 km/t). Det bør være to i bilen hvor en kjører og den andre gjennomfører registreringene. Registrering av skjevhet, bolter, høyde, skader og arbeidsbredde. Ved behov gå ut av bil for å inspisere. Objekt tilstandsgrad, rekkverk deles opp i 100 m lengder (skjer automatisk) og vurderes med skader for hver 100 m. Avhengig av totallengde på rekkverket vil noen lengder være forskjellig fra 100 m.

- *Spesielt ved manglende bolter:* Dersom det registreres manglende bolter, skal disse monteres fortløpende eller så snart som mulig! Dersom de monteres på stedet velges: *Ok, utbedret bolter på stedet.* Dersom de skal monteres senere velges manglende/løse bolter.

- *Registreringsverktøy:* NVDB 123, Vegreg, annet registreringsverktøy som kan hente informasjon fra NVDB, evt. excel-ark som printes ut fra NVDB. Dersom excel-ark benyttes er det viktig med en god dialog med geodataseksjonen med tanke på de endringene i vegnett som er planlagt gjennomført. Dersom inspektør er en entreprenør som ikke har egen løsning for objektregistrering kan NVDB 123 benyttes og tilgang kan skaffes ved å ta kontakt med IKT-forvaltning i Vegdirektoratet. E-post: Ikt-henvelser@vegvesen.no

  For hjelp med programvare/løsning for registrering av inspeksjoner så henvises det til geodataseksjonene på region.

- *Rapportering:* Data legges inn i NVDB fortløpende evt. etterregistreres manuelt inn i NVDB. Ved utbedringer i etterkant av inspeksjonene så må man være bevist på at endringene også lagres i NVDV

- *Frist*: for å gjennomføre inspeksjonene og sette i manglende bolter er satt til 1.10.2019

- *Ajourhold*: Man må sørge for at utbedringer i etterkant av inspeksjonene blir ajourført i NVDB.

Mer teknisk om selve registreringen

Det er laget et script som oppretter tilstandsobjekt automatisk til alle rekkverk inkludert oppdeling i 100 m lengder på rekkverk som er lengre enn 100 m. Dersom rekkverket er 130 meter så deles det opp i 2 lengder: Ett på 100 m og ett på 30 m. Dersom det er 129 meter deles det ikke opp. For hver 100 m av rekkverket registreres tilstand, også om tilstand er ok på hele strekningen. Ulike rekkverkstyper vil kreve noe ulik registrering. Eksempelvis for wirerekkverk og betongrekkverk er ikke bolter en egenskap som skal registreres.
Malene som er vedlagt på de siste sidene angir hvilke egenskapstyper som skal registreres for de ulike rekkverkstypene. Det er mulig å lage seg egne maler dersom man eksempelvis vet at man skal bare registrere en rekkverkstype.

Registrering i NVDB 123

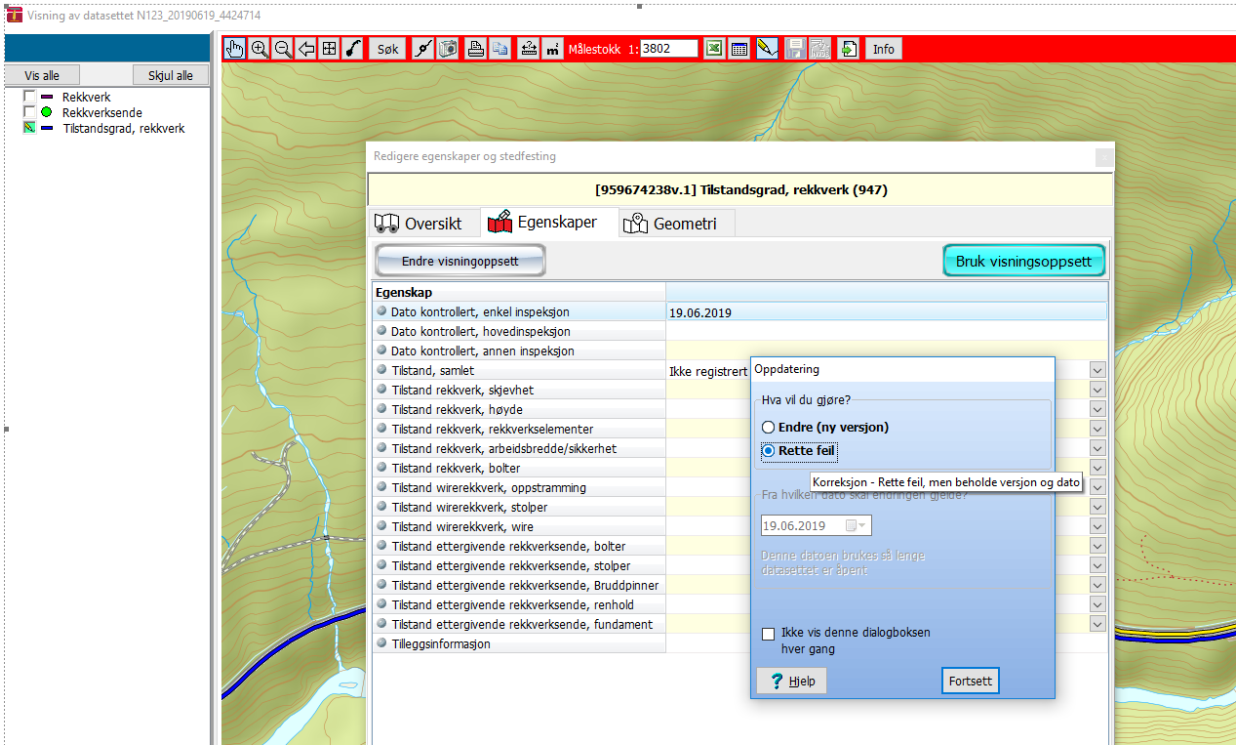
Man velger vegobjekttype «Tilstandsgrad, rekkverk», «Rekkverk» og «Rekkverksende» og setter disse oppdaterbare. Endringene skal hovedsakelig gjøres på «Tilstandsgrad, rekkverk», men det kan gjøres enkle justeringer på de andre om det er behov for det.

I NVDB opereres det med versjoner av registrerte vegobjekter/forekomster. Ved hjelp script opprettes en eller flere forekomster av «Tilstandsgrad, rekkverk» for hver forekomst av rekkverk på riksveg. Alle disse forekomstene av «Tilstandsgrad, rekkverk» vil være versjon nr 1.

Hver gang vi skal oppdatere en forekomst i NVDB må vi ta stilling til om vi skal rette eksisterende versjon, eller opprette en ny versjon. Om vi oppretter en ny versjon vil den eksisterende versjonen settes historisk, og disse dataene vil være tilgjengelige for ettertiden.

Første gangs registrering av tilstand ute på vegen gjøres ved å rette versjon 1 av Tilstandsgrad-objektet. Det må da i NVDB 123 velges «Rette feil» (se figur).

For hver registrering må det minimum angis verdi til «Dato kontrollert, enkel inspeksjon» og «Tilstand, samlet». For «Tilstand, samlet» velges enten *ok* eller *tiltaksbehov.* Ved *ok* er det ikke krav om å fylle inn mer info. Velges *tiltaksbehov* skal minst en av de andre tilstandsparameterne være forskjellig *fra ok.*



Andre registreringsløsninger

Sinus infra fra Triona blir tilgjengelig på ettersommeren. Nettleserbasert.
Excel. Må hentes ut på spesielt format som kan importeres tilbake til NVDB.
VegReg.

Spesialtilfeller ved registrering

Er det flere avvik på en 100 meters-strekning, eksempelvis at rekkverk er både for skjevt og for lavt, registreres valg «skjevhet» på egenskapstype «tilstand rekkverk, skjevhet» og for «høyt/lav» på «Tilstand rekkverk, høyde».

Dersom det på samme 100 meters strekning er både manglende bolter som må utbedres på stedet og manglende bolter som ikke utbedres på stedet, så registreres *manglende/løse bolter.* Ved ettermontering av bolter så velges *Rette feil* og *ok, utbedret bolter på stedet.* Dersom det registreres flere manglende bolter på en 100 meters strekning så skrives antall manglende bolter inn i tilleggsinformasjon.

Høyde: Kravet til høyde er i utgangspunkt avvik fra byggehøyde. I mange tilfeller er ikke byggehøyde kjent. Vi har kommet frem til at regelverket **godtar høyder mellom 650mm og 850 mm fra overkant asfalt til overkant skinne/profil**. Registreres lavere eller høyere verdier enn dette registreres det som avvik uavhengig av rekkverkstype. (Det er ikke meningen at man fysisk skal måle høyde på alle rekkverk, men er man i tvil kan en måling være fornuftig)

Årsak manglende bolter

Det er ønskelig å få en tilbakemelding fra de som gjennomfører inspeksjonene om hva de mener er årsaken til de evt. manglende boltene. Dette kommer vi tilbake til etter at inspeksjonen er utført.

HMS

Ved registrering må en sørge for varslingsplan med tilstrekkelig sikring og varsling. Krav er beskrevet i håndbok N301. Regionen bør vurdere putebil på sterkt trafikkerte strekninger.

Kontaktperson

Kontaktperson i Vegdirektoratet vil være Bård Nonstad, e-post: bard.nonstad@vegvesen.no , tlf 97654306.

Statens vegvesen

Mal for stålskinne m/ulike stolpetyper, stålrør med ulike stolpetyper, treskinne med ulike stolpetyper.

Egenskapstype	Valg	Forklaring
Tilstand, samlet	Ok	Indikerer at alt er ok og dermed ikke behov for tiltak. De andre egenskapstypene trengs ikke å angis ok.
	Tiltaksbehov	Behov for tiltak. Minst en tilstandsparameter skal være angitt med verdi forskjellig fra OK
	Ikke registrert	Det er ikke gjennomført tilstandsinspeksjon
Tilstand rekkverk, skjevhet	Ok	Mindre enn 10 cm avvik ift rekkverkslinje for topp rekkverk.
	Skjevhet	Mer enn 10 cm avvik ift rekkverkslinje for topp rekkverk
Tilstand rekkverk, høyde	Ok	Avvik er innenfor -10 cm til +5 cm ift. opprinnelig rekkverkshøyde
	For høyt/lav	Avvik er utenfor -10 cm til +5 cm ift. opprinnelig rekkverkshøyde
Tilstand rekkverk, rekkverkselementer	Ok	Ikke skade som reduserer styrke/egenskap/funksjon. Ikke andre skader pga. vinter og vinterdrift
	Ska	Skader som reduserer styrke/egenskap/funksjon, kan være skader pga. vinter og

	der	vinterdrift. Info om evt skade angis under tilleggsinformasjon
Tilstand rekkverk, arbeidsbredde/sikkerhet	Ok	Ingen trafikkfarlige objekter innenfor arbeidsbredde
	Trafikkfarlige objekter innenfor arbeidsbredde	Det finnes trafikkfarlige objekter innenfor arbeidsbredden
Tilstand rekkverk, bolter	OK	Bolter er ok innenfor gitt utstrekning
	Ok, utbedret bolter på stedet	Løse bolter er skrudd til, manglende bolter er erstattet

	Mangl ende/l øse bolt er	Det forekommer manglende bolter og/eller løse bolter

Statens vegvesen

Mal for betongrekkverk- alle typer.

Egenskapstype	Valg	Forklaring
Tilstand, samlet	Ok	Indikerer at alt er ok og dermed ikke behov for tiltak. De andre egenskapstypene trengs ikke å angis ok.
	Tiltaksbehov	Behov for tiltak. Minst en tilstandsparameter skal være angitt med verdi forskjellig fra OK
	Ikke registrert	Det er ikke gjennomført tilstandsinspeksjon
Tilstand rekkverk, skjevhet	Ok	Mindre enn 10 cm avvik ift rekkverkslinje for topp rekkverk.
	Skjevhet	Mer enn 10 cm avvik ift rekkverkslinje for topp rekkverk
Tilstand rekkverk, høyde	Ok	Avvik er innenfor -10 cm til +5 cm ift. opprinnelig rekkverkshøyde.
	For høyt/lav	Avvik er utenfor -10 cm til +5 cm ift. opprinnelig rekkverkshøyde
Tilstand rekkverk, rekkverkselementer	Ok	Ikke skade som reduserer styrke/egenskap/funksjon. Ikke andre skader pga. vinter og vinterdrift
	Ska	Skader som reduserer styrke/egenskap/funksjon, kan være skader pga. vinter og

	der	vinterdrift. Info om evt skade angis under tilleggsinformasjon

Statens vegvesen

Mal for wirerekkverk

Egenskapstype	Valg	Forklaring
Tilstand, samlet	Ok	Indikerer at alt er ok og dermed ikke behov for tiltak. De andre egenskapstypene trengs ikke å angis ok.
	Tiltaksbehov	Behov for tiltak. Minst en tilstandsparameter skal være angitt med verdi forskjellig fra OK
	Ikke registrert	Det er ikke gjennomført tilstandsinspeksjon
Tilstand rekkverk, skjevhet	Ok	Mindre enn 10 cm avvik ift rekkverkslinje for topp rekkverk.
	Skjevhet	Mer enn 10 cm avvik ift rekkverkslinje for topp rekkverk
Tilstand rekkverk, høyde	Ok	Avvik er innenfor -10 cm til +5 cm ift. opprinnelig rekkverkshøyde
	For høyt/lav	Avvik er utenfor -10 cm til +5 cm ift. opprinnelig rekkverkshøyde
Tilstand wirerekkverk, oppstramming	OK	Oppstramming og forankring er ok
	Behov	Det er behov for oppstramming av wirerekkverk

	for oppstramming	
Tilstand wirerekkverk, stolper	OK	Stolper er ok
	Mekanisk skade på stolper	Det forekommer mekanisk skade på en eller flere stolper
	Løse stolper	Det forekommer en eller flere løsestolper
	Mangler topphette	Det mangler topphette på en eller flere stolper
Tilstand wirerekkverk, wire	OK	Wirene er ok
	Mekan	Wire har en eller flere mekanske skader

	isk ska de	
	Lod rett still ing	Wire er ute av stilling

Statens vegvesen

Mal for ettergivende rekkverksende, alle typer

Egenskapstype	Valg	Forklaring
Tilstand, samlet	Ok	Indikerer at alt er ok og dermed ikke behov for tiltak. De andre egenskapstypene trengs ikke å angis ok.
	Tiltaksbehov	Behov for tiltak. Minst en tilstandsparameter skal være angtt med verdi forskjellig fra OK
	Ikke registrert	Det er ikke gjennomført tilstandsinspeksjon
Tilstand rekkverk, skjevhet	Ok	Mindre enn 10 cm avvik ift rekkverkslinje for topp rekkverk.
	Skjevhet	Mer enn 10 cm avvik ift rekkverkslinje for topp rekkverk
Tilstand rekkverk, høyde	Ok	Avvik er innenfor -10 cm til +5 cm ift. opprinnelig rekkverkshøyde
	For høyt/lav	Avvik er utenfor -10 cm til +5 cm ift. opprinnelig rekkverkshøyde
Tilstand rekkverk, rekkverkselementer	Ok	Ikke skade som reduserer styrke/egenskap/funksjon. Ikke andre skader pga. vinter og vinterdrift
	Skader	Skader som reduserer styrke/egenskap/funksjon, kan være skader pga. vinter og vinterdrift. Info om evt skade angis under tilleggsinformasjon
Tilstand rekkverk, arbeidsbredde/sikkerhet	Ok	Ingen trafikkfarlige objekter innenfor arbeidsbredde
	Trafikkfarlige objekter	Det finnes trafikkfarlige objekter innenfor arbeidsbredden

	innenfor arbeidsbredde	
Tilstand ettergivende rekkverksende, bolter	OK	Boltene i rekkverksende er ok
	Ok, utbedret bolter på stedet	Løse bolter er skrudd til, manglende bolter er erstattet
	Manglende/løse bolter	Manglende bolter eller løse bolter
Tilstand ettergivende rekkverksende, Bruddpinner	OK	Bruddpinner er ok
	Mangler bruddpinner	Det mangler en eller flere bruddpinner
Tilstand ettergivende rekkverksende, renhold	OK	Ikke partikler/gjenstander/masser under/i rekkverksende
	Partikler/gjenstander/masser	Det finnes partikler/gjenstander/masser under/i rekkverksende

	under /i rekkverksende	
Tilstand ettergivende rekkverksende, fundament	OK	Løsmasser bak stolpe som sikrer funksjon ved påkjørsel (frigjøring av stolpe)
	Manglende løsmasser bak stolper	Manglende løsmasser bak stolpe som sikrer funksjon ved påkjørsel (frigjøring av stolpe)

D GNS 2000

Navilock GNS 2000 MFi GPS GLONASS Bluetooth receiver with logger function

Description

The GNS 2000 is a small GNSS receiver. The Receiver can GPS and GLONASS signals handle and received simultaneously. Through the SBAS support and the use of up to 99 channels, it achieves a high accuracy. Routes and locations can be recorded using the logger function.

Specification

- MFi
- GNSS chipset: MT3333
- SBAS: WAAS, EGNOS, QZSS, MSAS GAGAN
- Cold start: 35 s
- Warm start: 33 s
- Hot start: < 1 s
- Data transmitting technology: Bluetooth 2.1 + EDR
- Baud Rate: 57600 Baud
- Data logger: max. 15 h
- Connector: mini USB
- Operating time: 10 h
- Voltage: 5 V DC
- Charching current: 500 mA
- Charching time: 3,5 h
- Operating temperature: 0 ~ 50 °C
- Dimension: 79,10 x 45,30 x 11,30 mm
- Weight: 50 g

System requirements

- x86 PC, notebook or tablett with BT
- Smartphone or tablett with Android >= 2.3 with BT
- iPhone, iPod, iPad ... with BT

Package content

- GNSS BT receiver
- Sucker for disk mounting
- Cable USB plug A > plug mini USB
- Car charching cable USB jack (max. 300 mA)

Item No. 60323

EAN: 4037735104860

Country of origin: GERMANY

Package:

E ECX-1420

ECX-1420/1320

9th Generation Intel® Xeon®/Core™ i7/i5/i3 (Coffee Lake Refresh) Expandable Fanless System with Intel® C246 Chipset, 6 GigE LAN with 4 PoE+, 4 Front-access SSD Tray, 1 PCIe x16, 1 PCIe x4, 6 USB 3.1, 2 M.2 Socket, 3 SIM, 32 Isolated DIO, High Performance, Rugged, -40°C to 75°C Extended Temperature

Features

- 8 Cores 9th Generation Intel® Xeon®/Core™ i7/i5/i3 (Coffee Lake Refresh) with workstation-grade Intel® C246 Chipset
- Fanless, -40°C to 75°C Operating Temperature
- Multiple USB 3.1 Gen 2 supports up to 10Gbps SuperSpeed data transfer
- 6 Independent GigE LAN with 4 IEEE 802.3at PoE+
- 32 Isolated DIO, 6 USB 3.1, 4 COM RS-232/422/485
- Storage : 4 2.5" SSD Tray, 1 CFast, 1 M.2, 4 SATA III
- Expansion : 1 PCIe x16, 1 PCIe x4, 2 Mini PCIe/mSATA, 1 M.2
- PCIe x16 expansion supports up to 200W Power Budget
- 3 External SIM Socket for WiFi/4G/3G/LTE/GPRS/UMTS
- 6V to 36V DC Power Input with 80V Surge Protection
- Configurable Ignition Power Control

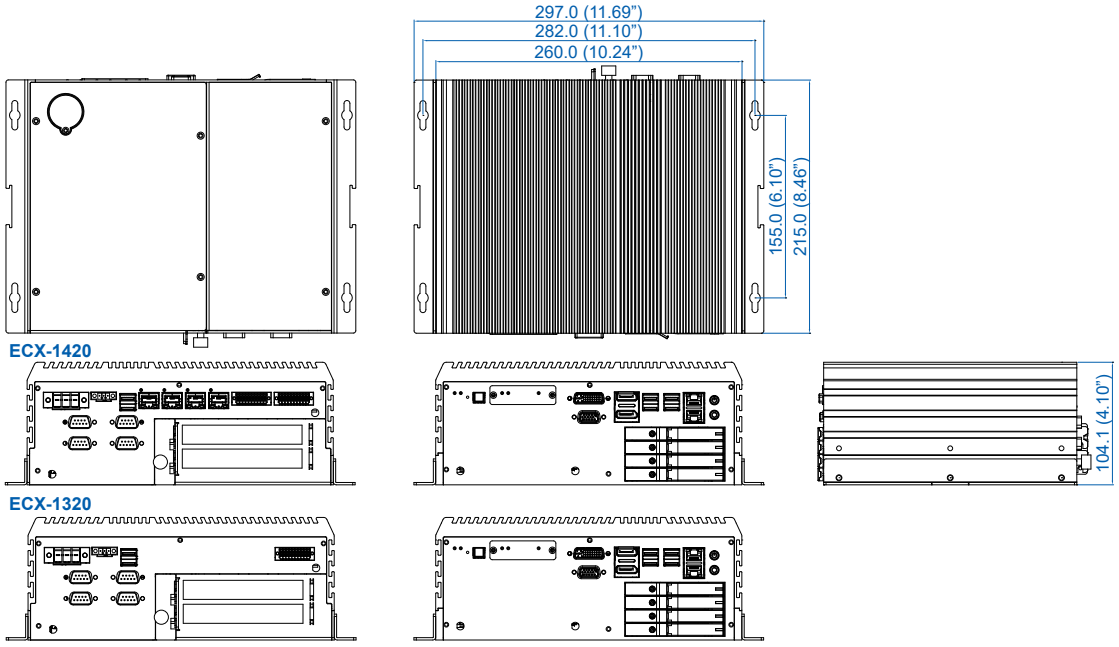
Specifications

System	
Processor	Intel® Xeon®/Core™ i7/i5/i3 Processor (CFL-R S/CFL-S)
Chipset	Intel® C246
BIOS	AMI
SIO	IT8786E
Memory	2 DDR4 2666MHz SO-DIMM, up to 64GB
I/O Interface	
Serial	4 COM RS-232/422/485 (ESD 8KV)
USB	▪ 6 USB 3.1 (6 External) ▪ 1 USB 2.0 (Internal)
Isolated DIO	ECX-1420 : 32 Isolated DIO : 16 DI, 16 DO ECX-1320 : 16 GPIO
LED	Power, HDD, Wireless, PoE (ECX-1400)
SIM Card	3 External SIM Card Socket
Expansion	
Mini PCIe	2 Full-size for PCIe/USB/External SIM Card/mSATA
PCIe	▪ 1 PCIe x16 ▪ 1 PCIe x4
M.2	1 M.2 Key E Socket
SUMIT A, B	2 SUMIT Slot (Optional)
Graphics	
Graphics Processor	Intel® UHD Graphics 630
Interface	▪ 1 VGA : Up to 1920 x 1200 @60Hz ▪ 1 DVI-D : Up to 1920 x 1200 @60Hz ▪ 2 DisplayPort : Up to 4096 x 2304 @60Hz
Storage	
SATA	4 SATA III (6Gbps) support S/W RAID 0, 1, 5, 10
mSATA	2 SATA III (Mini PCIe Type, 6Gbps)
M.2	1 M.2 Key B Socket
Storage Device	▪ 1 CFast Socket, Push-in/Push-out Ejector ▪ 4 Front-access 2.5" SSD/HDD Tray
Audio	
Audio Codec	Realtek ALC888S-VD, 7.1 Channel HD Audio
Audio Interface	1 Mic-in, 1 Line-out
Ethernet	
LAN 1	Intel® I219LM GigE LAN supports iAMT 12.0
LAN 2	Intel® I210 GigE LAN

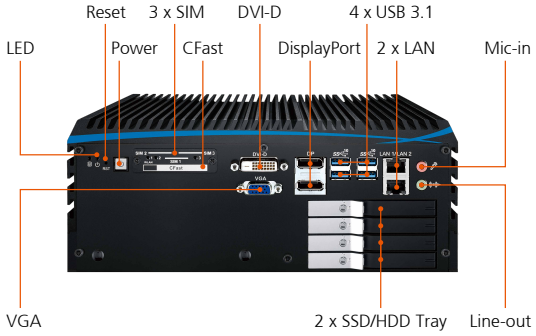
PoE (ECX-1400)	
LAN 3 to 6	GigE IEEE 802.3at (25.5W/48V) PoE+ by Intel® I350
Power	
Power Input	6V to 36V, DC-in
Power Interface	3-pin Terminal Block : V+, V-, Frame Ground
Ignition Control	16 Mode (Internal)
Remote Switch	3-pin terminal block : On, Off, IGN
Surge Protection	Up to 80V/1ms Transient Power
Others	
TPM	Optional Infineon SLB9665 supports TPM 2.0, LPC Interface
Watchdog Timer	Reset : 1 to 255 sec./min. per step
Smart Management	Wake on LAN, PXE supported
HW Monitor	Monitoring temperature, voltages. Auto throttling control when CPU overheats.
Software Support	
OS	Windows 10, Linux
Mechanical	
Dimensions (W x L x H)	260mm x 215mm x 104.1mm (10.2" x 8.46" x 4.0")
Weight	5.5 kg (12.13 lb)
Mounting	▪ Wallmount by mounting bracket ▪ DIN Rail Mount (Optional)
Environment	
Operating Temperature	35W TDP CPU : -40°C to 75°C (-40°F to 167°F) 65W TDP CPU : -40°C to 55°C (-40°F to 131°F) 80W TDP CPU : -40°C to 45°C (-40°F to 113°F)
Storage Temperature	-40°C to 85°C (-40°F to 185°F)
Humidity	5% to 95% Humidity, non-condensing
Relative Humidity	95% @ 75°C
Shock	▪ IEC 60068-2-27 ▪ SSD : 50G @ wallmount, Half-sine, 11ms
Vibration	▪ IEC 60068-2-64 ▪ SSD : 5Grms, 5Hz to 500Hz, 3 Axis
EMC	CE, FCC, EN50155, EN50121-3-2

Dimensions

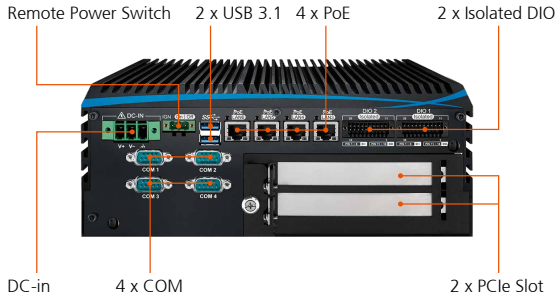
Unit : mm/ inch



Front I/O



Rear I/O



Order Information

ECX-1420	ECX-1400, 6 GigE LAN w/4 PoE+, 4 SSD Tray, 1 PCIe x16, 1 PCIe x4, 6 USB 3.1, 4 COM, 3 SIM, 32 Isolated DIO
ECX-1320	ECX-1300, 2 GigE LAN, 4 SSD Tray, 1 PCIe x16, 1 PCIe x4, 6 USB 3.1, 4 COM, 3 SIM, 16 GPIO

Accessories

DDR4 32G	Certified DDR4 32GB 2666MHz RAM
DDR4 16G	Certified DDR4 16GB 2666/2400/2133MHz RAM
DDR4 8G	Certified DDR4 8GB 2666/2400/2133MHz RAM
DDR4 4G	Certified DDR4 4GB 2666/2400/2133MHz RAM
PWA-160WB-WT	160W, 24V, 85V AC to 264V AC Power Adapter with 3-pin Terminal Block (7.62mm pitch), Wide Temperature -30°C to +70°C
PWA-280WB-WT	280W, 24V, 85V AC to 264V AC Power Adapter with 3-pin Terminal Block (7.62mm pitch), Wide Temperature -30°C to +70°C
VESA Mount	VESA Mounting Kit
DIN-RAIL Kit	DIN Rail and VESA Mounting Kit
TMK2-20P-100	Terminal Block 20-pin to Terminal Block 20-pin Cable, 100cm
TMK2-20P-500	Terminal Block 20-pin to Terminal Block 20-pin Cable, 500cm
TMB-TMBK-20P	Terminal Board with One 20-pin Terminal Block Connector and DIN-Rail Mounting
4G Module	Mini PCIe 4G/GPS Module with Antenna
WiFi & Bluetooth	WiFi & Bluetooth Module with Antenna

CPU List

Series	CPU	Cores	GHz	TDP (W)	ECC RAM
Intel® Xeon®	E-2176G	6	4.6	80	Yes
	E-2124G	4	4.5	71	
	E-2278GE	8	4.7	80	
	E-2278GEL	8	3.9	35	
	E-2226GE	6	4.6	80	
Intel® Core™	i7-8700	6	4.6	65	N/A
	i7-8700T	6	4	35	
	i7-9700E	8	4.4	65	
	i7-9700TE	8	3.8	35	
	i5-8500	6	4.1	65	
	i5-8500T	6	3.5	35	
	i5-9500E	6	4.2	65	
	i5-9500TE	6	3.6	35	
	i3-8100	4	3.6	65	Yes
	i3-8100T	4	3.1	35	
	i3-9100E	4	3.7	65	
	i3-9100TE	4	3.2	35	

F Altus aps3

ALTUS
POSITIONING
SYSTEMS

APS-3

The APS-3 is a high precision satellite receiver and communications unit specifically designed for the surveying market.

Integrated with state-of-the-art technology, the APS-3 provides surveyors with high productivity, performance and flexibility.

GPS + GLONASS RTK is delivered in a small and light design for ultra portability. The APS-3 has integrated wireless options that include a GSM/GPRS or CDMA modem for easy Real Time Network connection, and a digital UHF transceiver for Base/Rover operation on demand.

For ease of use and simplicity, the APS-3 is completely configurable from the data collector via Bluetooth for either Base or Rover operation with the internal UHF radio, or for Network Rover operation with the internal cellular modem.

FEATURES

- 136 Channel GPS + GLONASS receiver
- Integrated 3.5 G cellular modem, Tri-Band HSPA, Quad-Band GSM/GPRS/EDGE
- Integrated Digital UHF transceiver, 406-470 MHz
- Integrated Bluetooth
- Dual Hot-Swap Li-Ion Batteries with Fuel Gauge
- Removable min.2 GB SD Card storage

With Altus's open architecture philosophy, the user has the choice of data collector software from MicroSurvey FIELDGenius or Carlson SurvCE to custom software.Rugged field computers are used to communicate with the APS-3 via Bluetooth for a cable free operation. Furthermore, the APS-3 houses two hot swappable Li-Ion batteries for continuous operation. Re-charging is done within a few hours with the included charger.

Product	GPS	GLONASS	UHF Radio	GSM/GPRS or CDMA	Ext.Ant.Port
APS-3u	✓	✓	✓	✓	
APS-3m	✓	✓		✓	
APS-3x	✓	✓	✓	✓	✓

The APS-3 is Simply Better Value by Design.

GNSS precision and reliability are subject to anomalies due to multipath, obstructions, satellite geometry, and atmospheric conditions. Contact Altus for more information. Specifications are subject to change without notice.

SPECIFICATIONS

Channels	136
GPS	L1/L2/L2C
GLONASS	L1/L2
SBAS	WAAS, EGNOS
Standalone	H: 1.3 m, V: 1.9 m
SBAS	H: 0.6 m, V: 0.8 m
DGPS	H: 0.50 m, V: 0.90 m
RTK	H: 0.6 cm + 0.5 ppm, V: 1 cm + 1 ppm
Static	H: 2 mm + 0.5 ppm, V: 5 mm + 0.5 ppm
Output Rate	25 Hz
Measurement Rate	25 Hz
Latency	<20 msec
Average Time To Fixed RTK	<7 sec
Cold Start	<45 sec
Warm Start	<20 sec
Re-acquisition	<1.2 sec
Integrated UHF Radio	406 to 470 MHz
Integrated Cellular Modem	3.5 G HSPA Modem, HSDPA 7.2 Mbps, GPRS multi-slot class 12, EDGE (E-GPRS) multi-slot class 12, UMTS/HSPA 850/1900/2100 MHz, Quad-Band GSM 850/900/1800/1900 MHz
Integrated Bluetooth	Class 2
Memory Storage	2 GB SD card minimum, removable
User Interface	Power/Reset/Logging/Data Collector or PC
Serial Ports	2 Lemo
Waterproofing	IP67
Certification	CE, FCC Class B Part 15
Internal Battery	2 x Li-Ion, 5000 mAh, 7.4V
Current Drain	1.0 to 1.5A, 2.75A peak
External Power Input	9 to 18 VDC, Lemo
Weight	1.3 kg
Dimensions	178 mm Dia. X 89.7 mm
Operating Temperature	-20 to +65 degrees C
Storage Temperature	-40 to +75 degrees C
Shock/Drop	2 m

Each APS-3 is delivered with 2 x Li-Ion hot swappable batteries, a 2-Bay Battery Charger, a serial cable & a power cable. The APS-3 is ready to use direct from the shipping case.

20725 Western Avenue | Suite 100,Torrance | California 90501 | USA | T: 310-541-8139 | Fax: 310-541-8257 www.altus-ps.com

G Electrical drawings

A.1 Source code

https://github.com/Bachelor2021

A Source code Server/backend

```
import numpy as np
import sys
from typing import List
import cv2
from fastapi import FastAPI,Request
from starlette.responses import HTMLResponse, StreamingResponse
from starlette.websockets import WebSocket, WebSocketDisconnect
from starlette.concurrency import run_until_first_complete
from fastapi.staticfiles import StaticFiles
from fastapi.templating import Jinja2Templates
from fastapi.responses import RedirectResponse
from starlette.routing import WebSocketRoute
from starlette.exceptions import HTTPException as StarletteHTTPException
from core.cameraStream import CameraStream
from concurrent.futures.process import ProcessPoolExecutor
import json
from core.camera import Camera
from threading import Lock, Thread
from time import sleep
import cvb
import uvicorn
import csv
from core.FPS import FPS
```

```
from core.models import models
import time
from fastapi.middleware.cors import CORSMiddleware
from manager import ConnectionManager
from io import BytesIO
from core.timer import Timer
from core.imageSave import ImageSave
import queue
from gps import GpsHandler
import os
from os import name, path
from datetime import datetime
from multiprocessing import Process,Queue,Pool,Manager,Pipe
from pydantic import BaseModel
from core.helpers.helper_server import *
from core.helpers.helper_server import ConnectionManager
from core.models.models import GpsData ,freq,TripName,UsbPort
from core.serialService import SerialService
from core.merging import merge
from ctypes import c_wchar_p
# camera = Camera()
# camera.start_stream()


app = FastAPI()


manager = ConnectionManager()
image_freq = 20
gps_freq =0
debug = False
storage = {}
```

```
camerasDetected =[]
tempTrip =""

isConfigured =False




config_loaded = False



guruMode = False


#start the GPS
gpsData ={}
gps_status = {}
```

```
valider = True

abort = False
logging = False
closeServer = False
start_Puls = False
drive_in_use ="C:"
storageLeft_in_use = 50
gpsControl = True

#manage socket connections
manager = ConnectionManager()

#   Variables and lists for the gps coordinates
#   lists
longlitudelist = []
latitudelist = []
#   variables


#creates new folder for saving imaging
#createFolder()


started = False
```

```
origins = [
    "http://localhost.tiangolo.com",

    "https://localhost.tiangolo.com",
    "http://localhost",
    "http://localhost:8080",
    "http://localhost:3000",
    "http://localhost:8000",
    "http://10.22.182.47:4200",
    "http://10.0.222.1",
    "http://localhost:8000/changeFps",




]

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)




def sendFPS():
```

```
    global image_freq,start_Puls,gps_freq,gpsControl,started


    fps = image_freq

    if not gpsControl :
        fps = image_freq

    elif gpsControl  and started:


        fps = gps_freq

    fps = int(fps)
    serial.write([fps])
    read = serial.read()

    return read


@app.post('/connectFPS')
def connect(port:UsbPort):
    global serial
    port = port.port

    if port.connection:

        connected = serial.connect(port)

    else:
```

```
        connected = serial.close(port)

    return connected

@app.get('/gps')
async def getData():
    global gps_status,gps,gps_freq


    isConnected = gps.isConnected()
    gps_status = gps.getData()
    gps_status['connected'] = isConnected
    gps_freq = int(gps_status['velocity'])
    read = sendFPS()

    gps_status['pulse'] = read


    return gps_status

#   Open a csv file and appends coordinates in lists
#   return a list with the latitude coordinates and a list with longitude coordinates
@app.get('/GetCoordinates')
def getgpscoordinates():
    mainlist =[]
    tripnamelist =[]
    date = getDate()
    absolute_path = os.path.dirname(os.path.abspath(__file__))
    absolute_path = fixPathServer(absolute_path)
```

```
path = absolute_path+"/log/"

folders= os.listdir(path)

try:
    for folder in folders:
        subFolders = os.listdir(absolute_path+"/log/"+folder)


        for subFolder in subFolders:

            latitudelist = []
            longlitudelist =[]

            path1 = path+folder+"/"+subFolder+"/" +'merged.csv'
            exist = os.path.exists(path1)
            if exist:

                with open(path1, newline='') as csvgps:
                    gpsreader = csv.reader(csvgps, delimiter=',', quotechar='|')
                    next(gpsreader)
                    i = next(gpsreader)
                    for row in gpsreader:
                        #  make a list for each column in the csv file

                        latitudelist.append(row[3])
                        longlitudelist.append(row[4])


                    cordlist =list(zip(latitudelist,longlitudelist))
```

```
                    tripnamelist.append(subFolder)
                    mainlist.append(cordlist)
    except Exception as e:
        print(e)


    return {"lists":mainlist,"names":tripnamelist}




def log(data):
    global nameTrip



    date = getDate()
    absolute_path = os.path.dirname(os.path.abspath(__file__))
    new = fixPathServer(absolute_path)
    path = new+"/log/"+date+"/"+nameTrip.value

# Open gps csv file and make a csv reader object
    with open(path +'/log.csv','a', newline='') as csvgps:

        fieldnames = ['stream','images']
        writer = csv.DictWriter(csvgps, fieldnames=fieldnames)

        writer.writerow({"stream":data['stream'],"images":data['results']['images_ok']})
```

```
@app.get('/getStates')
async def states():

    states = getStates()



    return states




@app.post('/gpserror')
async def data(test):

    return test

@app.get('/RaspFPS')
def fps():
    global image_freq,start_Puls,gps_freq,gpsControl,started
```

```
    fps = image_freq

    if not gpsControl :
        fps = image_freq

    elif gpsControl  and started:


        fps = gps_freq









    return {'fps':fps,"start":start_Puls}

@app.post('/changeFps')
async def change(freq:freq):
    global image_freq


    image_freq = freq.fps

    return {"fps":image_freq}


@app.get('/merge')
def mergeFiles():
    global tempTrip
```

```
    date = getDate()
    absolute_path = os.path.dirname(os.path.abspath(__file__))
    absolute_path= fixPathServer(absolute_path)
    path = absolute_path+"/log/"+date+"/"+nameTrip.value
    payload = False
    try:

        result = merge(path)

        payload = {"event":"merging","result":result}

    except Exception:
        pass

    finally:

        return payload
```

```
@app.post('/start1')
def startA(tripname:TripName):
    global stream1_Stopped,drive_in_use
    nameTrip.value = tripname.name
    fps = FPS().start()
```

```
    date = getDate()
    if not bool(init1.value):
        return {"message": "stream 1 has stopped","images_ok":str(index1.value),"images"

    while 1:









        if bool(stream1_Stopped.value):
            break









    fps.stop()
    print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
    return {"message": "stream 1 has stopped","images_ok":str(index1.value),"images":"st




@app.post('/start2')
```

```
def startB(tripname:TripName):
    global stream2_Stopped, imageQueue_2,drive_in_use,nameTrip

    nameTrip.value = tripname.name
    date = getDate()
    fps = FPS().start()
    if not bool(init2.value):
        return  {"message": "stream 2 has stopped","images_ok":str(index2.value),"images
    while 1:



        if bool(stream2_Stopped.value):
            break







    fps.stop()
    print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))

    return {"message": "stream 2 has stopped","images_ok":str(index2.value),"images":"",
    # start bildetaking

@app.post('/start3')
def startC(tripname:TripName):

    global stream3_Stopped,imageQueue_3,drive_in_use,nameTrip
```

```
    nameTrip.value = tripname.name
    if not bool(init3.value):
        return  {"message": "stream 3 has stopped","images_ok":str(index3.value),"images
    while 1:




        if bool(stream3_Stopped.value):
            break




    return {"message": "stream 3 has stopped","images_ok":str(index3.value),"images":"",
    # start bildetaking


def emergencyStop():
    global abort,image_freq,capturing,isConfigured,started,stopStream,stream1_Stopped,st
    toggleGPSControl(False)
    abort =True
    isConfigured = False
    started =False
    image_freq = 0
    stopStream.value =1
    stream1_Stopped.value=1
    stream2_Stopped.value=1
    stream3_Stopped.value = 1
```

```
    capturing.value = 0




async def abortStream():
    """Stops all the streams

    By setting the pulse hz to zero
    then the capturing times out


"""
    global gps,start_Puls,image_freq,gpsControl,stream0,stream1,isConfigured,stopStream,
    print("stopping stream")
    started = False
    toggleGPSControl(False)
    image_freq =0
    isConfigured = False
    stopStream.value =1




async def start_acquisition():
    """ starts gps logging and turns on GPS control
```

```
    """
    global abort,image_freq,stream0,stream1,stream2
    print("Starting stream")
    image_freq =0
    toggleGPSControl(False)

    index1.value =0
    index2.value=0
    index3.value=0
    gps.toggleLogging(True)
    save.value = 1



    abort=False
    image_freq =0









def startPulse():
    """
    starts the image capturing
    """
    global image_freq,started,gps,capturing,stream0,stream1
```

```
    toggleGPSControl(True)
    gps.toggleLogging(True)
    capturing.value= 1
    started = True




def pause():
    """Pauses the image capturing

    """
    global gps,image_freq,capturing
    gps.toggleLogging(False)
    toggleGPSControl(False)
    image_freq =0
    capturing.value =0



@app.post('/changeimagefreq/')
async def change_image_freq(freq:freq):
    global image_freq

    image_freq = freq.fps

    return image_freq


def toggleGPSControl(value):
```

```
    global gpsControl

    gpsControl =  value


@app.get('/capturing')
async def getCapturing():
    global index1,index2,index3


    return {"camera1":index1.value,"camera2":index2.value,"camera3":index3.value}


@app.get('/storage')
async def getStorage():
    global storage,started,drive

    storages =checkStorageAllDrives()

    mostFree = most_free_space()


    if mostFree:
        drive.value = mostFree['name']



    payload = estimateStorageTime(storages,10)

    total = payload['total']['free']
```

```
    if int(total)<5 and started:
        emergencyStop()
        await manager.broadcast(json.dumps({"event":"storage","data":"empty"}))




    return payload

# estimate hows

def storageLeft(storages):
    global drive_in_use,imagesave,imagesave2,imagesave3,storageLeft_in_use

    drives = storages['drives']

    for i in drives:
        drive = drives[i]

        if(drive['name'] == drive_in_use):

            storageLeft_in_use = int(drive['free'])




async def initCameraA():
    global stream0, abort,camerasDetected
```

```
    status = "ok"



    try:



        if not bool(init1.value):
            status ="failed"

        else:
            camerasDetected.append("camera1")




    except Exception as e:
        print(e)
        print("initializing of camera 1 failed")
        status = "failed"
    finally:

        await manager.broadcast(json.dumps({"event": "initA", "data": status}))
```

```
async def initCameraB():
    global stream1,cameras,camerasDetected


    status = "ok"


    try:


        if not bool(init2.value):
            status ="failed"
        else:
            camerasDetected.append("camera2")









    except Exception as e:
        print(e)
        print("initializing of camera 1 failed")
        status = "failed"
    finally:

        await manager.broadcast(json.dumps({"event": "initB", "data": status}))

def index_in_list(a_list, index):
```

```
        test = index < len(a_list)
        return test

async def initCameraC():
    global stream0, abort,cameras,camerasDetected

    status = "ok"


    try:


        if not bool(init3.value):
            status ="failed"
        else:
            camerasDetected.append("camera3")




    except Exception as e:
        print(e)
        print("initializing of camera 1 failed")
        status = "failed"
    finally:

        await manager.broadcast(json.dumps({"event": "initC", "data": status}))
```

```
async def validate(cam):
    global camera_1, camera_2,camera_3
    try:

        camera = None
        if cam == 'A':
            camera = camera_1

        elif cam =="B":
            camera = camera_2

        elif cam == "C":
            camera = camera_3

        print(camera.isRunning())
        frame, status = camera.get_image()
        if status == cvb.WaitStatus.Ok:



            b64 = cvbImage_b64(frame)

            raw_data = {"event": "snapshot", "data": b64}

            data = json.dumps(raw_data)
```

```
            await manager.broadcast(data)
    except Exception as e:
        print(e)


@app.get('/stopFeed')
async def stoplive():
    global valider

    valider = False



def gen():
    global camera_1,valider

    while 1:

        if valider:
            try:
                frame = camera_1_Connection1.recv()

                #frame = np.array(frame)
                frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                frame = cv2.resize(frame, (640, 480))
                _, frame = cv2.imencode('.jpg', frame)

                image = frame.tobytes()

                yield (b'--frame\r\n'
```

```
                    b'Content-Type: image/jpeg\r\n\r\n' + image + b'\r\n')



            except Exception as e:
                pass









def gen1():
    global valider

    while 1:

        if valider:

            try:
                frame = camera_2_Connection1.recv()

                #frame = np.array(frame)
                frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                frame = cv2.resize(frame, (640, 480))
```

```
                _, frame = cv2.imencode('.jpg', frame)

                image = frame.tobytes()

                yield (b'--frame\r\n'
                       b'Content-Type: image/jpeg\r\n\r\n' + image + b'\r\n')


            except Exception as e:
                pass


def gen2():
    global camera_3,valider

    while 1:

        if valider:



            try:
                frame = camera_3_Connection1.recv()

                #frame = np.array(frame)
                frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                frame = cv2.resize(frame, (640, 480))
                _, frame = cv2.imencode('.jpg', frame)

                image = frame.tobytes()
```

```
                yield (b'--frame\r\n'
                       b'Content-Type: image/jpeg\r\n\r\n' + image + b'\r\n')


            except Exception as e:
                pass


@app.get('/video_feed1')
def video_feed1():
    global valider


    return StreamingResponse(gen(), media_type="multipart/x-mixed-replace; boundary=fram
```

```
@app.get('/video_feed2')
def video_feed2():
    global valider

    return StreamingResponse(gen1(), media_type="multipart/x-mixed-replace; boundary=fra
```

```
@app.get('/video_feed3')
def video_feed3():
    global valider

    return StreamingResponse(gen2(), media_type="multipart/x-mixed-replace; boundary=fra
```

```
def resett():
    global tempTrip,gps,isConfigured,started,stream1_Stopped,stream2_Stopped,stream3_Sto
    isConfigured= False
    started = False
    stream1_Stopped.value =0
    stream2_Stopped.value =0
    stream3_Stopped.value =0
    #capturing.value =0
    gps.toggleLogging(False)
    save.value =0
    stopStream.value =0
```

```
def initGPS():
    global gps
    gps.startInit()



def getImages():
    global index1,index2,index3

    return {"camera1":index1.value,"camera2":index2.value,"camera3":index3.value}


def startfps():
    global image_freq,capturing,isConfigured,gpsControl,streamStopped

    toggleGPSControl(False)
    image_freq = 5







def getStates():
    global capturing,config_loaded,started,isConfigured,gpsControl,guruMode,debug
    return {"capturing":capturing.value,"init_ok":config_loaded,"running":started,"isCon


@app.websocket("/stream/{client_id}")
```

```
async def websocket_endpoint(websocket: WebSocket, client_id: int):
    global started,config_loaded,isConfigured,gps,drive_in_use,gpsControl,valider,tempTr
    await manager.connect(websocket)
    await websocket.send_text(json.dumps({"event": "connected", "data": "connected to se
    try:
        while True:
            data = await websocket.receive_text()
            data = json.loads(data)
            event = data['event']
            msg = data['data']




            if(event == "onConnection"):

                await manager.broadcast(json.dumps({"connection": "connected"}))






            elif(event == 'start'):

                isConfigured = True
                tempTrip = str(msg)
                gps.setTripName(str(msg))
                drive_in_use = await createImageFolder(msg)
                if drive_in_use =="failed":
                    await manager.broadcast(json.dumps({"event": "error","data":"no driv
```

```
        else:

            drive.value = drive_in_use
            await start_acquisition()
            await manager.broadcast(json.dumps({"event": "starting"}))




    elif(event =="pulse"):

        startPulse()
    elif(event == 'stop'):

        await abortStream()

        await manager.broadcast(json.dumps({"event": "stopping"}))

    elif(event == "init"):

        camerasDetected = []
        await initCameraA()
        await initCameraB()
        await initCameraC()



        await manager.broadcast(json.dumps({"event":"detected","data":len(camera
```

```
elif(event == "validation"):
    await validate(msg)

elif(event =="gps"):
    print(msg)

elif(event == "start_acquisition"):
    status = start_acquisition()

elif(event == "create_trip"):
    await createImageFolder(data)
    await manager.broadcast(json.dumps({"event":"folderCreated"}))

elif(event == "toggleGps"):
    toggleGPSControl(msg)

elif(event == "live"):
    valider = msg
    viewer.value = int(msg)
    capturing.value = int(msg)



elif(event == "debug"):
    gps.setDebug(msg)
    debug= msg
```

```
elif(event=="search"):

    n =discoverCamerasLength()
    i =0


    print("Cameras:"+ str(n))
    for device in range(int(n)):
        camerasDetected.append(str(i))



        i=i+1
    message = json.dumps({"event":"search","data":len(camerasDetected)})
    await manager.send_personal_message(message, websocket)


elif(event == "guru"):
    guruMode = msg

elif(event=="initGPS"):
    initGPS()

elif(event == "pause"):
    pause()

elif(event =="reset"):
    startfps()
    resett()
```

```
                images = getImages()

                await manager.broadcast(json.dumps({"event":"stopped","data":images}))
                states = getStates()
                await manager.broadcast(json.dumps({"event":"states","data":states}))

            elif (event =="emergency"):
                emergencyStop()
                states = getStates()

                await manager.broadcast(json.dumps({"event":"states","data":states}))

            elif(event=="log"):
                log(msg)

            elif(event=="states"):

                states = getStates()

                await manager.broadcast(json.dumps({"event":"states","data":states}))




    except (WebSocketDisconnect,RuntimeError) as e:  # WebSocketDisconnect

        print("[WEBSOCKET] websocket disconnect")

        await manager.disconnect(websocket)
```

```
@app.on_event("startup")
async def startup():
    global camera_1,camera_2,camera_3,cameras,camerasDetected
    print("[startup] init cameras")
```

```
@app.on_event("shutdown")
def shutdown_event():
    global imagesave,imagesave2 ,closeServer,gps

    print("shutting down server")
    gps.raise_exception()
    gps.join()
```

```
def main(arg):
```

```
    #start the API server
    uvicorn.run(app, host="10.0.222.1", port=8000,log_level="error")


if __name__ == "__main__":

    # define all shared memory variables
    capturing = Manager().Value('i',0)
    stream1_Stopped = Manager().Value('i',0)
    stream2_Stopped = Manager().Value('i',0)
    stream3_Stopped = Manager().Value('i',0)
    stopStream = Manager().Value('i',0)
    viewer = Manager().Value('i',0)
    index1 = Manager().Value('i',0)
    index2 = Manager().Value('i',0)
    index3 = Manager().Value('i',0)
    save = Manager().Value('i',0)


    camera_1_Connection1,camera_1_Connection2  = Pipe(duplex=True)
    camera_2_Connection1,camera_2_Connection2  = Pipe(duplex=True)
    camera_3_Connection1,camera_3_Connection2  = Pipe(duplex=True)

    init1 = Manager().Value('i',0)
    init2 = Manager().Value('i',0)
    init3 = Manager().Value('i',0)


    drive = Manager().Value(c_wchar_p, "drive")
```

```
nameTrip = Manager().Value(c_wchar_p, "trip")



#construct the three camerastream processes
stream0 = CameraStream(2,"3",capturing,stream3_Stopped,index3,drive,nameTrip,stopStr
stream1 = CameraStream(1,"2",capturing,stream2_Stopped,index2,drive,nameTrip,stopStr
stream2 = CameraStream(0,"1",capturing,stream1_Stopped,index1,drive,nameTrip,stopStr

#start the processes
stream0.start()
stream1.start()
stream2.start()



# start the GPS handler thread
gps = GpsHandler(debug)
gps.daemon = True
gps.start()

#connect to serial
serial = SerialService()
serial.connect()



main(debug)
```

```
    # isi 192.168.10.153
    #169.254.108.159
    #192.168.0.100
```

app.py

```
import numpy as np
import sys
from typing import List
import cv2
from fastapi import FastAPI,Request
from starlette.responses import HTMLResponse, StreamingResponse
from starlette.websockets import WebSocket, WebSocketDisconnect
from starlette.concurrency import run_until_first_complete
from fastapi.staticfiles import StaticFiles
from fastapi.templating import Jinja2Templates
from fastapi.responses import RedirectResponse
from starlette.routing import WebSocketRoute
from starlette.exceptions import HTTPException as StarletteHTTPException
from core.cameraStream import CameraStream
from concurrent.futures.process import ProcessPoolExecutor
import json
from core.camera import Camera
from threading import Lock, Thread
from time import sleep
import cvb
import uvicorn
import csv
from core.FPS import FPS
from core.models import models
import time
```

```
from fastapi.middleware.cors import CORSMiddleware
from manager import ConnectionManager
from io import BytesIO
from core.timer import Timer
from core.imageSave import ImageSave
import queue
from gps import GpsHandler
import os
from os import name, path
from datetime import datetime
from multiprocessing import Process,Queue,Pool,Manager,Pipe
from pydantic import BaseModel
from core.helpers.helper_server import *
from core.helpers.helper_server import ConnectionManager
from core.models.models import GpsData ,freq,TripName,UsbPort
from core.serialService import SerialService
from core.merging import merge
from ctypes import c_wchar_p
# camera = Camera()
# camera.start_stream()

app = FastAPI()

manager = ConnectionManager()
image_freq = 20
gps_freq =0
debug = False
storage = {}
```

```
camerasDetected =[]
tempTrip =""

isConfigured =False




config_loaded = False



guruMode = False


#start the GPS
gpsData ={}
gps_status = {}

valider = True
```

```
abort = False
logging = False
closeServer = False
start_Puls = False
drive_in_use ="C:"
storageLeft_in_use = 50
gpsControl = True

#manage socket connections
manager = ConnectionManager()

#   Variables and lists for the gps coordinates
#   lists
longlitudelist = []
latitudelist = []
#   variables


#creates new folder for saving imaging
#createFolder()


started = False




origins = [
```

```
    "http://localhost.tiangolo.com",

    "https://localhost.tiangolo.com",
    "http://localhost",
    "http://localhost:8080",
    "http://localhost:3000",
    "http://localhost:8000",
    "http://10.22.182.47:4200",
    "http://10.0.222.1",
    "http://localhost:8000/changeFps",



]

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)




def sendFPS():
    global image_freq,start_Puls,gps_freq,gpsControl,started
```

```
    fps = image_freq

    if not gpsControl :
        fps = image_freq

    elif gpsControl  and started:


        fps = gps_freq

    fps = int(fps)
    serial.write([fps])
    read = serial.read()

    return read


@app.post('/connectFPS')
def connect(port:UsbPort):
    global serial
    port = port.port

    if port.connection:

        connected = serial.connect(port)

    else:
        connected = serial.close(port)
```

```
    return connected

@app.get('/gps')
async def getData():
    global gps_status,gps,gps_freq


    isConnected = gps.isConnected()
    gps_status = gps.getData()
    gps_status['connected'] = isConnected
    gps_freq = int(gps_status['velocity'])
    read = sendFPS()

    gps_status['pulse'] = read


    return gps_status

#   Open a csv file and appends coordinates in lists
#   return a list with the latitude coordinates and a list with longitude coordinates
@app.get('/GetCoordinates')
def getgpscoordinates():
    mainlist =[]
    tripnamelist =[]
    date = getDate()
    absolute_path = os.path.dirname(os.path.abspath(__file__))
    absolute_path = fixPathServer(absolute_path)

    path = absolute_path+"/log/"
```

```
    folders= os.listdir(path)

    try:
        for folder in folders:
            subFolders = os.listdir(absolute_path+"/log/"+folder)


            for subFolder in subFolders:

                latitudelist = []
                longlitudelist =[]

                path1 = path+folder+"/"+subFolder+"/" +'merged.csv'
                exist = os.path.exists(path1)
                if exist:

                    with open(path1, newline='') as csvgps:
                        gpsreader = csv.reader(csvgps, delimiter=',', quotechar='|')
                        next(gpsreader)
                        i = next(gpsreader)
                        for row in gpsreader:
                            #  make a list for each column in the csv file

                            latitudelist.append(row[3])
                            longlitudelist.append(row[4])


                        cordlist =list(zip(latitudelist,longlitudelist))
                    tripnamelist.append(subFolder)
                    mainlist.append(cordlist)
```

```
    except Exception as e:
        print(e)


    return {"lists":mainlist,"names":tripnamelist}




def log(data):
    global nameTrip



    date = getDate()
    absolute_path = os.path.dirname(os.path.abspath(__file__))
    new = fixPathServer(absolute_path)
    path = new+"/log/"+date+"/"+nameTrip.value

# Open gps csv file and make a csv reader object
    with open(path +'/log.csv','a', newline='') as csvgps:

        fieldnames = ['stream','images']
        writer = csv.DictWriter(csvgps, fieldnames=fieldnames)

        writer.writerow({"stream":data['stream'],"images":data['results']['images_ok']})
```

```
@app.get('/getStates')
async def states():

    states = getStates()



    return states




@app.post('/gpserror')
async def data(test):

    return test

@app.get('/RaspFPS')
def fps():
    global image_freq,start_Puls,gps_freq,gpsControl,started


    fps = image_freq
```

```
    if not gpsControl :
        fps = image_freq

    elif gpsControl  and started:


        fps = gps_freq









    return {'fps':fps,"start":start_Puls}

@app.post('/changeFps')
async def change(freq:freq):
    global image_freq


    image_freq = freq.fps

    return {"fps":image_freq}


@app.get('/merge')
def mergeFiles():
    global tempTrip

    date = getDate()
```

```
    absolute_path = os.path.dirname(os.path.abspath(__file__))
    absolute_path= fixPathServer(absolute_path)
    path = absolute_path+"/log/"+date+"/"+nameTrip.value
    payload = False
    try:

        result = merge(path)

        payload = {"event":"merging","result":result}

    except Exception:
        pass

    finally:

        return payload
```

```
@app.post('/start1')
def startA(tripname:TripName):
    global stream1_Stopped,drive_in_use
    nameTrip.value = tripname.name
    fps = FPS().start()
    date = getDate()
    if not bool(init1.value):
```

```
        return {"message": "stream 1 has stopped","images_ok":str(index1.value),"images"

    while 1:




        if bool(stream1_Stopped.value):
            break









    fps.stop()
    print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
    return {"message": "stream 1 has stopped","images_ok":str(index1.value),"images":"st




@app.post('/start2')
def startB(tripname:TripName):
    global stream2_Stopped, imageQueue_2,drive_in_use,nameTrip
```

```
    nameTrip.value = tripname.name
    date = getDate()
    fps = FPS().start()
    if not bool(init2.value):
        return  {"message": "stream 2 has stopped","images_ok":str(index2.value),"images
    while 1:



        if bool(stream2_Stopped.value):
            break







    fps.stop()
    print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))

    return {"message": "stream 2 has stopped","images_ok":str(index2.value),"images":"",
    # start bildetaking

@app.post('/start3')
def startC(tripname:TripName):

    global stream3_Stopped,imageQueue_3,drive_in_use,nameTrip

    nameTrip.value = tripname.name
    if not bool(init3.value):
```

```
        return  {"message": "stream 3 has stopped","images_ok":str(index3.value),"images
    while 1:




        if bool(stream3_Stopped.value):
            break




    return {"message": "stream 3 has stopped","images_ok":str(index3.value),"images":"",
    # start bildetaking


def emergencyStop():
    global abort,image_freq,capturing,isConfigured,started,stopStream,stream1_Stopped,st
    toggleGPSControl(False)
    abort =True
    isConfigured = False
    started =False
    image_freq = 0
    stopStream.value =1
    stream1_Stopped.value=1
    stream2_Stopped.value=1
    stream3_Stopped.value = 1
    capturing.value = 0
```

```
async def abortStream():
    """Stops all the streams

    By setting the pulse hz to zero
    then the capturing times out


"""
    global gps,start_Puls,image_freq,gpsControl,stream0,stream1,isConfigured,stopStream,
    print("stopping stream")
    started = False
    toggleGPSControl(False)
    image_freq =0
    isConfigured = False
    stopStream.value =1




async def start_acquisition():
    """ starts gps logging and turns on GPS control


    """
    global abort,image_freq,stream0,stream1,stream2
```

```
    print("Starting stream")
    image_freq =0
    toggleGPSControl(False)

    index1.value =0
    index2.value=0
    index3.value=0
    gps.toggleLogging(True)
    save.value = 1



    abort=False
    image_freq =0









def startPulse():
    """
    starts the image capturing
    """
    global image_freq,started,gps,capturing,stream0,stream1

    toggleGPSControl(True)
    gps.toggleLogging(True)
```

```
    capturing.value= 1
    started = True




def pause():
    """Pauses the image capturing

    """
    global gps,image_freq,capturing
    gps.toggleLogging(False)
    toggleGPSControl(False)
    image_freq =0
    capturing.value =0



@app.post('/changeimagefreq/')
async def change_image_freq(freq:freq):
    global image_freq

    image_freq = freq.fps

    return image_freq


def toggleGPSControl(value):
    global gpsControl
```

```
    gpsControl =  value


@app.get('/capturing')
async def getCapturing():
    global index1,index2,index3


    return {"camera1":index1.value,"camera2":index2.value,"camera3":index3.value}


@app.get('/storage')
async def getStorage():
    global storage,started,drive

    storages =checkStorageAllDrives()

    mostFree = most_free_space()


    if mostFree:
        drive.value = mostFree['name']



    payload = estimateStorageTime(storages,10)

    total = payload['total']['free']
    if int(total)<5 and started:
        emergencyStop()
```

```
        await manager.broadcast(json.dumps({"event":"storage","data":"empty"}))




    return payload

# estimate hows

def storageLeft(storages):
    global drive_in_use,imagesave,imagesave2,imagesave3,storageLeft_in_use

    drives = storages['drives']

    for i in drives:
        drive = drives[i]

        if(drive['name'] == drive_in_use):

            storageLeft_in_use = int(drive['free'])






async def initCameraA():
    global stream0, abort,camerasDetected

    status = "ok"
```

```
    try:


        if not bool(init1.value):
            status ="failed"

        else:
            camerasDetected.append("camera1")




    except Exception as e:
        print(e)
        print("initializing of camera 1 failed")
        status = "failed"
    finally:

        await manager.broadcast(json.dumps({"event": "initA", "data": status}))









async def initCameraB():
```

```
    global stream1,cameras,camerasDetected


    status = "ok"


    try:


        if not bool(init2.value):
            status ="failed"
        else:
            camerasDetected.append("camera2")









    except Exception as e:
        print(e)
        print("initializing of camera 1 failed")
        status = "failed"
    finally:

        await manager.broadcast(json.dumps({"event": "initB", "data": status}))

def index_in_list(a_list, index):
        test = index < len(a_list)
        return test
```

```
async def initCameraC():
    global stream0, abort,cameras,camerasDetected

    status = "ok"


    try:


        if not bool(init3.value):
            status ="failed"
        else:
            camerasDetected.append("camera3")




    except Exception as e:
        print(e)
        print("initializing of camera 1 failed")
        status = "failed"
    finally:

        await manager.broadcast(json.dumps({"event": "initC", "data": status}))
```

```
async def validate(cam):
    global camera_1, camera_2,camera_3
    try:

        camera = None
        if cam == 'A':
            camera = camera_1

        elif cam =="B":
            camera = camera_2

        elif cam == "C":
            camera = camera_3

        print(camera.isRunning())
        frame, status = camera.get_image()
        if status == cvb.WaitStatus.Ok:




            b64 = cvbImage_b64(frame)

            raw_data = {"event": "snapshot", "data": b64}

            data = json.dumps(raw_data)
            await manager.broadcast(data)
    except Exception as e:
```

```
        print(e)


@app.get('/stopFeed')
async def stoplive():
    global valider

    valider = False



def gen():
    global camera_1,valider

    while 1:

        if valider:
            try:
                frame = camera_1_Connection1.recv()

                #frame = np.array(frame)
                frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                frame = cv2.resize(frame, (640, 480))
                _, frame = cv2.imencode('.jpg', frame)

                image = frame.tobytes()

                yield (b'--frame\r\n'
                       b'Content-Type: image/jpeg\r\n\r\n' + image + b'\r\n')
```

```
            except Exception as e:
                pass
```

```
def gen1():
    global valider

    while 1:

        if valider:

            try:
                frame = camera_2_Connection1.recv()

                #frame = np.array(frame)
                frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                frame = cv2.resize(frame, (640, 480))
                _, frame = cv2.imencode('.jpg', frame)
```

```
                image = frame.tobytes()

                yield (b'--frame\r\n'
                       b'Content-Type: image/jpeg\r\n\r\n' + image + b'\r\n')


            except Exception as e:
                pass


def gen2():
    global camera_3,valider

    while 1:

        if valider:



            try:
                frame = camera_3_Connection1.recv()

                #frame = np.array(frame)
                frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                frame = cv2.resize(frame, (640, 480))
                _, frame = cv2.imencode('.jpg', frame)

                image = frame.tobytes()

                yield (b'--frame\r\n'
```

```
                    b'Content-Type: image/jpeg\r\n\r\n' + image + b'\r\n')

            except Exception as e:
                pass


@app.get('/video_feed1')
def video_feed1():
    global valider

    return StreamingResponse(gen(), media_type="multipart/x-mixed-replace; boundary=fram









@app.get('/video_feed2')
def video_feed2():
    global valider

    return StreamingResponse(gen1(), media_type="multipart/x-mixed-replace; boundary=fra
```

```
@app.get('/video_feed3')
def video_feed3():
    global valider

    return StreamingResponse(gen2(), media_type="multipart/x-mixed-replace; boundary=fra
```

```
def resett():
    global tempTrip,gps,isConfigured,started,stream1_Stopped,stream2_Stopped,stream3_Sto
    isConfigured= False
    started = False
    stream1_Stopped.value =0
    stream2_Stopped.value =0
    stream3_Stopped.value =0
    #capturing.value =0
    gps.toggleLogging(False)
    save.value =0
    stopStream.value =0
```

```
def initGPS():
```

```
    global gps
    gps.startInit()



def getImages():
    global index1,index2,index3

    return {"camera1":index1.value,"camera2":index2.value,"camera3":index3.value}



def startfps():
    global image_freq,capturing,isConfigured,gpsControl,streamStopped

    toggleGPSControl(False)
    image_freq = 5









def getStates():
    global capturing,config_loaded,started,isConfigured,gpsControl,guruMode,debug
    return {"capturing":capturing.value,"init_ok":config_loaded,"running":started,"isCon



@app.websocket("/stream/{client_id}")
async def websocket_endpoint(websocket: WebSocket, client_id: int):
    global started,config_loaded,isConfigured,gps,drive_in_use,gpsControl,valider,tempTr
```

```
    await manager.connect(websocket)
    await websocket.send_text(json.dumps({"event": "connected", "data": "connected to se
    try:
        while True:
            data = await websocket.receive_text()
            data = json.loads(data)
            event = data['event']
            msg = data['data']




            if(event == "onConnection"):

                await manager.broadcast(json.dumps({"connection": "connected"}))




            elif(event == 'start'):

                isConfigured = True
                tempTrip = str(msg)
                gps.setTripName(str(msg))
                drive_in_use = await createImageFolder(msg)
                if drive_in_use =="failed":
                    await manager.broadcast(json.dumps({"event": "error","data":"no driv

                else:
```

```
            drive.value = drive_in_use
            await start_acquisition()
            await manager.broadcast(json.dumps({"event": "starting"}))




    elif(event =="pulse"):

        startPulse()
    elif(event == 'stop'):

        await abortStream()

        await manager.broadcast(json.dumps({"event": "stopping"}))

    elif(event == "init"):

        camerasDetected = []
        await initCameraA()
        await initCameraB()
        await initCameraC()



        await manager.broadcast(json.dumps({"event":"detected","data":len(camera
```

```
elif(event == "validation"):
    await validate(msg)

elif(event =="gps"):
    print(msg)

elif(event == "start_acquisition"):
    status = start_acquisition()

elif(event == "create_trip"):
    await createImageFolder(data)
    await manager.broadcast(json.dumps({"event":"folderCreated"}))

elif(event == "toggleGps"):
    toggleGPSControl(msg)

elif(event == "live"):
    valider = msg
    viewer.value = int(msg)
    capturing.value = int(msg)



elif(event == "debug"):
    gps.setDebug(msg)
    debug= msg

elif(event=="search"):
```

```
            n =discoverCamerasLength()
            i =0


            print("Cameras:"+ str(n))
            for device in range(int(n)):
                camerasDetected.append(str(i))



                i=i+1
            message = json.dumps({"event":"search","data":len(camerasDetected)})
            await manager.send_personal_message(message, websocket)


        elif(event == "guru"):
            guruMode = msg

        elif(event=="initGPS"):
            initGPS()

        elif(event == "pause"):
            pause()

        elif(event =="reset"):
            startfps()
            resett()

            images = getImages()
```

```
                await manager.broadcast(json.dumps({"event":"stopped","data":images}))
                states = getStates()
                await manager.broadcast(json.dumps({"event":"states","data":states}))

            elif (event =="emergency"):
                emergencyStop()
                states = getStates()

                await manager.broadcast(json.dumps({"event":"states","data":states}))

            elif(event=="log"):
                log(msg)

            elif(event=="states"):

                states = getStates()

                await manager.broadcast(json.dumps({"event":"states","data":states}))




    except (WebSocketDisconnect,RuntimeError) as e:   # WebSocketDisconnect

        print("[WEBSOCKET] websocket disconnect")

        await manager.disconnect(websocket)
```

```
@app.on_event("startup")
async def startup():
    global camera_1,camera_2,camera_3,cameras,camerasDetected
    print("[startup] init cameras")
```

```
@app.on_event("shutdown")
def shutdown_event():
    global imagesave,imagesave2 ,closeServer,gps

    print("shutting down server")
    gps.raise_exception()
    gps.join()
```

```
def main(arg):

    #start the API server
```

```
    uvicorn.run(app, host="10.0.222.1", port=8000,log_level="error")


if __name__ == "__main__":

    # define all shared memory variables
    capturing = Manager().Value('i',0)
    stream1_Stopped = Manager().Value('i',0)
    stream2_Stopped = Manager().Value('i',0)
    stream3_Stopped = Manager().Value('i',0)
    stopStream = Manager().Value('i',0)
    viewer = Manager().Value('i',0)
    index1 = Manager().Value('i',0)
    index2 = Manager().Value('i',0)
    index3 = Manager().Value('i',0)
    save = Manager().Value('i',0)


    camera_1_Connection1,camera_1_Connection2  = Pipe(duplex=True)
    camera_2_Connection1,camera_2_Connection2  = Pipe(duplex=True)
    camera_3_Connection1,camera_3_Connection2  = Pipe(duplex=True)

    init1 = Manager().Value('i',0)
    init2 = Manager().Value('i',0)
    init3 = Manager().Value('i',0)


    drive = Manager().Value(c_wchar_p, "drive")
    nameTrip = Manager().Value(c_wchar_p, "trip")
```

```
#construct the three camerastream processes
stream0 = CameraStream(2,"3",capturing,stream3_Stopped,index3,drive,nameTrip,stopStr
stream1 = CameraStream(1,"2",capturing,stream2_Stopped,index2,drive,nameTrip,stopStr
stream2 = CameraStream(0,"1",capturing,stream1_Stopped,index1,drive,nameTrip,stopStr

#start the processes
stream0.start()
stream1.start()
stream2.start()


# start the GPS handler thread
gps = GpsHandler(debug)
gps.daemon = True
gps.start()

#connect to serial
serial = SerialService()
serial.connect()



main(debug)


# isi 192.168.10.153
#169.254.108.159
```

```
    #192.168.0.100
```

cameraStream.py

```
import os
from os import path
from threading import Thread
import cvb
import uuid
import queue
from core.timer import Timer
import time
from multiprocessing import Process,Manager,Queue
from core.camera import Camera
from core.FPS import FPS
from datetime import datetime
from core.imageSave import ImageSave
import cv2



class CameraStream(Process):

    def __init__(self, port,name,capturing,streamStopped,index,drive,tripName,stopStream
        super(CameraStream,self).__init__()
        self.daemon =True
        self.viewerPipe = pipe
        self.camera = Camera(port)
        self.savingQueue = queue.Queue()
        self.name=name
        self.trip = tripName
```

```
        self.drive = drive
        self.imagesave= ImageSave(self.savingQueue,save)
        self.capturing = capturing
        self.port = port
        self.image_name = index
        self.list_of_images = []
        self.isInit = False
        self.streamStopped=streamStopped
        self.drive = drive
        self.stopstream = stopStream
        self.rate_counter = cvb.RateCounter()
        self.init_camera = init
        self.viewer = viewer
        self.wait = wait




    def init(self):




        self.camera.init()


    def start_stream(self):


        self.camera.start_stream()
```

```
    def startCapturing(self):
        self.capturing =True


    def stopCapturing(self):
        self.capturing =False


    def getDate(self):


        return datetime.today().strftime('%Y-%m-%d')
    def resetIndex(self):
        self.image_name =0

    def setInit(self):
        self.isInit = True



    def run(self):

        #using sleep when initian cameras. CVB https://forum.commonvisionblox.com/t/acqu
        time.sleep(self.wait)
        init =self.camera.init(self.port)
```

```
        #if camera not init dont start the saving thread
        if not init:
            return "not started"




        self.imagesave.start()
        self.camera.start_stream()

        self.init_camera.value =1
        prev_frame_time = 0



        new_frame_time = 0
        date = self.getDate()
        fps = FPS().start()
        print(f'stream running: {self.camera.isRunning()}')
        while init:


            try:
                if bool(self.capturing.value):

                    image, status = self.camera.get_image()
```

```
                if status == cvb.WaitStatus.Ok:

                    cameraStamp = int(image.raw_timestamp/1000)
                    timeStamp = int(time.time() * 1000)




                    if self.image_name.value >0 :
                        newstamp =  starttime+ (cameraStamp-firstCameraStamp)



                        if not self.viewer.value:
                            data = {"image": image, "camera": self.name, "index": se
                            self.sendToSaving(data)
                        else:
                            imageArray = cvb.as_array(image,copy=True)
                            self.sendToViewer(imageArray)

                        self.rate_counter.step()


                        self.image_name.value = self.image_name.value +1

                    if self.image_name.value ==0:
                        print("reset index")
                        starttime = timeStamp
                        firstCameraStamp = cameraStamp
```

```
                self.image_name.value = self.image_name.value +1



        elif status == cvb.WaitStatus.Abort :
            print("stream 2 abort")



        elif status == cvb.WaitStatus.Timeout and bool(self.stopstream.value
            print("stopped stream")

            print("Acquired with: " + str(self.rate_counter.rate) + " fps")

            self.streamStopped.value =1

            self.capturing.value =0



        elif  status == cvb.WaitStatus.Timeout:
            print("timeout")


        else:
```

```
                    pass



            except Exception as e:
                print(e)
                pass


        self.camera.stopStream()


    def stop(self):
        self.running = False


    def sendToSaving(self,data):
        self.savingQueue.put(data)

    def sendToViewer(self,image):
        self.viewerPipe.send(image)



    def getBufferImage(self):

        if self.lastBufferImage is not self.bufferImage:
            self.lastBufferImage = self.bufferImage
        return self.bufferImage
```

```
    def reset(self):
        self.image_name.value = 0


    def getDevice(self):
       return self.camera.getDevice()




    def isRunning(self):
        return self.camera.isRunning()
```

imageSave.py

```
from threading import Thread

from datetime import datetime
from core.timer import Timer
import os
from multiprocessing import Process
from os import path
from datetime import datetime
import cvb
import ctypes
import csv
import json
from core.helpers.helper_server import most_free_space
from multiprocessing import Process,Manager
import cv2
```

```
import time
import sys

class ImageSave(Thread):
    def __init__(self, imageQueue,save):
        super(ImageSave,self).__init__()
        self.deamon = True
        self.tripName = "first"
        self.image_queue = imageQueue
        self.isRunning = True
        self.save = save
        self.path = os.path.dirname(os.path.abspath(__file__))



        self.path = self.fixPath(self.path)
        self.saving = False
        self.date = self.getDate()
        self.storageLeft = 50



    def fixPath(self,path):

            test = path.split("\\")
            if len(test)>1:

                test.pop()
                newPath = '/'.join(test)
                return newPath
            else:
                path = path.split("/")
```

```
            path.pop()
            path = '/'.join(path)



            return path
    def run(self):

        i = 0
        date = self.getDate()

        try:
            while True:

                #print(self.queue.qsize())
                if self.image_queue.empty():



                    pass
                else:




                    try:
                        if bool(self.save.value):
                            data = self.image_queue.get(timeout=500)
```

```
image = data['image']
camera = data['camera']
index = data['index']
timestamp = data['timeStamp']
cameraStamp =data['cameraStamp']
tripName = data['tripName']
drive = data['drive']

image_name = str(index)+"_"+str(timestamp)

try:
    imageArray = cvb.as_array(image,copy=True)
    imageArray = cv2.cvtColor(imageArray, cv2.COLOR_BGR2RGB)

    image_path = drive+"/"+"bilder/"+str(date)+"/"+str(tripN


    cv2.imwrite(image_path,imageArray)
except Exception:
    sys.stderr.write("[SAVING THREAD  ERROR]"% (type(e).__na


if int(camera) ==1:


    path = self.path+"/log"+"/"+self.date+"/"+str(tripName)+
    write_header = not os.path.exists(path)
    with open(path,'a',newline='')as csvfile:
```

```
                                fieldnames = ['index', 'tripname', "camera","raw_sta
                                writer = csv.DictWriter(csvfile, fieldnames=fieldnam
                                if write_header:
                                    writer.writeheader()

                                row = ({'index':str(index),'tripname':tripName,"came
                                writer.writerow(row)

                        else:
                            self.image_queue.get(timeout=500)


                    except Exception as e:


                        sys.stderr.write("[SAVING THREAD  ERROR]"% (type(e).__name__, e)


                    finally:
                        pass




            except Exception as e:
                print("[saving thread]:  "+e)
                pass
```

```
        return "Stopped"

    def stop(self):

        self.isRunning = False

    def setDrive(self,drive):

        self.drive = drive


    def getDate(self):

        return datetime.today().strftime('%Y-%m-%d')


    def setStorageLeft(self,storage):

        self.storageLeft = storage



    def saveImages(self):
        self.saving = True

    def getTimeStamp(self,now):



        timenow = datetime.today().strftime('%H:%M:%S')
```

```
        milliseconds = '%03d' % int((now - int(now)))
        return str(timenow) +":"+ str(milliseconds)

    def createFolder(self):
        date = self.getDate()

            #   Making a folder for the images
        if not path.exists('log/'+date):

            os.mkdir('log/'+date)

    def setTripName(self,name):

        self.tripName = name



    def raise_exception(self):
        thread_id = self.get_id()
        res = ctypes.pythonapi.PyThreadState_SetAsyncExc(thread_id,
              ctypes.py_object(SystemExit))
        if res > 1:
            ctypes.pythonapi.PyThreadState_SetAsyncExc(thread_id, 0)
            print('Exception raise failure')


    def get_id(self):

        # returns id of the respective thread
        if hasattr(self, '_thread_id'):
```

```
            return self._thread_id
        for id, thread in threading._active.items():
            if thread is self:
                return id
```

gps.py

```
from multiprocessing import Process
from threading import Thread
import requests
import pynmea2, serial, os, time, sys, glob, datetime
from datetime import datetime
import json
import enum
import io
import os
import csv


class GPS_QUALITY(enum.Enum):
   BEST = 3
   GOOD = 2
   LOW = 1
   BAD = 0

class Error(Exception):
    """Base class for other exceptions"""
    pass

class InitGPS(Error):
    """Raised when initilizing start to break loop"""
```

```
    pass


class GpsHandler(Thread):
    def __init__(self,debug):
        super(GpsHandler,self).__init__()
        self.debug = debug
        self.message  ={"quality":0,"satelites":0,"velocity":0,"timestamp":"","gpsTime":
        self.data = {"quality":0,"satelites":0,"velocity":0,"timestamp":"","gpsTime":"",
        self.logging = False
        self.path = os.path.dirname(os.path.abspath(__file__))
        self.tripName =""
        self.date = self.getDate()
        self.serial = None
        self.init = False
        self.connected = False
        self.initStarted = False


    def run(self):



        try:
            while True:
                ports = self.scan_ports()
                if len(ports) == 0:
                    if self.debug:
                        sys.stderr.write('No ports found, waiting 5 seconds...press Ctrl
                    time.sleep(5)
                    continue
```

```
for port in ports:
# try to open serial port
    if self.debug:
        sys.stderr.write('Trying port %s\n' % port)
    try:
# try to read a line of data from the serial port and parse
        with serial.Serial(port, 115200, timeout=5) as ser:

            self.serial = ser
            self.read = io.TextIOWrapper(io.BufferedRWPair(ser, ser))
              #sends commands to gps
            if self.init:
                self.initGPS()
                self.init = False
# 'warm up' with reading some input
            for i in range(10):
                data = ser.readline()
                #print(data)
            # try to parse (will throw an exception if input is not vali

            pynmea2.parse(ser.readline().decode('ascii', errors='replace


            self.connected = True
            while True:


                if self.init:
                    raise InitGPS
```

```
line = self.read.readline()
#.decode('ascii', errors='replace')


if not line=="":

    line = self.__trimLine(line)
    #print(line)
    try:

        msg = pynmea2.parse(line)
        #print(msg)
    except:
        pass



    message = self.createMessage(msg)
    if message =="no_data":
        pass
    else:

        self.data = message



        if(self.logging and self.data['new'] ==True):
            path= self.path+"/log"+"/"+self.date +"/"+se
            write_header = not os.path.exists(path)
            with open(path,'a',newline='')as csvfile:
```

```
                                    fieldnames = ['tripname','quality', 'vel
                                    writer = csv.DictWriter(csvfile, fieldna
                                    if write_header:
                                        writer.writeheader()

                                    row = ({'tripname':self.tripName,'qualit
                                    writer.writerow(row)

                        else:
                            raise Exception("no new lines")

                except Exception as e:
                    if self.debug: sys.stderr.write('Error reading serial port %s: %
                    self.data = {"quality":0,"velocity":0,"timestamp":"","gpsTime":"
                except KeyboardInterrupt as e:
                    pass
                    sys.stderr.write('Ctrl-C pressed, exiting log of %s to %s\n' % (
```

```
                    except InitGPS:
                        print("Initilizing the GPS")
                        break
                if self.debug: sys.stderr.write('Scanned all ports, waiting 5 seconds...
                self.data = {"quality":0,"velocity":0,"timestamp":"","gpsTime":"","lat":
                time.sleep(3)
                self.connected = False
        except KeyboardInterrupt:
            self.serial.close()
            self.read.close()
            #sys.stderr.write('Ctrl-C pressed, exiting port scanner\n')

    def getData(self):

        return self.data

    def setTripName(self,tripName):
        self.tripName = tripName

    def getDate(self):

        return datetime.today().strftime('%Y-%m-%d')


    def setDebug(self,value):
        print(value)

        self.debug = value
```

```
    def convertDatetime(self,dt):
        now = datetime.now()
        x = datetime(now.year, now.month, now.day,now.hour,dt.minute,dt.second,dt.micros


        timestamp = datetime.timestamp(x)



        return int(timestamp*1000)



    def isConnected(self):
        return self.connected

    def getTimeStamp(self):

        now = time.time()

        timenow = datetime.today().strftime('%H:%M:%S')
        milliseconds = '%03d' % int((now - int(now)) * 1000)
        return str(timenow) +":"+ str(milliseconds)

    def createMessage(self,msg):
        now = self.getTimeStamp()
        milliseconds = int(time.time() * 1000)
        fix_quality = 0
        hdop = 5
        velocity = 0
        gpstime = ""
        gpstimes = ""
```

```
if(msg.sentence_type =="RMC"):
    update = False
    if self.debug:

        print("RMC received")
        print(msg.timestamp)

    if msg.status =="A":
        velocity = self.__knotsToKmh(msg.spd_over_grnd)
        velocity = self.__kmhToMs(velocity)
        gpstime = msg.timestamp
        update =True

        gpstimes = self.convertDatetime(gpstime)


    self.message.update({"velocity":str(velocity),"timestamp":str(gpstime),"gpsT
    return self.message




if(msg.sentence_type=="GGA"):
    if self.debug:

        print("GGA received")
        print(msg)
```

```
            satelites = msg.num_sats
            fix_quality = int(msg.gps_qual)

            try:
                hdop = float(msg.horizontal_dil)
            except:
                pass


            quality = self.getGpsQuality(fix_quality,hdop)

            self.message.update({"quality":int(quality),"satelites":satelites,"new":Fals
            return self.message




        return "no_data"

    def getGpsQuality(self,fixQuality, hdop):
        gpsQuality = GPS_QUALITY.BAD.value
        if (fixQuality and hdop > 0):
            if (fixQuality == 4):
                gpsQuality = GPS_QUALITY.BEST.value
            elif ((fixQuality == 5 and hdop < 2) or hdop <= 1):
                gpsQuality = GPS_QUALITY.GOOD.value
            else:
```

```
                gpsQuality = GPS_QUALITY.LOW.value


        return gpsQuality


    def toggleLogging(self,value):

        self.logging = value

    def send(self, cmd):
        # self.gps.write_line(cmd)
        try :
            self.serial.write(cmd.encode())
        except:
            sys.stderr.write('Not Connected To GPS')
            #self.serial.close()

        time.sleep(3)

        num = self.serial.inWaiting()
        # try:
        #     print(self.gps.read(num).decode())
        # except:
        #     print(self.gps.read(num))
        recieved = self.serial.read(num)
        if self.debug:
            print(recieved.decode('utf-8'))
```

```
    def startInit(self):


        self.init =True
    def initGPS(self):
        self.initStarted =True
        self.send('SetNMEAOutput, Stream1+Stream2+Stream7+Stream8, none, none, off\r\n')
        self.send('SetGPIOFunctionality, GP1, Output, none, LevelLow\r\n')
        self.send('SetGPIOFunctionality, GP2, Output, none, LevelHigh\r\n')
        self.send('SetGPIOFunctionality, GP3, Output, none, LevelHigh\r\n')
        self.send('setDataInOut, COM1,DC1,DC2\r\n')
        self.send('setDataInOut, COM3,DC2,DC1\r\n')
        self.send('#PSPO,1\n')
        self.send('SSSSSSSSSS\r\n')
        self.send('setDataInOut, COM1, CMD, SBF+NMEA\r\n')
        self.send('SSSSSSSSSS\r\n')
        self.send('SetSBFOutput,Stream1+Stream2+Stream3+Stream4+Stream5,none,none,off\r\
        self.send('setPVTMode, Rover, all\r\n')
        self.send('sem,+PVT,10\r\n')
        self.send('setDataInOut, COM1, CMD, SBF\r\n')
        self.send('setGeoidUndulation, auto\r\n')
        self.send('setSmoothingInterval,all,100,1\r\n')
        self.send('setRAIMLevels,on,-4,-4,-3\r\n')
        self.send('setPVTMode, Rover, all\r\n')
        self.send('setFixReliability, RTK, 0.2, 4.40\r\n')
        self.send('setDiffCorrUsage,,,auto,0\r\n')
        self.send('setDataInOut, COM2, RTCMv3, SBF+NMEA\r\n')
```

```
        self.send('setDataInOut, COM2,DC1,DC2\r\n')
        self.send('setDataInOut, COM3,DC2,DC1\r\n')
        self.send('+++\r\n')
        self.send('AT\r\n')
        self.send('+++\r\n')
        self.send('AT\r\n')
        self.send('+++\r\n')
        self.send('ATE1\r\n')
        self.send('AT+MGEER=2\r\n')
        self.send('AT+CMEE=2\r\n')
        self.send('AT+CBST?\r\n')
        self.send('AT+CPIN?\r\n')
        self.send('AT+CSQ\r\n')
        self.send('AT+MIPCALL?\r\n')
        self.send('AT+MIPCALL=1,"telenor","","\r\n')
        self.send('AT+MIPOPEN?\r\n')
        self.send('AT+MIPODM=1,1200,"159.162.103.14",2101,0\r\n')
        self.send('GET /CPOSHREF HTTP/1.0\r\n')
        self.send('User-Agent: NTRIP Altus\r\n')
        self.send('Authorization: Basic NTgwMDAwNzEzNTMzOkdKRVJERTEzMDU=\r\n')
        self.send('Accept: */*\r\n')
        self.send('Connection: close\r\n')
        self.send('\r\n')
        self.send('\r\n')
        self.send('SSSSSSSSSS\r\n')
        self.send('setDataInOut, COM3, CMD, SBF+NMEA\r\n')
        self.send('setDataInOut, COM2, RTCMv3, SBF+NMEA\r\n')
        self.send('SetNMEAOutput, Stream1, COM2, GGA, sec1\r\n')
        self.send('SetNMEAOutput, Stream8, COM3, GGA+VTG+RMC, msec100\r\n')
```

```
        self.initStarted = False

    def scan_ports(self):

        if sys.platform.startswith('win'):
            ports = ['COM%s' % (i + 1) for i in range(6)]
        elif sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):
            # this excludes your current terminal "/dev/tty"
            patterns = ('/dev/ttyUSB*','' )
            ports = [glob.glob(pattern) for pattern in patterns]
            ports = [item for sublist in ports for item in sublist]  # flatten
        elif sys.platform.startswith('darwin'):
            patterns = ('/dev/*serial*', '/dev/ttyUSB*')
            ports = [glob.glob(pattern) for pattern in patterns]
            ports = [item for sublist in ports for item in sublist]  # flatten
        else:
            raise EnvironmentError('Unsupported platform')
        return ports




    def __calculateFrequency(self, distance):

        factor = 3.6
        velocityInMs = self.velocity / factor
        result = velocityInMs / distance
        return result
```

```
def __knotsToKmh(self, knots):
    factor = 1.852
    result = factor * knots
    return result



def __kmhToMs(self,speedKmh):

    return int(speedKmh/3.6)



def setClock(self,time):

    gpsutc = time.year+time.month+time.day+(time.second) + (time.microsecond)

    os.system('sudo date -u -set="%s"' % gpsutc)

def raise_exception(self):
    thread_id = self.get_id()
    res = ctypes.pythonapi.PyThreadState_SetAsyncExc(thread_id,
          ctypes.py_object(SystemExit))
    if res > 1:
        ctypes.pythonapi.PyThreadState_SetAsyncExc(thread_id, 0)
        print('Exception raise failure')


def get_id(self):
```

```
        # returns id of the respective thread
        if hasattr(self, '_thread_id'):
            return self._thread_id
        for id, thread in threading._active.items():
            if thread is self:
                return id

    def __trimLine(self, msg):
        message = ''
        message = msg
        start = message.find('$')
        # -7 removes checksum, -5 if we add a test on the checksum
        end = len(message)-5
        result = ''
        result = message[start:end]
        return result
```

helperserver.py

```
import cvb
import cv2
import base64
import os,string
from os import path
import os
from datetime import datetime
import math
import uuid
from starlette.websockets import WebSocket, WebSocketDisconnect
import time
```

```
import shutil
import sys
import getpass



class ConnectionManager:
    """class that handles the socket connections


    """
    def __init__(self):
        self.active_connections: List[WebSocket] = []

    async def connect(self, websocket: WebSocket):
        try:
            await websocket.accept()
            self.active_connections.append(websocket)

        except Exception as e:
            print(e)

    async def disconnect(self, websocket: WebSocket):
        self.active_connections.remove(websocket)

    async def send_personal_message(self, message: str, websocket: WebSocket):
        await websocket.send_text(message)

    async def broadcast(self, message: str):
        for connection in self.active_connections:
```

```
            await connection.send_text(message)

def cvbImage_b64(frame):
    """Return a b64 string

    parameter: cvb Image
    """

    image_np = cvb.as_array(frame, copy=False)
    image_np = cv2.cvtColor(image_np, cv2.COLOR_BGR2RGB)
    image_np = cv2.resize(image_np, (640, 480))
    _, frame = cv2.imencode('.jpg', image_np)
    im_bytes = frame.tobytes()
    im_b64 = base64.b64encode(im_bytes)
    base64_string = im_b64.decode('utf-8')

    return base64_string


def getTimeStamp():

    now = time.time()

    timenow = datetime.today().strftime('%H:%M:%S')
    milliseconds = '%03d' % int((now - int(now)) * 1000)
    return str(timenow) +":"+ str(milliseconds)


def checkStorageAllDrives():
    available_drives = []
```

```
nested_storage= {'drives':{} }


if sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):

    username = getpass.getuser()

    path = "/media/"+username
    directory_contents = os.listdir(path)

    for item in directory_contents:
        available_drives.append(path+"/"+str(item))



elif sys.platform.startswith('win'):
    for d in string.ascii_uppercase: # Iterating through the english alphabet
        path = '%s:' % d
        if os.path.exists(path): # checks if path exists
            available_drives.append(path)  #  append the path from a drive to a list



try:

    number= 1 # Variable for
    for i in available_drives:  #  Iterating through all drives

        total, used, free = shutil.disk_usage(i) # Return disk usage statistics abou
```

```
            dictname = "hd"+ str(number)
            nested_storage['drives'].update ({dictname:{ "name": i,"total": "%d" % (tota
            , "used": "%d" % (used // (2**30)), "free": "%d" % (free // (2**30))}})  #
            number += 1



    except:
        pass

    finally:

        return nested_storage




def most_free_space():
    onedrive={}
    useableDrives = {}
    drives = checkStorageAllDrives()
    drives = drives['drives']



    for drive in drives:

        hd = drives[drive]
```

```
        if(hd['name']=='C:'):
            pass

        else:
            useableDrives.update({hd['name']:hd})
            onedrive = hd
    bestDrive ={}
    if len(useableDrives)> 1:




        for drive in useableDrives:
            drive1 = useableDrives[drive]
            free = drive1['free']
            for drives in useableDrives:
                drive2 = useableDrives[drives]
                compare = int(drive2['free'])
                if int(free) > compare:
                    bestDrive = drive1

                elif int(free)< compare:
                    bestDrive = drive2

    else:

        bestDrive = onedrive
```

```
    return bestDrive




def getExternalStorage():
    available_drives = []


    for d in string.ascii_uppercase: # Iterating through the english alphabet
        path = '%s:' % d
        if os.path.exists(path): # checks if path exists
            available_drives.append(path)  #  append the path from a drive to a list


    return available_drives


def createFolder():

    """Creates a folder at startup in 'bilder' with todays date

    parameter: cvb Image
    """
    date = getDate()
```

```
        #   Making a folder for the images
    if not path.exists('bilder/'+date):

        os.mkdir('bilder/'+date)


async def createImageFolder(tripName):
    path2 = os.path.dirname(os.path.abspath(__file__))

    path2 = fixPath(path2)

    print(path2)

    date = getDate()

    if not path.exists(str(path2)+'/log/'+str(date)+"/"+str(tripName)):


        os.makedirs(str(path2)+'/log/'+str(date)+"/"+str(tripName))


    bestDrive = most_free_space()
    if bestDrive:
        drive = bestDrive['name']
        print(drive)

        if not path.exists(drive+"/"+"bilder/"+str(date)+"/"+str(tripName)+"/"+"kamera1"
```

```
            os.makedirs(drive+"/bilder/"+str(date)+"/"+str(tripName)+"/"+"kamera1")


        if not path.exists(drive+"/bilder/"+str(date)+"/"+str(tripName)+"/"+"kamera2"):


            os.makedirs(drive+"/bilder/"+str(date)+"/"+str(tripName)+"/"+"kamera2")

        if not path.exists(drive+"/bilder/"+str(date)+"/"+str(tripName)+"/"+"kamera3"):


            os.makedirs(drive+"/bilder/"+str(date)+"/"+str(tripName)+"/"+"kamera3")









        return drive

    else: return "failed"

def estimateStorageTime(storages,fps):
    # 20 bilder/sek
    # 1 bilde 8000kB

    total = 0

    bilder_pr_sek = 20
    bilde_size = 6  # Mb
```

```
for storage in storages['drives']:



    free = int(storages['drives'][storage]["free"])
    if storages['drives'][storage]['name'] == "C:":

        pass
    else:

        total = total + free

    free = free*1000
    seconds_left = free/(bilder_pr_sek*bilde_size)
    minleft = math.floor(seconds_left/60)
    hoursLeft = int(minleft/60)
    minleft = minleft % 60

    storages['drives'][storage]['h'] = hoursLeft
    storages['drives'][storage]['m'] =minleft



storages['total'] = {'free':total,'timeleft':{}}
free = total*1000
seconds_left = free/(bilder_pr_sek*bilde_size)
minleft = math.floor(seconds_left/60)
hoursLeft = int(minleft/60)
minleft = minleft % 60
```

```
    storages['total'].update({'timeleft':{'h':hoursLeft,'m':minleft}})

    storages['total'].update({'h': hoursLeft})
    storages['total'].update({'m':minleft})




    return storages



# def fixPath(path):

#     test = path.split("\\")
#     if len(test)>1:

#         test.pop()
#         test.pop()
#         newPath = '/'.join(test)

#         return newPath
#     else:

#         path = path.split("/")
#         path.pop()
#         path.pop()
```

```
#         path = '/'.join(path)

#         return path

def fixPathServer(path):

    test = path.split("\\")
    if len(test)>1:


        newPath = '/'.join(test)

        return newPath
    else:

        path = path.split("/")
        path = '/'.join(path)

        return path



def getDate():

    return datetime.today().strftime('%Y-%m-%d')



def uniqueID():
```

```
    return uuid.uuid4()




def discoverCameras():

    discover =[]

    try:

        discover = cvb.DeviceFactory.discover_from_root(cvb.DiscoverFlags.IgnoreVins,tim



    except Exception as e:
        print(e)

    finally:
        return discover




def discoverCamerasLength():
    discovered =0
```

```
    discover = cvb.DeviceFactory.discover_from_root(cvb.DiscoverFlags.IgnoreVins)

    if sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):
        discovered = len(discover)

    elif sys.platform.startswith('win'):

        discovered = int(len(discover)/2)


    return discovered
```

manager.py

```
from starlette.websockets import WebSocket, WebSocketDisconnect

# manage socket Connections


class ConnectionManager:
    def __init__(self):
        self.active_connections: List[WebSocket] = []

    async def connect(self, websocket: WebSocket):
        await websocket.accept()
        self.active_connections.append(websocket)

    def disconnect(self, websocket: WebSocket):
        self.active_connections.remove(websocket)

    async def send_personal_message(self, message: str, websocket: WebSocket):
```

```
        await websocket.send_text(message)

    async def broadcast(self, message: str):
        for connection in self.active_connections:
            await connection.send_text(message)
```

merging.py

```
import csv
from collections import defaultdict
from bisect import bisect_left
import collections
import os
from core.timer import Timer
from core.helpers.helper_server import *
#---start for variables and lists for merging csv---#

#----lists----#

#   image informastion lists:

picnumberlist = []
millisk1list = []

#   gps informastion lists
millisgpslist = []
longlist = []
latlist = []
fixlist = []
speedlist = []
```

```
#   Modified coordinate list
modyfiedlatlist = []
modyfiedlonglist = []

closesttimelist = []



expandedmodifiedlongitudelist = []
expandedmodifiedlatitudelist = []
#   gps dictionary

#   index lists

indexlist = []
closesttimelist = []
test = []
dublicateindexlist = []
dublicateindexvaluelist = []
newindexlist = []

timedifferencelist = []
#   values for iterating



before = None
after = None
```

```
#---end for variables and lists for merging csv---#


#-----------------------------------------#

#----start of functions for csv merging----#

#-----------------------------------------#

#----end of functions for csv merging----#
#  finds the closest time between image list and gps list
#  return list of index
def take_closest(myList, myNumber):
    """
    Assumes myList is sorted. Returns closest value to myNumber.

    If two numbers are equally close, return the smallest number.
    """
    pos = bisect_left(myList, myNumber)

    if pos == 0:
        return myList[0], pos
    if pos == len(myList):
        return myList[-1], pos
    before = myList[pos - 1]
    after = myList[pos]
    if abs(int(after) - int(myNumber)) < abs(int(myNumber) - int(before)):
        return after, pos
    else:
```

```
        return before, pos

#  compare time from pic list and gps list
#   return
def calculateindexposition(gpsmillislist, picmillislist):
    #fix so pic log can star before gps log


    indexlist.clear()
    testlist = [0]
    #before = gpsmillislist[0]
    for n in picmillislist:
        closesttime, position = take_closest(gpsmillislist, n)
        #if position != 0:
        #    position -= 1
        diff = n - closesttime
        before = position

        #print('diff= '+str(closesttime)+' - '+str(n))
        #print('index: '+str(gpsmillislist.index(n)) +' pos:'+str(position)+' indexgps:
        closesttimelist.append(diff)
        #print(gpsmillislist.index(closesttime))
        indexlist.append(gpsmillislist.index(closesttime))
        after = position

        #if position < 100:
        #   print('pos:'+str(position)+' timediff: '+str(diff))

    #print(indexlist)
    #print(closesttimelist)
```

```
    return indexlist, closesttimelist




def choosefromindex(valuelist1, wantindex):
    preflist = []
    for n in wantindex:
        preflist.append(valuelist1[n])

    return preflist



#   get time difference between picture taken and coordinate read
#   return time difference list
def gettimedifference(gpstimelist, pictimelist):
    timediff = []
    for n in range(len(gpstimelist)-1):
        diff = gpstimelist[n] - pictimelist[n]
        timediff.append(diff)
```

```
    return timediff




#   modifies coordinates based on time difference between picture taken and coordinate r
#   speed, and constant acceleration corresponding
#   return list of modified coordinates
def modifycoordinatesbasedontimedifference(index, timedifflist, coordiantelist, gpsmilli

    shiftedcoord = []


    for n in range(len(index)-1):
        time = timedifflist[n]
        i = index[n]







        if i != 0  and len(gpsmillis)-1 != i:
            if timedifflist[n] < 0:
                ratio = (timedifflist[n])/(gpsmillis[i]-gpsmillis[i-1])
                shifted = ratio*(coordiantelist[i]-coordiantelist[i-1])
            elif timedifflist[n] > 0:
                ratio = timedifflist[n]/(gpsmillis[i+1]-gpsmillis[i])
```

```
                shifted = ratio*(coordiantelist[i+1]-coordiantelist[i])



        #print(coordiantelist[n])
        shiftedcoord.append(shifted+coordiantelist[i])
    return shiftedcoord









#---------------------------------------#

#----end of functions for csv merging----#

#---------------------------------------#



#-----------------------------------------#

#---beginning of program for csv merging---#

#-----------------------------------------#
def merge(path):
```

```
try:
    # Open image csv file and make a csv reader object
    with open(path +'/images.csv', newline='') as csvpic:
        picreader = csv.reader(csvpic, delimiter=',', quotechar='|')
        next(picreader)
        for row in picreader:
            #  make a list for each column in the csv file
            picnumberlist.append(row[0])
            millisk1list.append(int(row[4]))




    # Open gps csv file and make a csv reader object
    with open(path +'/gps.csv', newline='') as csvgps:
        gpsreader = csv.reader(csvgps, delimiter=',', quotechar='|')
        next(gpsreader)
        for row in gpsreader:
            #  make a list for each column in the csv file
            fixlist.append(row[1])
            speedlist.append(float(row[2]))
            millisgpslist.append(int(row[4]))
            latlist.append(float(row[5]))
            longlist.append(float(row[6]))

#for testing

except Exception as e:
    print(e)
```

```
print('millispic: '+ str(len(millisk1list)))
print('lat: '+ str(len(latlist)))

matchinglatitudelist = []
matchinglongitudelist = []
matchingspeedlist = []
matchinggpsmillislist = []
matchingfixlist = []
matchingnumberlist = []

indexlist, closesttimelist = calculateindexposition(millisgpslist, millisk1list)


matchinggpsmillislist = choosefromindex(millisgpslist,indexlist)
matchinglatitudelist = choosefromindex(latlist,indexlist)
matchinglongitudelist = choosefromindex(longlist,indexlist)
matchingspeedlist = choosefromindex(speedlist,indexlist)
matchingfixlist = choosefromindex(fixlist,indexlist)



timedifferencelist = gettimedifference(matchinggpsmillislist, millisk1list)

#   Modifies latitude and longitude list
modyfiedlatlist = modifycoordinatesbasedontimedifference(indexlist, closesttimelist,
modyfiedlonglist = modifycoordinatesbasedontimedifference(indexlist, closesttimelist
```

```
    with open(path+ '/merged.csv','w', newline='') as test1:
        fieldnames = ['number','picmillis','gpsmillis','latitude','longitude', 'modyfied
        writer = csv.DictWriter(test1, fieldnames=fieldnames)

        for n in range(len(millisk1list)-1):

            writer.writerow({'number':picnumberlist[n],'picmillis': millisk1list[n],'gps
            'latitude':matchinglatitudelist[n],'longitude':matchinglongitudelist[n],
            'modyfiedlat': modyfiedlatlist[n], 'modyfiedlong': modyfiedlonglist[n], 'fix

    with open(path+ '/mapping.csv','w', newline='') as test2:
        fieldnames = ['lat','lng']
        writer1 = csv.DictWriter(test2, fieldnames=fieldnames)
        writer1.writeheader()
        for n in range(550,749):
            writer1.writerow({'lat':matchinglatitudelist[n],'lng':matchinglongitudelist[

    with open(path+ '/mappingmoved.csv','w', newline='') as test3:
        fieldnames = ['lat','lng']
        writer2 = csv.DictWriter(test3, fieldnames=fieldnames)
        writer2.writeheader()
        for n in range(550,749):
            writer2.writerow({'lat':modyfiedlatlist[n],'lng':modyfiedlonglist[n]})



#----------------------------------------#
```

```
#---end of program for csv merging---#

#-----------------------------------------#



# absolute_path = os.path.dirname(os.path.abspath(__file__))
# absolute_path = fixPath(absolute_path)

# path = absolute_path+"/log/"+"2021-05-14/Oppover"

# t = Timer("merging")
# t.start()
# merge(path)
# t.stop()
```

serialService.py

```
import serial
from threading import Thread
import sys
class SerialService():
    def __init__(self):
        self.serial =None



    def connect(self,port="ttyACMo"):

        connected = False
```

```
        if sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):
            try:
                self.serial = serial.Serial(f'/dev/{port}',9600, timeout=1)
                print("success")
                connected = True

            except Exception as e:
                print(f"couldnt connect to arduino: {e}")



            finally:
                return connected


        elif sys.platform.startswith('win'):
            try:
                self.serial = serial.Serial(f'{port}',9600, timeout=1)
                print("success")
                connected = True

            except Exception as e:
                print(f"couldnt connect to arduino: {e}")



            finally:
                return connected
```

```
    def write(self,fps):



        try:

            self.serial.write(fps)

        except Exception as e:
            #print("failed to write serial port")
            pass



    def read(self):

        data = False

        try:

            data = self.serial.readline().decode()



        except Exception as e:
            #print("failed to read serial port")
            pass

        finally:
            return data
```

```
    def close(self):

        self.serial.close()

        return False
```

camera.py

```
import cvb
import os


class Camera:

    def __init__(self, port):
        self.port = port
        self.image = None
        self.device = None
        self.stream = None
```

```
        self.config_path ="core/config/conf.gcs"
        self.tempimg = None
        self.clock =None



    def loadConfig(self):


        self.device = cvb.DeviceFactory.open(os.path.join(
            cvb.install_path(), "drivers", "GenICam.vin"), port=self.port)


        self.stream = self.device.stream
        # self.device_node_map = self.device.node_maps["Device"]
        # self.device_node_map.load_settings(file_name=self.config_path)



    def init(self,name):

        init = False
        print(f'open port: {str(name)}')

        try:

            self.device = cvb.DeviceFactory.open_port(os.path.join(
                cvb.install_path(), "drivers", "GenICam.vin"), port=self.port)
```

```
            # self.device_node_map = self.device.node_maps["Device"]
            # self.clock = self.device_node_map['timestampControlReset']


            self.stream = self.device.stream
            init = True
        except Exception as e:
            print(e)
            print(f"failed opening camera: {str(name)}")


        finally:
            return init



    def getDevice(self):

        return self.device

    def init2(self,device):

        try:

            self.device = device
            test =self.device.read_property(cvb.DiscoveryProperties.DeviceId )

            print(test)
```

```
            self.stream = self.device.stream
            self.stream.start()

        except Exception as e:

            print(e)



    def start_stream(self):

        self.stream.start()


    def resetClock(self):
        self.clock.execute()

    def get_image(self):

        image, status = self.stream.wait_for(6000)
        return image, status

    def abortStream(self):
        self.stream.abort()
        self.device = None
        self.stream = None

    def stopStream(self):
        try:
            self.stream.stop()
```

```
        except Exception as e:
            pass

    def getPort(self):
        return self.port

    def terminate(self):
        self.device.close()

    def getSnapShot(self):
        image, status = self.stream.get_snapshot()
        return image, status

    def isRunning(self):
        isRunning = False
        try:
            isRunning = self.stream.is_running

        except Exception as e:
            pass

        finally:

            return isRunning
```

B GUI source code

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>GUIRailingApp</title>
    <base href="/" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="icon" type="image/x-icon" href="favicon.ico" />
    <link
      href="https://fonts.googleapis.com/css?family=Roboto:300,400,500&display=swap"
      rel="stylesheet"
    />
    <link
      rel="stylesheet"
      type="text/css"
      href="./node_modules/leaflet/dist/leaflet.css"
    />
    <link
      href="https://fonts.googleapis.com/icon?family=Material+Icons"
      rel="stylesheet"
    />
    <link rel="preconnect" href="https://fonts.gstatic.com" />
    <link
      href="https://fonts.googleapis.com/css2?family=Montserrat&display=swap"
      rel="stylesheet"
    />
    <link rel="preconnect" href="https://fonts.gstatic.com" />
```

```
    <link
      href="https://fonts.googleapis.com/css2?family=Open+Sans:wght@300&display=swap"
      rel="stylesheet"
    />
  </head>
  <body>
    <app-root></app-root>
  </body>
</html>
```

app.component

```
.mat-typography {
  background: lawngreen;
}

.mat-sidenav-container {
  height: 100%;
}

.mat-sidenav {
  width: 20%;
}

@media screen and (max-width: 768px) {
  .mat-sidenav {
    width: 50%;
  }
}

.content {
```

```
  width: 100%;
  height: 100%;
}

.homeicon{
  font-size: 140%;
}

.connection {

}
.cameraicon{

  margin-right: 1em;
}
.connectionicon{

  margin-right: 1em;

}

.home {

  right: 50%;
  cursor: pointer;
  position: absolute;

}
```

```
.toolbar {
  position: inherit;
  z-index: 2;
}
mat-toolbar {
  position: inherit;
  z-index: 2;
}
```

```
<mat-toolbar class="toolbar" color="dark">
  <div style="flex: 1 1 auto">
    <app-change-mode></app-change-mode>
  </div>
  <div matRipple class="home">
    <mat-icon
      class="homeicon"
      routerLink="/dash"
      routerLinkActive="activebutton"
      ><svg
        xmlns="http://www.w3.org/2000/svg"
        height="24"
        viewBox="0 0 24 24"
        width="24"
      >
        <path d="M0 0h24v24H0z" fill="none" />
        <path d="M10 20v-6h4v6h5v-8h3L12 3 2 12h3v8z" /></svg
    ></mat-icon>
  </div>

  <div class="connection">
```

```
    <!-- <mat-icon [matTooltip] ="cameraConnection" class="cameraicon" [ngStyle]="camera
    <mat-icon

    class="cameraicon"
    [matTooltip]="pulseToolTip"
    [ngStyle]="pulseStatus"
    *ngif =" pulseConnection; else disconnected"

  >
  <svg xmlns="http://www.w3.org/2000/svg" height="24px" viewBox="0 0 24 24" width="24px"
  </mat-icon>

  <mat-icon
  #disconnected
  class="cameraicon"
  [matTooltip]="pulseToolTip"
  [ngStyle]="pulseStatus"


>
<svg xmlns="http://www.w3.org/2000/svg" enable-background="new 0 0 24 24" height="24px"
</mat-icon>


    <mat-icon

      class="cameraicon"
      [matTooltip]="cameraConnection"
      [ngStyle]="camerastatus"
      (click)="testSearch()"
```

```
>
  <svg
    xmlns="http://www.w3.org/2000/svg"
    height="24"
    viewBox="0 0 24 24"
    width="24"
  >
    <circle cx="12" cy="14" r="3.2" />
    <circle cx="12" cy="14" fill="none" r="5" />
    <path
      d="M16 3.33c2.58 0 4.67 2.09 4.67 4.67H22c0-3.31-2.69-6-6-6v1.33M16 6c1.11 0 2
    />
    <path d="M24 0H0v24h24V0z" fill="none" />
    <path
      d="M17 9c0-1.11-.89-2-2-2V4H9L7.17 6H4c-1.1 0-2 .9-2 2v12c0 1.1.9 2 2 2h16c1.1
    />
  </svg>
</mat-icon>
<mat-icon
  class="connectionicon"
  [matTooltip]="connection"
  [ngStyle]="connectionStatus"
  (click) ="connect()"
>
  <svg
    xmlns="http://www.w3.org/2000/svg"
    height="24"
    viewBox="0 0 24 24"
    width="24"
  >
```

```
        <path d="M0 0h24v24H0z" fill="none" />
        <path
          d="M12 11c-1.1 0-2 .9-2 2s.9 2 2 2 2-.9 2-2-.9-2-2-2zm6 2c0-3.31-2.69-6-6-6s-6
        />
      </svg>
    </mat-icon>
  </div>
  <button mat-icon-button (click)="sideNav.toggle()">
    <mat-icon>
      <svg xmlns="http://www.w3.org/2000/svg" height="24" viewBox="0 0 24 24" width="24"
    </mat-icon>
  </button>
</mat-toolbar>

<mat-sidenav-container>
  <mat-sidenav #sideNav mode="push" position="end">
    <mat-toolbar class="toolbar" color="dark">
      <h1>Naviger</h1>
      <div style="flex: 1 1 auto"></div>
      <button mat-icon-button (click)="sideNav.toggle()">
        <mat-icon>
          <svg xmlns="http://www.w3.org/2000/svg" height="24px" viewBox="0 0 24 24" widt
        </mat-icon>
      </button>
    </mat-toolbar>
    <div class="items">
      <mat-nav-list>
        <mat-list-item
        routerLink="/dash" routerLinkActive="activebutton">
          <!-- <button mat-button routerLink="/dash" routerLinkActive="activebutton">
```

```
            Dashboard
          </button> -->
          Dashboard
        </mat-list-item>
        <mat-list-item routerLink="/map"
        routerLinkActive="activebutton">

          Kart
        </mat-list-item>
        <mat-list-item
        routerLink="/config"
          routerLinkActive="activebutton">
          Konfig
        </mat-list-item>
        <mat-list-item routerLink="/camera"
        routerLinkActive="activebutton">

          Kamera visning
        </mat-list-item>
      </mat-nav-list>
    </div>
  </mat-sidenav>

  <mat-sidenav-content>
    <div [@triggerName]="prepareRoute(outlet)">
      <router-outlet #outlet="outlet"></router-outlet>
    </div>
  </mat-sidenav-content>
</mat-sidenav-container>
```

```
import { Component, ViewChild } from '@angular/core';
import { catchError, map, takeUntil, tap } from 'rxjs/operators';
import { DataService } from './services/data.service';
import { Observable, of, Subject, Subscription } from 'rxjs';
import { webSocket } from 'rxjs/webSocket';
import { Store } from '@ngrx/store';
import { State,Cameras } from './store/reducers/connected.reducer';
import {Pulse} from './store/models/connected.model'
import { ColormodeService } from './services/colormode.service';
import {environment} from '../environments/environment.prod'
import { routeTransitionAnimations } from './route-transition-animations';
import { RouterOutlet } from '@angular/router';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
  animations:[routeTransitionAnimations]
})
export class AppComponent {
  title = 'GUI-Railing-App';
  public connected$!: Observable<State>;
  public camera$!: Observable<Cameras>;
  public pulse$!: Observable<Pulse>;
  connection = '';
  cameraConnection="Disconnected"
  pulseToolTip ="Ikke koblet til. prøv en annen port"
  pulseConnection = false
  hideDelay = 2000;
  subsciption = new Subject();
```

```
public connectionStatus = {
  color: '#ff5e57',
};

public camerastatus={
  color: '#ff5e57'
}

public pulseStatus={
  color: '#ff5e57'
}

constructor(
  private websocket: DataService,
  private colormode: ColormodeService,
  private store: Store<{ statusState: State,camera:Cameras,pulse:Pulse }>,

) {
  this.connected$ = store.select((state) => state.statusState);
  this.camera$ = store.select((state)=> state.camera)
  this.pulse$ = store.select((state)=>state.pulse )

  this.connected$.pipe(takeUntil(this.subsciption)).subscribe((data) => {

    if (data.status == 'Connected') {
      this.connection = 'Connected'
      this.connectionStatus.color = '#0be881';
    } else if (data.status == 'Disconnected') {
```

```
      this.connection = "Disconnected"
      this.connectionStatus.color = '#ff5e57';
    } else if (data.status == 'Reconnecting') {
      this.connection ="Reconnecting"
      this.connectionStatus.color = '#FFDC00';
    }
  });



  this.pulse$.pipe(takeUntil(this.subsciption)).subscribe((data)=>{


    if (data.pulse){

      this.pulseStatus.color ='#0be881'
      this.pulseToolTip ="Tilkoblet"

    }
    else{
    this.pulseStatus.color ='#ff5e57'
    this.pulseToolTip ="Ikke koblet til. prøv en annen port"

    }
  })

  this.camera$.pipe(takeUntil(this.subsciption)).subscribe((data)=>{
```

```
      if (data.connectionA=="ok" || data.connectionB=="ok"){

        this.cameraConnection = `connected: ${data.detected} cameras `
        this.camerastatus.color ='#0be881'
      }
      else if(data.connectionA=="config_failed" || data.connectionB=="config_failed"){
        this.cameraConnection ="Connect camera, press to reconnect"
        this.camerastatus.color = '#FFDC00'
      }

      else if(data.connectionA=="failed" && data.connectionB=="failed"){

        this.cameraConnection ="Disconnected"
        this.camerastatus.color = '#ff5e57'


      }
    })

    this.colormode.load();
  }

  /**
   * Prepares route
   * @param outlet will emit an activate event any time a new component is being instant
   *               and a deactivate event when it is being destroyed
   * @returns true if route, false otherwise
   */
   prepareRoute(outlet: RouterOutlet): boolean {
    return (
```

```
      outlet &&
      outlet.activatedRouteData &&
      outlet.activatedRouteData["animationState"]
    );
  }
  reconnect(){

    this.websocket.send({ event: 'init', data: '' });

  }
  connect(){

  }

  ngOnInit(): void {

  }
  testSearch(){

    this.websocket.send({event:"search",data:""})
  }

  ngOnDestroy(): void {
    //Called once, before the instance is destroyed.
    //Add 'implements OnDestroy' to the class.
    this.subsciption.unsubscribe()
    this.subsciption.complete()
  }
  ngAfterViewInit(): void {}
}
```

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { CameraViewComponent } from './components/camera-view/camera-view.component';
import { AppComponent } from './app.component';



const routes: Routes = [


  {
    path: 'dash',
    loadChildren: () => import('./dash/dash.module').then((m) => m.DashModule),
  },
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule],
})
export class AppRoutingModule {

}

import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

import { SidebarComponent } from './components/sidebar/sidebar.component';
```

```
import { HttpClientModule } from '@angular/common/http';
import { StoreModule } from '@ngrx/store';
import { reducer, camera, storageR,reducer2,statesReducer,_pulseReducer } from './store/
import { EffectsModule } from '@ngrx/effects';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatSliderModule } from '@angular/material/slider';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { MatSidenavModule } from '@angular/material/sidenav';
import { MatRadioModule } from '@angular/material/radio';
import { MatButtonModule } from '@angular/material/button';
import { effects } from './store/effects/connected.effect';
import { MatIconModule } from '@angular/material/icon';
import { MatGridListModule } from '@angular/material/grid-list';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatListModule } from '@angular/material/list';
import { ChangeModeComponent } from './components/change-mode/change-mode.component';
import { CameraViewComponent } from './components/camera-view/camera-view.component';
import { MatSlideToggleModule } from '@angular/material/slide-toggle';
import { DashModule } from './dash/dash.module';
import { MatTooltipModule } from '@angular/material/tooltip';
import { APP_INITIALIZER } from '@angular/core';
import { AppLoadModule } from './app-load/app-load.module';
import { DataService } from './services/data.service';
import { MatRippleModule } from '@angular/material/core';
import {
  HashLocationStrategy,
  LocationStrategy,
  PathLocationStrategy,
} from '@angular/common';
import { LogDialogComponent } from './components/log-dialog/log-dialog.component';
```

```
@NgModule({
  declarations: [
    AppComponent,
    SidebarComponent,
    ChangeModeComponent,
    CameraViewComponent,
    LogDialogComponent,




  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    MatSliderModule,
    MatTooltipModule,
    DashModule,
    MatSidenavModule,
    MatRadioModule,
    FormsModule,
    ReactiveFormsModule,
    MatButtonModule,
    MatGridListModule,
```

```
    MatToolbarModule,
    MatSlideToggleModule,
    MatIconModule,
    MatRippleModule,
    MatListModule,

    BrowserAnimationsModule,
    StoreModule.forRoot({
      statusState: reducer,
      camera: camera,
      storage: storageR,
      aqustion:reducer2,
      states:statesReducer,
      pulse:_pulseReducer
    }),
    EffectsModule.forRoot([effects]),
  ],
  providers: [
    DataService,
    { provide: LocationStrategy, useClass: HashLocationStrategy },
  ],
  bootstrap: [AppComponent],
})
export class AppModule {}
```

cameraview.component

```
.camerabox {
  width: 50vw;
  border: 3px solid green;
}
```

```
<mat-grid-list cols="2" rowHeight="2:1">

  <mat-grid-tile>


    <button mat-raised-button (click)="startFeed()" >Start/stop kamera strøm</button>

  </mat-grid-tile>
<mat-grid-tile>


  <img *ngIf ="!isRunning" [src]="videoUrl3">

</mat-grid-tile>
<mat-grid-tile>

    <img *ngIf ="!isRunning" [src]="videoUrl1" />
    <!-- <img src="{{ image }}" alt="" /> -->
</mat-grid-tile>
<mat-grid-tile>
    <img *ngIf ="!isRunning" [src]="videoUrl2">
</mat-grid-tile>
</mat-grid-list>
```

```
import { Component, OnInit } from '@angular/core';
import { Observable, Subject } from 'rxjs';
import { acquisitionState } from 'src/app/store/reducers/connected.reducer';
import {environment} from '../../../environments/environment.prod'
import { Store } from '@ngrx/store';
import { DataService } from 'src/app/services/data.service';
import { takeUntil } from 'rxjs/operators';
```

```
import { HttpService } from 'src/app/services/http.service';
@Component({
  selector: 'app-camera-view',
  templateUrl: './camera-view.component.html',
  styleUrls: ['./camera-view.component.css']
})
export class CameraViewComponent implements OnInit {
  destroyed$ = new Subject();
  videoUrl1 = environment.httpEndpoint+"/video_feed1"
  videoUrl2 = environment.httpEndpoint+"/video_feed2"
  videoUrl3 = environment.httpEndpoint+"/video_feed3"
  feed = false
  isRunning =false
  acquisitionStatus$!:Observable<acquisitionState>
  constructor(
    public websocket: DataService,
    private store: Store<{ aqustion: acquisitionState }>,
    private http: HttpService
  ) {
    this.acquisitionStatus$ = store.select((state) => state.aqustion);
   }

   ngOnDestroy() {
    this.websocket.send({event:'live',data:false})
    this.http.stopFeed().subscribe((data)=>{



    })
```

```
    this.destroyed$.next()
    this.destroyed$.complete()
   }
   ngAfterViewInit() {


    this.websocket.connect();

  }
  ngOnInit(): void {




    this.acquisitionStatus$.pipe(takeUntil(this.destroyed$)).subscribe((status)=>{

      if(status.status=="stopped"){

        this.isRunning = false


      }
      else if(status.status ="running"){

        this.isRunning = true
      }


    })
```

```
  }

  startFeed(){

    this.feed =! this.feed
    this.websocket.send({event:'live',data:this.feed})
  }

}
```

change-mode.component

```
<div>
  <mat-slide-toggle (change)="setTheme($event.checked)"></mat-slide-toggle>

  <mat-icon *ngif="icon == 'wb_sunny'" ><svg xmlns="http://www.w3.org/2000/svg" height="
  <mat-icon *ngif="icon == 'brightness_3'" ><svg xmlns="http://www.w3.org/2000/svg" heig
</div>

import { Component, OnInit } from '@angular/core';
import { ColormodeService } from '../../services/colormode.service';

@Component({
  selector: 'app-change-mode',
  templateUrl: './change-mode.component.html',
  styleUrls: ['./change-mode.component.css'],
})
export class ChangeModeComponent implements OnInit {
  public themes = [
```

```
    {
      name: 'dark',
      icon: 'brightness_3',
    },
    {
      name: 'light',
      icon: 'wb_sunny',
    },
  ];
  public icon = 'wb_sunny';
  public mode = 'light';

  constructor(public colorMode: ColormodeService) {}

  ngOnInit(): void {}

  setTheme(theme: any) {
    this.mode = 'light';
    this.icon = 'wb_sunny';

    if (theme) {
      this.mode = 'dark';
      this.icon = 'brightness_3';
    }
    this.colorMode.update(this.mode);
  }
}
```

chart-canvas.component

```
<div class="chart-container" style=" ">
    <canvas #canvas></canvas>
</div>

import { AfterViewInit, Component, Input, OnInit, ViewChild ,SimpleChanges} from '@angul


import { ChartType,Chart } from 'chart.js';


@Component({
  selector: 'app-chart-canvas',
  templateUrl: './chart-canvas.component.html',
  styleUrls: ['./chart-canvas.component.css']
})
export class ChartCanvasComponent implements OnInit, AfterViewInit {
  @Input() data:any;
  @ViewChild('canvas') canvas:any
  doughnutChart: any;
  constructor() {


   }


  ngOnInit(): void {
  }



  ngOnChanges(changes:SimpleChanges) {

      console.log(changes.data.currentValue)
```

```
  }

  ngAfterViewInit() {

    console.log('render new data', this.data, this.canvas);
    this.createChart()

    this.chartAdd()




  }


  chartAdd(){

    let newdata = [this.data['used'],this.data['free']]
    console.log("new data")
    console.log(newdata)
    let name = this.data['name']

    this.doughnutChart.options.title.text = name



    this.doughnutChart.data.datasets.forEach((dataset:any)=>{

      dataset.data = newdata
```

```
    })




    this.doughnutChart.update()




}


createChart(){
    this.doughnutChart = new Chart(this.canvas.nativeElement, {
      type: 'doughnut',
      data: {
          labels: ["Used", "Free"],
          datasets: [{
              label: 'Space',
              data: [1,1,1],
              backgroundColor: [
                  'rgb(255,94,87)',
                  'rgb(11,232,129)',


              ]
          }]
```

```
      },
      options: {
        title: {
            display: true,
            text: 'hei'
        }
    }
  });
}


}
```

gpsdialog.component

```
<mat-card>
    <mat-card-title>GPS Data</mat-card-title>

    <mat-card-content>


        <mat-list>
            <mat-list-item>
             Fart:   {{data.velocity}}
            </mat-list-item>
            <mat-divider></mat-divider>
            <mat-list-item>
              Kvalitet:  {{data.quality}}
            </mat-list-item>
            <mat-divider></mat-divider>
```

```
            <mat-list-item>
              satelites:  {{data.satelites}}
            </mat-list-item>
            <mat-divider></mat-divider>
            <mat-list-item>
              Lat:  {{data.lat}}
            </mat-list-item>
            <mat-divider></mat-divider>
            <mat-list-item>
              Lon:  {{data.lon}}
            </mat-list-item>

          </mat-list>


    </mat-card-content>



</mat-card>
```

```
import { Component, OnInit } from '@angular/core';
import { Subject, Subscription } from 'rxjs';
import { takeUntil } from 'rxjs/operators';
import {HttpService} from '../../services/http.service'

@Component({
  selector: 'app-gpsdialog',
  templateUrl: './gpsdialog.component.html',
  styleUrls: ['./gpsdialog.component.css']
})
```

```
export class GpsdialogComponent implements OnInit {
  data ={velocity:0,
  lon:0,
lat:0,
quality:0,
satelites:0
}
destroyed$ = new Subject();
  gpsObservable!: Subscription;
  constructor(
    private http: HttpService,
  ) { }

  ngOnInit(): void {

  this.gpsObservable = this.http.gpsData.pipe(takeUntil(this.destroyed$)).subscribe((dat


    this.data = data


   })
  }

  ngOnDestroy():void{

    this.gpsObservable.unsubscribe()
```

```
  }

}
```

imagedialog.component

```
<mat-card>
    <mat-card-title>Image Data</mat-card-title>

    <mat-card-content>


        <mat-list>
            <mat-list-item>
             camera1:   {{data.camera1}}
            </mat-list-item>
            <mat-divider></mat-divider>
            <mat-list-item>
              camera2:  {{data.camera2}}
            </mat-list-item>
            <mat-divider></mat-divider>
            <mat-list-item>
              camera3:  {{data.camera3}}
            </mat-list-item>


          </mat-list>


    </mat-card-content>
```

```
</mat-card>

import { Component, OnInit } from '@angular/core';
import { interval, Subject, Subscription } from 'rxjs';
import { switchMap, takeUntil } from 'rxjs/operators';
import {HttpService} from '../../services/http.service'
@Component({
  selector: 'app-imagedialog',
  templateUrl: './imagedialog.component.html',
  styleUrls: ['./imagedialog.component.css']
})
export class ImagedialogComponent implements OnInit {
  data ={camera1:0,
    camera2:0,
  camera3:0,

  }
  imageObserable!: Subscription
  destroyed$= new Subject();

  constructor(
    private http: HttpService,
  ) { }


  ngOnInit(): void {
```

```
    this.imageObserable =   interval(2000)
    .pipe(switchMap(() => this.http.getImageData().pipe(takeUntil(this.destroyed$))))
    .subscribe((result) => {


      this.data = result

    })
   }

   ngOnDestroy():void{

    this.imageObserable.unsubscribe()



   }
  }
```

log-dialog.component

```
<p>log-dialog works!</p>
```

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-log-dialog',
  templateUrl: './log-dialog.component.html',
  styleUrls: ['./log-dialog.component.css']
})
```

```
export class LogDialogComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

map-page.component

```
p {
  font-family: Lato;
}

.fs-container {
  width: 100vw;
  height: 75vh;
  position: absolute;
  top: 0px;
  left: 0px;
}

.leaflet-control-container .leaflet-routing-container-hide {
  display: none;
}

.form{

  z-index: 9999999999;
  margin: auto;
```

```
  text-align: center;
  position: fixed;
  left:75%;
  font-weight: 700;



}
.mat-form-field{

  font-weight: 900;
}

<div class="fs-container">

  <form class="form">

    <mat-form-field class="example-full-width">
      <mat-label>Søk veireferanse</mat-label>
      <input (keydown.enter)="onEnter($event)" type="search"  matInput placeholder="62.5
    </mat-form-field>
  </form>

  <app-map (map$)="receiveMap($event)" (zoom$)="receiveZoom($event)" (coord$)="recieveC
</div>

import { Component, OnInit } from '@angular/core';
import { LatLngExpression, Map,marker,polyline,point,circleMarker, popup, featureGroup,L
import { HttpService } from 'src/app/services/http.service';
import {FormBuilder, FormGroup, Validators} from '@angular/forms';
```

```
@Component({
  selector: 'app-map-page',
  templateUrl: './map-page.component.html',
  styleUrls: ['./map-page.component.css'],
})
export class MapPageComponent implements OnInit {
  private map!: Map;
  private zoom!: number;
  private latlng!:any
  coord =""

  constructor(
    private http: HttpService,
    private _formBuilder: FormBuilder,
  ) {}

  ngOnInit(): void {}

  receiveMap(map: Map) {
    this.map = map;









  }
```

```
  getRoadReference(search:any){

   this.http.getRoadReference(search).subscribe((data)=>{


     let section = data[0].vegsystemreferanse.kortform

    let newmarker = circleMarker([search.lat,search.lng]).bindPopup(section).addTo(this


    this.map.fitBounds([[search.lat,search.lng]]);

     newmarker.openPopup()
      newmarker.addEventListener("click",function(event){

       newmarker.remove()

      })


   })


}

getPostition(ref:any){


   this.http.getpositonRoadRef(ref).subscribe((data)=>{
```

```
      console.log(data)

      // let point = data.geometri.wkt









    })

}

onEnter(e:any){

  let data:string = e.target.value

  if (data.match(/^\d+\.\d+/)) {

    let latlon=  data.split(",")

  let lat = parseFloat(latlon[0].trim())
  let lon =parseFloat(latlon[1].trim())

   let search = {lat:lat,lng:lon}

    this.getRoadReference(search)
```

```
  }
  else{

   this.getPostition(data)



  }









}
getColor(){

  let colors = ["red","blue","yellow","green","black","purple","brown"]

  return colors
}

addToMap(lists:Array<any>,names:Array<any>){

  for(let i=0 ; i< lists.length;i++){
    let colors = this.getColor()
    var color:string = colors[Math.floor(Math.random() * colors.length)];
```

```
      let latlngs = lists[i]
      let line = polyline(latlngs, {color: color}).addTo(this.map);
      line.bindTooltip(names[i]).openTooltip()
      this.map.fitBounds(line.getBounds());

    }


  }

 ngAfterViewInit() {

  this.getGpsLog()

}


  getGpsLog(){

    this.http.Getcoordinates().subscribe(data=>{

      this.addToMap(data.lists,data.names)
    })
```

```
}


convertcsv(){

}

receiveZoom(zoom: number) {
  this.zoom = zoom;
}

recieveCoord(latlong:any){



  this.http.getRoadReference(latlong).subscribe((data)=>{


    let section = data[0].vegsystemreferanse.kortform

   let newmarker = circleMarker([latlong.lat,latlong.lng],{radius:2}).bindPopup(sectio

    newmarker.openPopup()
     newmarker.addEventListener("click",function(event){


      newmarker.remove()


     })
```

```
    })



  }
}
```

map.component

```
.map-container {
  height: 100vw;
  width: 100vw;
  position: inherit;
}
```

```
<div
  class="map-container"
  leaflet
  [leafletOptions]="options"
  (leafletMapReady)="onMapReady($event)"
  (leafletMapZoomEnd)="onMapZoomEnd($event)"
  (leafletClick) ="onClick($event)"
></div>
```

```
import {
  Component,
  EventEmitter,
  Input,
  OnDestroy,
  OnInit,
  Output,
} from '@angular/core';
```

```
import {
  Map,
  Control,
  DomUtil,
  ZoomAnimEvent,
  Layer,
  MapOptions,
  tileLayer,
  latLng,
  popup



} from 'leaflet';
import "leaflet-routing-machine";
declare let L:any
@Component({
  selector: 'app-map',
  templateUrl: './map.component.html',
  styleUrls: ['./map.component.css'],
})
export class MapComponent implements OnInit {
  @Output() map$: EventEmitter<Map> = new EventEmitter();
  @Output() zoom$: EventEmitter<number> = new EventEmitter();
  @Output() coord$: EventEmitter<any> = new EventEmitter();
  options = {
    layers: [
      tileLayer('http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
        opacity: 0.7,
        maxZoom: 19,
        detectRetina: true,
```

```
      attribution:
        '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> co
    }),
  ],
  zoom: 14,
  center: latLng(62.563074682835925, 7.682671781391905),
};

public map!: Map;
public zoom!: number;
constructor() {}

ngOnInit(): void {




}

onMapReady(map: Map) {
  console.log('map ready');
  this.map = map;
  this.map$.emit(map);
  this.map.locate({ setView: true, maxZoom: 16 });
  this.zoom = map.getZoom();
  this.zoom$.emit(this.zoom);



}
```

```
  onClick(e:any){

    let latlng = e.latlng

    this.coord$.emit(latlng)


  }


  onMapZoomEnd(e: any) {
    console.log(e.target.getZoom());
    this.zoom = e.target.getZoom();
    this.zoom$.emit(this.zoom);
  }
}
```

sidebar.component

```
body {
  overflow-x: hidden;
}
#sidebar-wrapper {
  min-height: 100vh;
  margin-left: -15rem;
  -webkit-transition: margin 0.25s ease-out;
  -moz-transition: margin 0.25s ease-out;
  -o-transition: margin 0.25s ease-out;
  transition: margin 0.25s ease-out;
}
```

```
#sidebar-wrapper .sidebar-heading {
  padding: 0.875rem 1.25rem;
  font-size: 1.2rem;
}
#sidebar-wrapper .list-group {
  width: 15rem;
}
#page-content-wrapper {
  min-width: 100vw;
}
#wrapper.toggled #sidebar-wrapper {
  margin-left: 0;
}
@media (min-width: 768px) {
  #sidebar-wrapper {
    margin-left: 0;
  }
  #page-content-wrapper {
    min-width: 0;
    width: 100%;
  }
  #wrapper.toggled #sidebar-wrapper {
    margin-left: -15rem;
  }
}
```

```
<router-outlet></router-outlet>
```

```
import { Component, OnInit } from '@angular/core';
import { FormControl } from '@angular/forms';
import * as $ from 'jquery';
```

```
@Component({
  selector: 'app-sidebar',
  templateUrl: './sidebar.component.html',
  styleUrls: ['./sidebar.component.css'],
})
export class SidebarComponent implements OnInit {
  mode = new FormControl('over');
  constructor() {}

  ngOnInit(): void {
    //Toggle Click Function
    $('#menu-toggle').click(function (e) {
      e.preventDefault();
      $('#wrapper').toggleClass('toggled');
    });
  }
}
```

stepper.component

```
.topbox {
  height: 25vh;
  margin-top: 5rem;
  border-radius: 25px;
}
h1 {
  margin: 0;
  position: absolute;
  top: 50%;
  left: 50%;
  -ms-transform: translate(-50%, -50%);
```

```
  transform: translate(-50%, -50%);

  font-family: "Open Sans", sans-serif;
  font-size: 8vh;

  letter-spacing: -1px;
  line-height: 1;
  text-align: center;
}
.camerabox {
  width: 20vw;



  z-index: 9999;
  position: absolute;
  padding: 10px;
  border: 1px solid #77aaff;
  box-shadow:  -1px 1px #77aaff,
       -2px 2px #77aaff,
       -3px 3px #77aaff,
       -4px 4px #77aaff,
       -5px 5px #77aaff;
}


.check {
  color: greenyellow;
}

.check {
```

```
  color: red;
}
.steppercontainer {
  height: 25vh;
  margin-top: 5rem;
  border-radius: 25px;
}

<div class="steppercontainer"  >
  <mat-horizontal-stepper [linear]="isLinear" #stepper   >
    <mat-step [completed]="step1()">
      <ng-template matStepLabel>Koble til kamera</ng-template>


      <div>
        <button
          mat-raised-button
          color="primary"
          [disabled]="step1()"
          (click)="sendEvent('init', 'empty')"
        >
          Start
        </button>


        <button
        mat-button
        (click)="goBack()"
      >
        Tilbake
      </button>
      </div>
```

```
</mat-step>
<!-- <mat-step [completed]="step2()">
  <ng-template matStepLabel>Valider bilde</ng-template>
  <button
    [disabled]="step2()"
    mat-button

    (click)="sendEvent('validation', 'A')"
  >
    Camera A
  </button>
  <button
    [disabled]="step2()"
    mat-button

    (click)="sendEvent('validation', 'B')"
  >
    Camera B
  </button>

  <button
  [disabled]="step2()"
  mat-button

  (click)="sendEvent('validation', 'C')"
>
```

```
    Camera C
  </button>

    <div>
      <button mat-raised-button   (click)="finalStep()" >Next</button>
      <button
      mat-button
      (click)="goBack()"
    >
      Tilbake
    </button>
    </div>

    <div>
      <div class="camerabox">
        <img [src]=image alt="image" />
      </div>
    </div>
  </mat-step> -->

  <mat-step >
    <form [formGroup]="firstFormGroup">
    <ng-template matStepLabel>Trip navn</ng-template>
    <mat-form-field>
      <mat-label>Name</mat-label>
      <input matInput placeholder="Trip navn" formControlName="firstCtrl" [(ngModel)]=
```

```
      </mat-form-field>

    <div>
    <button
    mat-button
    [disabled]="!checkInput()"
    (click)="start(tripName)"

    mat-raised-button
  >
    Skriv inn navn og start
  </button>
</div>
</form>
    </mat-step>



  </mat-horizontal-stepper>
</div>

import {
  Component,
  EventEmitter,
  Input,
  OnDestroy,
  OnInit,
  Output,
  SimpleChange,
  SimpleChanges,
  ViewChild,
} from '@angular/core';
```

```
import { DataService } from '../../services/data.service';
import { Store } from '@ngrx/store';
import { Observable, Subject } from 'rxjs';
import {
  camera,
  State,
  Cameras,
} from 'src/app/store/reducers/connected.reducer';
import {

  reset,

} from '../../store/actions/connected.action';
import { takeUntil } from 'rxjs/operators';
import { websocket } from 'src/app/store/actions/connected.action';
import { HttpService } from 'src/app/services/http.service';
import { MatSnackBar, MatSnackBarHorizontalPosition, MatSnackBarVerticalPosition } from
import { MatStepper } from '@angular/material/stepper';
import {FormBuilder, FormGroup, Validators} from '@angular/forms';
import {DomSanitizer} from '@angular/platform-browser';
import { animate, style, transition, trigger } from '@angular/animations';
@Component({
  selector: 'app-stepper',
  templateUrl: './stepper.component.html',
  styleUrls: ['./stepper.component.css'],
  animations:[

  ]
})
export class StepperComponent implements OnDestroy, OnInit {
```

```
@Input() image:any;
@Output() stepperDone = new EventEmitter<string>();
@ViewChild('stepper') private Stepper!: MatStepper;
public isLinear = true;
firstFormGroup!: FormGroup
public status = 'start';
public statusColor = {
  'background-color': ' #0be881',
};

tripName =""

public connected$!: Observable<Cameras>;
public cameraStatus!: Cameras;
destroyed$ = new Subject();
stopStream$ = new Subject();
imageValidated = true;
horizontalPosition: MatSnackBarHorizontalPosition = 'center';
verticalPosition: MatSnackBarVerticalPosition = 'top';
constructor(
  private websocketService: DataService,
  private store: Store<{ camera: Cameras }>,
  private http: HttpService,
  private snackBar: MatSnackBar,
  private _formBuilder: FormBuilder,
  public _DomSanitizationService: DomSanitizer
) {
  this.connected$ = store.select((state) => state.camera);
}
```

```
  step1() {
    let temp = false;
    if (
      this.cameraStatus.connectionA == 'ok' ||
      this.cameraStatus.connectionB == 'ok'
    ) {
      temp = true;
    }

    return temp;
  }

  step2() {
    if(this.imageValidated){
      return true
    }else{
    return false;
    }
  }

  step3() {
    if(this.imageValidated){
      return true
    }else{
    return false;
    }
  }
  ngOnChanges(changes:SimpleChanges):void{

    console.log(changes)
```

```
}
ngOnInit(): void {
  this.firstFormGroup = this._formBuilder.group({
    firstCtrl: ['', Validators.required]
  });
  this.connected$.pipe(takeUntil(this.destroyed$)).subscribe((data) => {
    this.cameraStatus = data;



    this.snackBarHandler(this.cameraStatus);
  });
}

checkInput(){

  let isValid:boolean = this.firstFormGroup.valid

  return isValid
}

snackBarHandler(cameraStatus: Cameras) {
  if (cameraStatus.connectionA == 'ok' || cameraStatus.connectionB == 'ok') {


      this.Stepper.next();
      let snackBarRef = this.snackBar.open('Camera connected', 'Next', {
        horizontalPosition: this.horizontalPosition,
```

```
        verticalPosition: this.verticalPosition,
        duration:2000
      });



  }


  if (
    cameraStatus.connectionA == 'failed' &&
    cameraStatus.connectionB == 'failed'
  ) {
    let snackBarRef = this.snackBar.open(
      'Camera not connected',
      'Try again',
      {horizontalPosition: this.horizontalPosition,
        verticalPosition: this.verticalPosition,}
    );

    snackBarRef.onAction().subscribe(() => {
      this.sendEvent('init', 'empty');
    });
  }
}
ngOnDestroy(): void {
  this.destroyed$.next();
}
finalStep=() :void=>{
  this.imageValidated = true
```

```
    console.log("final")
    this.Stepper.next()


  }

  /**
   * Sends http request to endpoint and the server starts the streming-loops
   */
  start(tripName:string) {
    this.Stepper.reset();
    this.stepperDone.emit(tripName);



    this.cameraStatus = {
      connectionA: '',
      connectionB: '',
      detected: 0,
      streamB: '',
    };
    this.store.dispatch(reset());
  }


   goBack(){
    this.Stepper.reset()
    this.stepperDone.emit("reset")
    this.store.dispatch(reset());
  }
```

```
  sendEvent(event: string, data: any) {
    this.websocketService.send({ event: event, data: data });
  }
}
```

config-page.component

```
.togglebuttons{
    margin: 5vh;
    display: grid;
}

<mat-grid-list cols="2" rowHeight="2:1">
    <mat-grid-tile>
        <div class="togglebuttons">




      <mat-list>
        <mat-list-item>
          <mat-slide-toggle
          (change) ="toggleGps(gpsControl)"
          [(ngModel)]="gpsControl"

          >Toggle GPS fps</mat-slide-toggle>

        </mat-list-item>
        <mat-list-item>
```

```
        <button style="margin:5vh" mat-raised-button (click)="initGPS()" > Init GPS </
      </mat-list-item>

    </mat-list>



</div>
</mat-grid-tile>


<mat-grid-tile>
  <div class="FPS">
  <div>
  <form>
    <mat-form-field class="example-full-width">
      <mat-label>Endre Fps og trykk Enter</mat-label>
      <input (keydown.enter)="onEnter($event)" type="number"  matInput placeholder="
    </mat-form-field>
  </form>
</div>
  <div>
  <form [formGroup]="firstFormGroup">
    <ng-template matStepLabel>COM port</ng-template>
    <mat-form-field>
      <mat-label>COM port</mat-label>
      <input matInput placeholder="Ex. ttyUSB1" formControlName="firstCtrl" [(ngMode
    </mat-form-field>

  <div>
```

```
        <button
        mat-button
        [disabled]="!checkInput()"
        (click)="connect(comPort)"

        mat-raised-button
      >
        Koble fra
      </button>
      <button
      mat-button
      [disabled]="!checkInput()"
      (click)="disConnect(comPort)"

      mat-raised-button
    >
      Koble til
    </button>

  </div>
  </form>
</div>
</div>




    </mat-grid-tile>
    <mat-grid-tile>

      <mat-slide-toggle
```

```
      (change) ="debug(debugMode)"
      [(ngModel)]="debugMode"

      >Toggle Debug</mat-slide-toggle>

    </mat-grid-tile>
    <mat-grid-tile>
      <mat-slide-toggle
      (change) ="guru(guruMode)"
      [(ngModel)]="guruMode"

      >Toggle GURU MODE</mat-slide-toggle>

    </mat-grid-tile>
  </mat-grid-list>
```

```
import { Component, OnInit } from '@angular/core';
import { data } from 'jquery';
import { interval, Observable, Subject, Subscription } from 'rxjs';
import { switchMap, takeUntil, tap } from 'rxjs/operators';
import { DataService } from '../services/data.service';
import { HttpService } from '../services/http.service';
import { Store } from '@ngrx/store';
import {environment} from '../../environments/environment.prod'
import {FormBuilder, FormGroup, Validators} from '@angular/forms';
import {
  acquisitionState,_pulseReducer

} from 'src/app/store/reducers/connected.reducer';
```

```
import {pulse} from '../store/actions/connected.action'
import { States,Pulse} from '../store/models/connected.model'

@Component({
  selector: 'app-config-page',
  templateUrl: './config-page.component.html',
  styleUrls: ['./config-page.component.css']
})
export class ConfigPageComponent implements OnInit {
  gpsControl = false
  image:any
  guruMode =false
  firstFormGroup!: FormGroup
  debugMode = false
  fps = 5
  destroyed$ = new Subject();
  fpshttp!: Subscription;
  states!: Subscription;
  comPort =""

  constructor(
    public websocket: DataService,
    private http: HttpService,
    private store: Store<{ states:States,pulse:Pulse }>,
    private _formBuilder: FormBuilder,
  ) {


   }
```

```
  onEnter(event:any){

     let fps = event.target.value

    let payload ={fps:fps}

     this.http.setFPS(payload).subscribe((data)=>{

       this.fps = data.fps
     })

  }

  checkInput(){

   let isValid:boolean = this.firstFormGroup.valid

   return isValid
}

ngOnInit(): void {

   this.firstFormGroup = this._formBuilder.group({
     firstCtrl: ['', Validators.required]
   });
this.states =  this.http.getStates().subscribe((data)=>{

     this.guruMode = data.guruMode
     this.debugMode = data.debug
     this.gpsControl = data.gpsControl
```

```
  })



 this.fpshttp = this.http.getFPS().subscribe((data:any)=>{

    this.fps = data.fps

  })

}

ngOnDestroy() {
  this.fpshttp.unsubscribe()
  this.states.unsubscribe()

}

connect(port:any){

  port ={ port:port,connection:true}


  this.http.connectFps(port).subscribe((data)=>{
```

```
      console.log(data)
      this.store.dispatch(pulse({isPulse:data}))
    })

  }


  disConnect(port:any){

    port ={ port:port,connection:false}


    this.http.connectFps(port).subscribe((data)=>{

      console.log(data)
      this.store.dispatch(pulse({isPulse:data}))
    })

  }


initGPS(){

  this.websocket.send({ event: 'initGPS', data: "" })

}
 toggleGps(checked:boolean){
```

```
    this.websocket.send({ event: 'toggleGps', data: this.gpsControl })


}

guru(checked:boolean){

 this.websocket.send({ event: 'guru', data: this.guruMode })


}

debug(checked:boolean){


 this.websocket.send({ event: 'debug', data: this.debugMode })


}

ngAfterViewInit() {
```

```
}

}
```

dash.component

```
.topbox {
  height: 25vh;
  margin-top: 5vh;
  /* border-radius: 25px; */
}

.card {
  margin-top: 8vw;

  /* border-radius: 25px; */

  font-family: "Open Sans", sans-serif;
}
.button {
  cursor: pointer;
}
@media only screen and (max-width: 768px) {

  .card-text{
    flex: auto;
    font-size: xx-small;
```

```
    font-family: "Open Sans", sans-serif;

  }

}
.card-text{
  flex: auto;


  font-family: "Open Sans", sans-serif;

}

.status{

  font-family: "Open Sans", sans-serif;

}
.pulsing {


  animation: pulse-green 2s infinite;
}

@keyframes pulse-green {
  0% {
    transform: scale(1);

  }
```

```
  50% {
    transform: scale(1.1);

  }




  100% {
    transform: scale(1);

  }



}
.storagetext{

  margin-bottom: auto;
}
.container {
  text-align: center;
  min-width: none;
}

.mat-icon {
  font-size: 50px;
}
```

```
.stop svg{
  fill:#ff5e57
}

.playPause svg{
  fill:#0be881
}
h1 {
  margin: 0;
  position: absolute;
  top: 50%;
  left: 50%;
  -ms-transform: translate(-50%, -50%);
  transform: translate(-50%, -50%);

  font-family: "Open Sans", sans-serif;
  font-size: 8vh;

  letter-spacing: -1px;
  line-height: 1;
  text-align: center;
}

.info {

}
```

```
<div class="container">
  <div class="row">
```

```
<div class="col">
  <div
    matRipple
    matTooltip="Helhetlig status om systemet"
    *ngIf="!startStepper"

    class="card"
    class="topbox"
    [ngStyle]="mainStatus()"




  >
    <div class="card-body" >

      <div class="card-title" >
        <h1 >{{ captureStatus}}</h1> </div>



  </div>









  </div>

  <app-stepper
    (stepperDone)="onStepperDone($event)"
    *ngIf="startStepper"
    [image]="image"
```

```
      ></app-stepper>


    </div>
  </div>
  <div class="row" style="height: 15vh;" *ngIf ="_isRunning() || _isConfigured()" [@inOu
    <div  class="col">
      <div>
      <button matTooltip="Starter bildetaking"  (click)="startPulse()" mat-button [ngCla
    </div>
      <ng-template #pause>
    <button [ngClass]="startClass" matTooltip="Pauser bildetaking" mat-button  class="pl
      <svg matRipple xmlns="http://www.w3.org/2000/svg" enable-background="new 0 0 24 24
    </button>

  </ng-template>

    </div>


    <div class="col">

      <button [disabled]="!_isRunning()" matTooltip=" Stopper biletaking" (click) ="stop


        <svg xmlns="http://www.w3.org/2000/svg" height="15vh" viewBox="0 0 24 24" width=

      </button>
```

```
    </div>

  <div class="col">
    <button [disabled]="!_isRunning()" matTooltip="Nødstopp stopper momentant" (click) =

      <svg xmlns="http://www.w3.org/2000/svg" enable-background="new 0 0 24 24" height="

    </button>
  </div>
  </div>

  <div class="row" >

    <div class="col">

      <div
        class="card"
        matTooltip="Trykk for mer info"
        matRipple
        [ngStyle]="gpsStatus"
        (click)="openGpsDialog()"
      >
        <div class="card-body">
          <mat-icon><svg xmlns="http://www.w3.org/2000/svg" height="24" viewBox="0 0 24
          <h2 class="card-title">GPS</h2>
          <p class="card-text">
            {{ gpsQuality }}
          </p>
        </div>
      </div>
```

```
    </div>
    <div class="col">
      <div class="card"  [matTooltip]="getTooltip()" matRipple (click)="toggle()"  [ngS

        <div class="card-body" >
          <mat-icon><svg xmlns="http://www.w3.org/2000/svg" height="24" viewBox="0 0 24
          <h2 class="card-title">Bilde</h2>
          <p class="card-text">
            {{ imageQuality }}
          </p>
        </div>
      </div>
    </div>
    <div class="col">

      <div class="card"  matTooltip="Trykk for mer info" matRipple [ngStyle]="storage" (

        <div class="card-body">
          <mat-icon><svg xmlns="http://www.w3.org/2000/svg" height="24" viewBox="0 0 24
          <h2 class="card-title">Lagring</h2>


          <div class="card-text">
          <p class="storagetext" >{{ this.timeLeftStorage}}</p>
          </div>

        </div>
      </div>
    </div>
  </div>
```

```
</div>

import { ChangeDetectorRef, Component, Input, OnInit, ViewChild } from '@angular/core';
import { Observable, Subject, Subscription } from 'rxjs';

//import AppState from '../store/state/app.state';
import { interval } from 'rxjs';
import { Store } from '@ngrx/store';
import { acquisitionState, State } from '../store/reducers/connected.reducer';
import { Storage,States } from '../store/models/connected.model';
import {StoragedialogComponent} from '../components/storagedialog/storagedialog.componen
import {GpsdialogComponent} from '../components/gpsdialog/gpsdialog.component'
import {ImagedialogComponent} from '../components/imagedialog/imagedialog.component'
import {StepperComponent} from '../components/stepper/stepper.component'
import {
  websocket,
  initA,
  initB,
  streamB,
  detectedCameras,
  storage,
  reset,
  acquisition,
  states,
  running,
  captruring,
  configured,
  pulse
} from '../store/actions/connected.action';

import { DataService } from '../services/data.service';
```

```
import { GpsService } from '../services/gps.service';
import { HttpService } from '../services/http.service';
import { catchError, map, switchMap, takeUntil, tap } from 'rxjs/operators';
import Gps from '../models/gps';
import {
  trigger,
  state,
  style,
  animate,
  transition,
} from '@angular/animations';
import {MatDialog, MatDialogRef, MAT_DIALOG_DATA} from '@angular/material/dialog';
import { MatSnackBar,MatSnackBarHorizontalPosition } from '@angular/material/snack-bar';
import { fadeinout, routeTransitionAnimations } from '../route-transition-animations';
import { RouterOutlet } from '@angular/router';
import { MatHorizontalStepper } from '@angular/material/stepper';

@Component({
  selector: 'app-dash',
  templateUrl: './dash.component.html',
  styleUrls: ['./dash.component.css'],
  animations: [
    fadeinout,
    routeTransitionAnimations


  ]
})
export class DashComponent implements OnInit {
  @ViewChild(StepperComponent)
  stepper!: StepperComponent;
```

```
private destroySubject$: Subject<void> = new Subject();
public connected$!: Observable<string>;
public status: String ="s";
public gpsQuality = 'Ingen Kontakt';
public startStepper = false;
public imageQuality = 'Konfigurer ny ';
public storageStatus = 'Dårlig';
public centered =true
public initDone =false
captureStatus ="Klar"

public startClass ="playPause"
public random = Math.floor(Math.random() * 10);
image:any;
public statusColor = {
  'background-color': ' #0be881',
};
myColor = '#0be881';
public gpsStatus = {
  'background-color': '#C0C0C0',
};

public imageStatus = {
  'background-color': '#C0C0C0',
};

public storage = {
  'background-color': '#ff5e57',
};
isRunning = false;
```

```
  isConfigured = false
  destroyed$ = new Subject();
  messages: any;
  timeLeftStorage: string ="ingen data";
  totalStorage!: { total: number; used: number; free: number };
  liveData$!: Observable<any>;
  configLoaded =false;
  gpsobservable!: Subscription
  storageObservable!: Subscription;
  websocketObservable: Subscription;
  gpsControl: any;
  isCapturing= false;
  public states$!: Observable<States>;
  guruMode: any;
  constructor(
    private store: Store<{ statusState: State,acquisition:acquisitionState,states:States
    public dialog: MatDialog,
    public websocketService: DataService,
    public gps: GpsService,
    private http: HttpService,
    private snackBar: MatSnackBar,

    private cdRef: ChangeDetectorRef
  ) {
    this.liveData$ = this.websocketService.messages$.pipe(
      map((rows) => rows),
      catchError((error) => {
        throw error;
      }),
      tap({
```

```
        error: (error) => console.log('[Live component] Error:', error),
        complete: () => console.log('[Live component] Connection Closed'),
      })
    );


  this.websocketObservable = this.liveData$.pipe(takeUntil(this.destroyed$)).subscribe(
     this.messageHandler(data);
   });

   this.starting()



   this.states$ = store.select((state)=>state.states)









}
   /**
  * Prepares route
  * @param outlet will emit an activate event any time a new component is being instant
  *               and a deactivate event when it is being destroyed
  * @returns true if route, false otherwise
  */
    prepareRoute(outlet: RouterOutlet): boolean {
```

```
    return (
      outlet &&
      outlet.activatedRouteData &&
      outlet.activatedRouteData["animationState"]
    );
  }

  _imageStatus(){

    return this.imageStatus
  }

  _isRunning(){

    return this.isRunning
  }

  _isConfigured(){

    return this.isConfigured
  }

mainStatus(){
  let color = {
    'background-color': '#ff5e57',
  };
  if(this.storageStatus=="Ok" && (this.gpsQuality =="Dårlig"||this.gpsQuality=="Ok")){

    color = {
      'background-color': '#FFDC00',
```

```
      };

      this.status ="Ok"
    }

   else if(this.storageStatus=="Perfekt" && this.gpsQuality =="Perfekt" && this.isConfig

      color = {
        'background-color': '#0e881',
      };

      this.status ="Perfekt"
    }

    else{

      color = {
        'background-color': '#C0C0C0',
      };

      this.status ="Kan ikke starte"

    }


    return color
  }

  ngOnDestroy() {
```

```
    this.gpsobservable.unsubscribe()
    this.storageObservable.unsubscribe()
    //this.websocketObservable.unsubscribe()




  }

  testSearch(){

    this.websocketService.send({event:"search",data:""})
  }

  connect() {
    this.websocketService.send({ message: 'message' });
  }



  openStorageDialog() {
    const dialogRef = this.dialog.open(StoragedialogComponent);

    dialogRef.afterClosed().subscribe(result => {
      console.log(`Dialog result: ${result}`);
    });
  }
```

```
openGpsDialog() {
  const dialogRef = this.dialog.open(GpsdialogComponent);

  dialogRef.afterClosed().subscribe(result => {
    console.log(`Dialog result: ${result}`);
  });
}

openImageDialog() {
  const dialogRef = this.dialog.open(ImagedialogComponent);

  dialogRef.afterClosed().subscribe(result => {
    console.log(`Dialog result: ${result}`);
  });
}

connectToServer() {}

test(){
  this.isRunning =! this.isRunning
}

startStep() {
  this.startStepper = true;
}

resetStates():void{
  this.startClass ="playPause"
  this.isConfigured = false
  this.isCapturing = false
```

```
    this.store.dispatch(running({running:false}))
    this.store.dispatch(configured({configured:false}))
    this.store.dispatch(captruring({capturing:false}))
    this.startStepper = false;
    this.imageQuality = 'Trykk for å konfigurere ny ';
     this.imageStatus = {
      'background-color': '#C0C0C0',
    };

    this.captureStatus ="Stanset"


  }
  emergencyStop(){
    this.captureStatus ="Stopper:Lagrer bilder"

    //this.resetStates()
    this.store.dispatch(acquisition({newStatus:"Stanset"}))
    this.websocketService.send({ event: 'emergency', data: 'stopping stream' });

  }
  stop() {

    this.captureStatus ="Stopper:Lagrer bilder"


    this.store.dispatch(acquisition({newStatus:"Stopper:Lagrer bilder"}))

    this.websocketService.send({ event: 'stop', data: 'stopping stream' });
    // this.http.stopstream$.next()
```

```
    // this.http.stopstream$.complete()
  }

  stopped(images:any){

    this.resetStates()
let  data = `Camera1: ${images.camera1}  Camera2: ${images.camera2}  Camera3: ${images.c

    let snackBarRef = this.snackBar.open(data, 'OK', {

    });

  }

  pauseImage(){
    this.isCapturing = false
    this.store.dispatch(captruring({capturing:false}))
    this.captureStatus ="Satt på pause"

    this.store.dispatch(acquisition({newStatus:"paused"}))

    this.websocketService.send({ event: 'pause', data: 'pause stream' });

  }

  reset(){

    this.startStepper = false
  }
```

```
toggle() {
  if (!this.isConfigured ) {

    this.captureStatus ="Klar"

    this.startStep();
  }
  else if (this.isConfigured){

    this.openImageDialog()
  }



}

getTooltip(){

 let status ="Konfigurer ny"

  if (this.isConfigured) {

    status = "Trykk for info "

  }



  return status
}

starting() {
```

```
  this.gpsobservable =   interval(2000)
    .pipe(switchMap(() => this.http.getGpsData().pipe(takeUntil(this.destroyed$))))
    .subscribe((result) => {

      this.gpsDataHandler(result);
    })

 this.storageObservable =   interval(5000)
    .pipe(switchMap(() => this.http.getStorageData().pipe(takeUntil(this.destroyed$)))
    .subscribe((result:any) => {
      this.storageDataHandler(result);
    });
}

ngOnInit = () : void => {
  console.log("[dash] init")




  // this.http.getStates().subscribe((states:any)=>{

  //   this.isRunning = states.running
  //     this.configLoaded = states.init_ok
```

```
    //      this.isConfigured = states.isConfigured
    //      this.isCapturing = states.capturing
    //      this.guruMode = states.guruMode




    // })









    //get gps data on intervall
  }

  ngAfterViewInit() {
    console.log("[dash] after view init")

    this.websocketService.connect();
    //this.websocketService.send({event:"init",data:""})
    this.websocketService.send({event:"states",data:""})


    this.states$.pipe(takeUntil(this.destroyed$)).subscribe((data)=>{
```

```
this.isCapturing= data.capturing
this.gpsControl= data.gpsControl
this.isConfigured= data.isConfigured
this.isRunning = data.running
this.configLoaded = data.init
this.guruMode = data.gurumode



if (this.isConfigured){

 this.imageQuality = "OK"
 this.imageStatus['background-color']='#0be881'

}else{
 this.imageQuality = 'Konfigurer ny ';
 this.imageStatus['background-color']='#C0C0C0'

}

if(!this.isRunning){

 this.captureStatus = "Stanset"

}

if(this.isCapturing){


 this.captureStatus="Tar bilder"
```

```
      this.startClass ="pulsing"
     }
     else{


       this.startClass="playPause"
     }



   })


}
storageDataHandler = (result:any) :void => {

   console.log(result)



   this.store.dispatch(storage({ storage: result }));

   this.setStorageStatus(result.total.timeleft);
   this.timeLeftStorage = this.getTimeString(result);

   this.totalStorage = result.total.free;



}
```

```
setStorageStatus =(timeleft: any) :void => {
  let h: number = timeleft.h;
  let m: number = timeleft.m;

  if (h >= 1 && m>20) {
    this.storageStatus = 'Perfekt';
    this.storage['background-color'] = '#0be881';
  }
  else if (h <= 1 && m >20) {
    this.storageStatus = 'Ok';
    this.storage['background-color'] = '#FFDC00';
  }

  else if (h == 0 && m<20 ) {
    this.storageStatus = 'Dårlig';
    this.storage['background-color'] = '#ff5e57';
  }
}
getTimeString(result:any) {
  let timeleft = result.total.timeleft;
  let timestring = `Hour:${timeleft.h}  Minutes:${timeleft.m}`;
  return timestring;
}
gpsDataHandler(result: any) {

  console.log(result)

  if(result.pulse){

    this.store.dispatch(pulse({ isPulse:true}))
```

```
    }

    this.http.gpsData.next(result)
    let status: number = result.quality;
    let velocity = result.velocity;

    switch (status) {
      case 0:
        this.gpsQuality = 'Ingen Kontakt';
        this.gpsStatus['background-color'] = '#C0C0C0';
        break;

      case 1:
        this.gpsQuality = 'Dårlig';
        this.gpsStatus['background-color'] = '#ff5e57';
        break;

      case 2:
        this.gpsQuality = 'Ok';
        this.gpsStatus['background-color'] = '#FFDC00';
        break;

      case 3:
        this.gpsQuality = 'Perfekt';
        this.gpsStatus['background-color'] = '#0be881';
        break;
    }
  }
```

```
messageHandler = (message: any) :void => {
  console.log(message);
  switch (message.event) {
    case 'connected':
      this.store.dispatch(websocket({ newStatus: 'Connected' }));

      this.websocketService.send({ event: 'init', data: '' });

      break;
    case 'disconnected':
      this.store.dispatch(websocket({ newStatus: 'Disconnected' }));
      break;

    case 'starting':
      this.status = 'Trykk for å starte';


      break;
    case 'stopping':
      this.status = 'Start';

      this.store.dispatch(reset());

      break;

    case 'stopped':


    this.stopped(message.data)
```

```
  break;

case 'search':


this.websocketService.send({ event: 'init', data: '' });
console.log(message.data)

break

case 'initA':
  this.store.dispatch(initA({ connected: message.data }));

  break;
case 'initB':
  this.store.dispatch(initB({ connected: message.data }));

  break;

case 'loadConfig':

 let ref = this.snackBar.open(`[KONFIG] ${message.data}`, 'OK', {});

 ref.onAction().subscribe(()=>{

  if (message.data =="config_ok"){
  this.configLoaded = true
  }else{
    this.configLoaded =false
```

```
    }
   })



  break;



  case 'snapshot':

    this.image =`${ message.data }`
    this.stepper.image = `${ message.data }`
    break;

  case 'states':



  let update ={running: message.data.running,
    init: message.data.init_ok,
    isConfigured: message.data.isConfigured,
    gpsControl:message.data.gpsControl,
   capturing:message.data.capturing,
   gurumode:message.data.guruMode}

  this.store.dispatch(states({states:update}))
  this.store.dispatch(running({running:message.data.running}))
    this.setStates(message.data)
```

```
      break;

      case "detected":

        let detected = message.data

        console.log(detected)

        this.store.dispatch(detectedCameras({detected:detected}))
        break



        case 'folderCreated':
          console.log("folder created")
          break

      default:
        console.log(message.data);
        break;
    }
  }

  setStates=(data:any):void=>{

        this.isRunning = data.running
        this.configLoaded = data.init_ok
        this.isConfigured = data.isConfigured
        this.gpsControl = data.gpsControl
```

```
        this.isCapturing = data.capturing
        this.guruMode = data.guruMode



}

startStream(tripName:any){

 let payload = {name:tripName}

  this.http
  .start1(payload)
  .pipe(takeUntil(this.http.stopstream$))
  .subscribe((results)=>{
    let payload = {stream:"stream1",results:results}
    this.captureStatus ="Klar"
    this.store.dispatch(running({running:false}))
    this.websocketService.send({event:"reset",data:""})
    this.websocketService.send({event:"log",data:payload})
    this.merge()
  })
this.http
  .start2(payload)
  .pipe(takeUntil(this.http.stopstream$))
  .subscribe((results)=>{

    let payload = {stream:"stream2",results:results}

    this.websocketService.send({event:"log",data:payload})
```

```
  })


    this.http
    .start3(payload)
    .pipe(takeUntil(this.http.stopstream$))
    .subscribe((results)=>{
      console.log(results)
      let payload = {stream:"stream3",results:results}

      this.websocketService.send({event:"log",data:payload})
    })
}


merge() {
  this.http.merge().subscribe((d:any)=>{


    console.log(d)
  })
}

startPulse(){
  this.startClass ="pulsing"
  this.isRunning = true

  this.store.dispatch(running({running:true}))
```

```
    this.isCapturing = true
    this.store.dispatch(captruring({capturing:true}))
    this.captureStatus ="Tar bilder"

    this.store.dispatch(acquisition({newStatus:"running"}))
    this.websocketService.send({ event: 'pulse', data: 'start' });
  }
  onStepperDone(event: any) {


    if(event =="reset"){

      this.startStepper = !this.startStepper;
      this.isRunning = false

    }else{
    this.captureStatus="pulsing"
    this.startStream(event)
    this.websocketService.send({ event: 'start', data:event });
    this.startStepper = !this.startStepper;
    this.store.dispatch(configured({configured:true}))
    this.imageQuality = "OK"
    this.imageStatus['background-color']='#0be881'
    this.store.dispatch(running({running:true}))

    }
  }

  public toCameraView() {
```

```
    console.log('STOPPING');
    this.websocketService.send({ event: 'stop', data: ' ' });
  }
}

import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { MapPageComponent } from '../components/map-page/map-page.component';
import {ConfigPageComponent} from '../config-page/config-page.component'
import { DashComponent } from './dash.component';
import {CameraViewComponent} from '../components/camera-view/camera-view.component'

const routes: Routes = [{ path: '', component: DashComponent,
data: { animationState: "One" }
 }, { path: 'map', component: MapPageComponent,
 data: { animationState: "Two" }},
{ path: 'config', component: ConfigPageComponent,
data: { animationState: "Three" } },{ path: 'camera', component: CameraViewComponent,
data: { animationState: "Four" }} ];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class DashRoutingModule { }

import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';

import { DashRoutingModule } from './dash-routing.module';
import { DashComponent } from './dash.component';
```

```
import { MatIconModule } from '@angular/material/icon';
import { MatStepperModule } from '@angular/material/stepper';
import { StepperComponent } from '../components/stepper/stepper.component';
import { MatTabsModule } from '@angular/material/tabs';
import { MatButtonModule } from '@angular/material/button';
import { MatRippleModule } from '@angular/material/core';
import { MatSnackBarModule } from '@angular/material/snack-bar';
import {MatFormFieldModule} from '@angular/material/form-field'
import {MatInputModule} from '@angular/material/input';
import {MatExpansionModule} from '@angular/material/expansion';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import {MatDialogModule} from '@angular/material/dialog';
import { ChartsModule } from 'ng2-charts';
import {StoragedialogComponent} from '../components/storagedialog/storagedialog.componen
import {GpsdialogComponent} from '../components/gpsdialog/gpsdialog.component'
import {ImagedialogComponent} from '../components/imagedialog/imagedialog.component'
import { ChartCanvasComponent } from '../components/chart-canvas/chart-canvas.component'
import {MatCardModule} from '@angular/material/card';
import {MatDividerModule} from '@angular/material/divider';
import {MatListModule} from '@angular/material/list';
import { MapComponent } from '../components/map/map.component';
import { MapPageComponent } from '../components/map-page/map-page.component';
import { ConfigPageComponent } from '../config-page/config-page.component';
import {MatGridListModule} from '@angular/material/grid-list';
import { LeafletModule } from '@asymmetrik/ngx-leaflet';
import { MatSlideToggleModule } from '@angular/material/slide-toggle';
import {MatProgressBarModule} from '@angular/material/progress-bar'
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import {MatTooltipModule} from '@angular/material/tooltip';
@NgModule({
```

```
  declarations: [DashComponent, StepperComponent,StoragedialogComponent,ChartCanvasCompo
    MapPageComponent,
    ConfigPageComponent],
  imports: [
    CommonModule,
    DashRoutingModule,

    MatIconModule,
    MatStepperModule,
    MatTabsModule,
    MatButtonModule,
    MatRippleModule,
    MatSnackBarModule,
    MatTooltipModule,
    MatInputModule,
    MatSlideToggleModule,
    FormsModule,
    MatListModule,
    MatProgressBarModule,
    ChartsModule,
    MatDividerModule,
    MatExpansionModule,
    MatDialogModule,
    MatCardModule,
    ReactiveFormsModule,
    MatGridListModule,
    LeafletModule,],
})
export class DashModule {}
```

colormode.service

```
import { Injectable, Renderer2, RendererFactory2 } from '@angular/core';

@Injectable({
  providedIn: 'root',
})
export class ColormodeService {
  private renderer: Renderer2;

  private colorScheme: any;
  // Define prefix for more clear and readable styling classes in scss files

  // Define prefix for more clear and readable styling classes in scss files
  private colorSchemePrefix = 'color-scheme-';

  constructor(rendererFactory: RendererFactory2) {
    // Create new renderer from renderFactory, to make it possible to use renderer2 in a
    this.renderer = rendererFactory.createRenderer(null, null);
  }

  _detectPrefersColorScheme() {
    // Detect if prefers-color-scheme is supported
    if (window.matchMedia('(prefers-color-scheme)').media !== 'not all') {
      // Set colorScheme to Dark if prefers-color-scheme is dark. Otherwise set to light
      this.colorScheme = window.matchMedia('(prefers-color-scheme: dark)')
        .matches
        ? 'dark'
        : 'light';
    } else {
      // If browser dont support prefers-color-scheme, set it as default to dark
```

```
      this.colorScheme = 'dark';
    }
  }

  _setColorScheme(scheme: any) {
    this.colorScheme = scheme;
    // Save prefers-color-scheme to localStorage
    localStorage.setItem('prefers-color', scheme);
  }

  _getColorScheme() {
    // Check if any prefers-color-scheme is stored in localStorage
    if (localStorage.getItem('prefers-color')) {
      // Save prefers-color-scheme from localStorage
      this.colorScheme = localStorage.getItem('prefers-color');
    } else {
      // If no prefers-color-scheme is stored in localStorage, Try to detect OS default
      this._detectPrefersColorScheme();
    }
  }

  load() {
    this._getColorScheme();
    this.renderer.addClass(
      document.body,
      this.colorSchemePrefix + this.colorScheme
    );
  }

  update(scheme: any) {
```

```
    this._setColorScheme(scheme);
    // Remove the old color-scheme class
    this.renderer.removeClass(
      document.body,
      this.colorSchemePrefix + (this.colorScheme === 'dark' ? 'light' : 'dark')
    );
    // Add the new / current color-scheme class
    this.renderer.addClass(document.body, this.colorSchemePrefix + scheme);
  }

  currentActive() {
    return this.colorScheme;
  }
}
```

http.service

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Observable, Subject } from 'rxjs';
import { environment } from '../../environments/environment';
import { catchError, tap } from 'rxjs/operators';

@Injectable({
  providedIn: 'root',
})
export class HttpService {
  private _baseUrl: string;
  private _http: HttpClient;
  public stopstream$ = new Subject();
  public stopGps$ = new Subject();
```

```
public gpsData =  new Subject<any>();
constructor(private http: HttpClient) {
  this._http = http;
  this._baseUrl = environment.httpEndpoint;
}

getGpsData(): Observable<any> {
  return this._http.get(this._baseUrl + '/gps/').pipe(tap(
   error=>console.log(error)
  ))
}



getImageData(): Observable<any> {
  return this._http.get(this._baseUrl + '/capturing/').pipe(tap(
   error=>console.log(error)
  ))
}

getStorageData(): Observable<any> {
  return this._http.get(this._baseUrl + '/storage/')
}

start1(trip:any): Observable<any> {
  return this._http.post(this._baseUrl + '/start1/',trip).pipe(tap(

  ))
}
start2(trip:any): Observable<any> {
```

```
    return this._http.post(this._baseUrl + '/start2/',trip).pipe()
  }
  start3(trip:any): Observable<any> {
    return this._http.post(this._baseUrl + '/start3/',trip).pipe(
    )
  }

  connectFps(port:any): Observable<any> {
    return this._http.post(this._baseUrl + '/connectFPS/',port).pipe(
    )
  }

  startGpsLoop(): Observable<any> {
    return this._http.get(this._baseUrl + '/gpsLoop/');
  }

  stop(): Observable<any> {
    return this._http.get(this._baseUrl + '/stop');
  }



  getLiveStream(): Observable<any> {
    return this._http.get(this._baseUrl + '/video_feed1/')
  }

  getFPS(): Observable<any> {
    return this._http.get(this._baseUrl + '/RaspFPS/')
  }

  Getcoordinates(): Observable<any> {
```

```
    return this._http.get(this._baseUrl + '/GetCoordinates/')
  }

  getStates(): Observable<any> {
    return this._http.get(this._baseUrl + '/getStates/')
  }


  setFPS(fps:any): Observable<any> {
    return this._http.post<any>(this._baseUrl + '/changeFps/',fps).pipe(tap(error=>conso
  }

  stopFeed(): Observable<any> {
    return this._http.get(this._baseUrl + '/stopFeed/')
  }
  merge(): Observable<any> {
    return this._http.get(this._baseUrl + '/merge/')
  }

  getRoadReference(latlong:any):Observable<any>{

    const headerDict = {

      'Accept': 'application/vnd.vegvesen.nvdb-v3+json',
    }

    const requestOptions = {
      headers: new HttpHeaders(headerDict),
    };
```

```
    let lat = latlong.lat
    let lon = latlong.lng


    return this._http.get(`https://nvdbapiles-v3.atlas.vegvesen.no/posisjon?lat=${lat}&l
  }
  getpositonRoadRef(ref:any):Observable<any>{

    const headerDict = {

      'Accept': 'application/vnd.vegvesen.nvdb-v3+json',
    }

    const requestOptions = {
      headers: new HttpHeaders(headerDict),
    };




    return this._http.get(`https://nvdbapiles-v3.atlas.vegvesen.no/veg?vegsystemreferans
  }

}
```

web socket.service

```
import { Injectable, Inject } from '@angular/core';
import { of, Observable, Subject, timer } from 'rxjs';
import { connectableObservableDescriptor } from 'rxjs/internal/observable/ConnectableObs
import {
```

```
  catchError,
  delay,
  delayWhen,
  filter,
  map,
  retryWhen,
  switchAll,
  switchMap,
  tap,
} from 'rxjs/operators';
import { webSocket, WebSocketSubject } from 'rxjs/webSocket';
import { environment } from '../../environments/environment';
import { Store } from '@ngrx/store';
import { State } from '../store/reducers/connected.reducer';
import { websocket } from '../store/actions/connected.action';
import { EMPTY } from 'rxjs';
export const WS_ENDPOINT = environment.wsEndpoint;

@Injectable({
  providedIn: 'root',
})


//https://javascript-conference.com/blog/real-time-in-angular-a-journey-into-websocket-a

export class DataService {

  private connection$: any;
  RETRY_SECONDS = 10;
  messagesSubject$: any = new Subject();
```

```
public messages$ = this.messagesSubject$.pipe(
  switchAll(),
  catchError((e) => {
    throw e;
  })
);

constructor(private store: Store<any>) {}

ngOnDestroy() {
  this.closeConnection();
}


public connect(cfg: { reconnect: boolean } = { reconnect: false }): void {

  if(cfg.reconnect){
    this.store.dispatch(websocket({ newStatus: 'Reconnecting' }));
  }
 if (!this.connection$ || this.connection$.closed ||this.connection$ == undefined) {


  this.connection$ = this.getNewWebSocket();

    const messages = this.connection$.pipe(
      cfg.reconnect ? this.reconnect  : (o: any) => o,
      tap({

        error: (error) => console.log(error),
      }),
```

```
      catchError((_) => EMPTY)
    )
    this.messagesSubject$.next(messages);
  }
}

/**
 * Return a custom WebSocket subject which reconnects after failure
 */
private getNewWebSocket() {
  console.log(WS_ENDPOINT + '/stream/' + Date.now());
  return webSocket({
    url: WS_ENDPOINT + '/stream/' + Date.now(),
    openObserver: {
      next: () => {
        console.log('[DataService]: connection ok');
      },
    },
    closeObserver: {
      next: () => {
        this.store.dispatch(websocket({ newStatus: 'Disconnected' }));
        console.log('[DataService]: connection closed');
        this.connection$ = undefined;

        this.connect({ reconnect: true });
      },
    },
  });
}
```

```
private reconnect(observable: Observable<any>): Observable<any> {

  return observable.pipe(
    retryWhen((errors) =>
      errors.pipe(
        tap((val) => {




          console.log('[DataService]: reconnecting', val);
        }),
        delayWhen((_) => timer(2000))
      )
    )
  );

}

public  retry() {
 this.store.dispatch(websocket({ newStatus: 'Reconnecting' }));
}
closeConnection() {
  if (this.connection$) {
    this.connection$.complete();
    this.connection$ = undefined
  }
  else{
    this.connect()
  }
}
```

```
  send(data: any) {
    if (this.connection$) {
      this.connection$.next(data);
    } else {
      console.error('Did not send data, open a connection first');
    }
  }

  close() {
    this.connection$.complete();
    this.connection$ = undefined;
  }
}
```

NGRX store

```
import { Action } from '@ngrx/store';

import { Storage,States } from '../models/connected.model';
import { createAction, props } from '@ngrx/store';
import { createDirective } from '@angular/compiler/src/core';

export const connect = createAction('[Socket] connect');

export const websocket = createAction(
  '[WebSocket] connect',
  props<{ newStatus: string }>()
);

export const initA = createAction(
```

```
  '[CameraA] init',
  props<{ connected: string }>()
);
export const initB = createAction(
  '[CameraB] init',
  props<{ connected: string }>()
);

export const detectedCameras = createAction(
  '[CameraB] stream',
  props<{ detected: number }>()
);
export const streamB = createAction(
  '[CameraB] stream',
  props<{ stream: string }>()
);
export const running = createAction(
  '[States] running',
  props<{  running: boolean }>()
);
export const configured = createAction(
  '[States] configuring',
  props<{  configured: boolean }>()
);
export const captruring = createAction(
  '[States] capturing',
  props<{  capturing: boolean }>()
);
```

```
export const reset = createAction('[Camera] reset');

export const storage = createAction(
  '[Storage] add',
  props<{ storage: Storage }>()
);

export const states = createAction(
  '[States] update ',
  props<{states:States}>()
);


export const acquisition = createAction(
  '[Acquisition] connect',
  props<{ newStatus: string }>()
);


export const pulse = createAction(
  '[Pulse] connect',
  props<{ isPulse: boolean }>()
);

export interface Connected {
  connectionStatus: string;
  timeStamp: string | any;
  tryConnecting: boolean;
}

export interface Storage {
```

```
  storage: {  };

}

export interface States{
  running:boolean;
  capturing:boolean;
  init :boolean;
  isConfigured:boolean;
  gpsControl:boolean
  gurumode :boolean
}


export interface Pulse{

  pulse:boolean
}

import { Connected } from '../models/connected.model';
import { Storage,States,Pulse } from '../models/connected.model';
import {
  connect,
  websocket,
  initA,
  initB,
  detectedCameras,
  streamB,
  storage,
  reset,
  acquisition,
```

```
  states,
  running,
  configured,
  captruring,
  pulse
} from '../actions/connected.action';

import { Action, createReducer, on } from '@ngrx/store';
import { state } from '@angular/animations';

export interface State {
  status: string;
}
export interface acquisitionState  {
  status: string;
}
export const initialStorage: Storage = {
  storage: {  }

};



export const initialPulse: Pulse = {
  pulse: false

};

export const initialStates: States = {

  isConfigured:false,
```

```
  capturing:false,
  gpsControl:true,
  gurumode:false,
  running:false,
  init:false


};

export interface Cameras {
  connectionA: string;
  detected: number;
  connectionB: string;
  streamB: string;
}

export const initialCamera: Cameras = {
  connectionA: 'disconnected',
  detected: 0,
  connectionB: 'disconnected',
  streamB: 'stopped',
};

export const initialState: State = { status: 'disconnected' };
export const initialStateAcquisition: acquisitionState = { status: 'stopped' };

const websocketReducer = createReducer(
  initialState,
  on(websocket, (state, { newStatus }) => ({ status: newStatus }))
);
```

```
const pulseReducer = createReducer(
  initialPulse,
  on(pulse, (state, { isPulse  }) => ({...state, pulse: isPulse }))
);
const StatesReducer = createReducer(
  initialStates,
  on(states, (state,{states} ) => ({...state,

    capturing: states.capturing,
    gpsControl:states.gpsControl,
    gurumode:states.gurumode,
    init:states.init,
    isConfigured:states.isConfigured,
    running:states.running




  })),
  on(running,(state,{running}) =>({...state,running:running})),
  on(configured,(state,{configured})=>({...state,isConfigured:configured})),
  on(captruring,(state,{capturing})=>({...state,capturing:capturing}))
);

const cameraAReducer = createReducer(
  initialCamera,
  on(initA, (state, { connected }) => ({ ...state, connectionA: connected })),
  on(initB, (state, { connected }) => ({ ...state, connectionB: connected })),
```

```
  on(detectedCameras, (state, { detected }) => ({ ...state, detected: detected })),
  on(streamB, (state, { stream }) => ({ ...state, streamB: stream })),
  on(reset, (state) => ({
    ...state,
    connectionA: '',
    connectionB: '',
    streamA: '',
    streamB: '',
  }))
);

const storageReducer = createReducer(
  initialStorage,

  on(storage, (state, { storage }) => ({
   storage:storage
  }))
);
const acquisitionReducer = createReducer(
  initialStateAcquisition,
  on(acquisition, (state, { newStatus }) => ({ status: newStatus }))
);

export function reducer2(state: acquisitionState | undefined, action: Action) {
  return acquisitionReducer(state, action);
}

export function _pulseReducer(state: Pulse | undefined, action: Action) {
  return pulseReducer(state, action);
}
```

```
export function reducer(state: State | undefined, action: Action) {
  return websocketReducer(state, action);
}
export function camera(state: Cameras | undefined, action: Action) {
  return cameraAReducer(state, action);
}

export function storageR(state: Storage | undefined, action: Action) {
  return storageReducer(state, action);
}

export function statesReducer(state: States | undefined, action: Action) {
  return StatesReducer(state, action);
}
```

C Raspberry pi

pulsegenerator

```
import RPi.GPIO as GPIO
import time
import requests

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
myPWM = GPIO.PWM(18,50)
myPWM.start(10)
fps = 5
i = 0
Restget ="http://10.0.222.1:8000/RaspFPS"
```

```
while True:

    time.sleep(2)

    try:

        response = requests.get(Restget)
        data = response.json()

        fps_new = data['fps']
        start = data['start']
        fps = fps_new

        if int(fps)==0:
            myPWM.ChangeDutyCycle(0)
        else:
            myPWM.ChangeDutyCycle(50)
            myPWM.ChangeFrequency(int(fps))









    except Exception as e:

        myPWM.ChangeDutyCycle(0)
```

```
GPIO.cleanup()
```

D Arduino

E pulsegenerator

```
int FPS = 10;
//https://www.arduino.cc/en/Tutorial/SecretsOfArduinoPWM
void setup() {
  pinMode(9, OUTPUT);                                  // Set digital pin 9 (D9) to an outp
  TCCR1A = _BV(COM1A1) | _BV(WGM11);                   // Enable the PWM output OC1A on dig
  TCCR1B = _BV(WGM13) | _BV(WGM12) | _BV(CS12);        // Set fast PWM and prescaler of 256
  ICR1 = 12499;                                        // Set the PWM frequency to 5Hz: 16M
  OCR1A = 1249;                                        // Set the duty-cycle to 10%: 62499

  Serial.begin(9600);                                  //listen to serial port
}



void loop() {
if(Serial.available() > 0)  {
    int incomingData= Serial.read();
    Serial.print(incomingData);
    int frequency = incomingData;
    int FPS =  16000000/(256*frequency)-1; // calculate new FPS
    ICR1 = FPS;
    OCR1A = FPS/10;

}
```

```
}
```