




6. OKTOBER 2020

AOBE – AUTOMATISK OMRÅDEBESKYTTENDE ENHED
PROCESRAPPORT

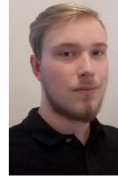
BENJAMIN ØSTNER, NICKY HANSEN, NICKLAS PEDERSEN, BJØRN DELEURAN
ZBC RINGSTED
Ahorn allé 3-5



Titelblad

Projekttitel

- Automatisk Områdebeskyttende Enhed



Benjamin Østner

Benjamin Østner

Dato

- 14.09.2020 - 6.10.2020



Bjørn Deleuran

Bjørn Deleuran

Skole

- ZBC Ringsted



Nicklas Pedersen

Nicklas P.

Vejleder

- Camilla



Nicky Hansen

Nicky Hansen

Indholdsfortegnelse

TITELBLAD	1
INDHOLDSFORTEGNELSE	2
LÆSEVEJLEDNING	3
LÆSEFLOW	3
ANDET	3
INDLEDNING	4
PROBLEMFORMULERING	4
AFGRÆNSNING	4
PROJEKTPLANLÆGNING	5
ANALYSE	5
PROBLEMET	5
IDÉER TIL LØSNINGER	5
DEN VALGTE LØSNING	6
HJEMMESIDE – SPA ANGULAR	6
OBJECT DETECTION SOFTWARE	7
STREAMING SERVER – RTMP MODUL PÅ NGINX	7
NODE.JS – API	8
KONKLUSION	9
TIDSPLAN	9
REALISERET TIDSPLAN	11
ARBEJDSFORDELING	11
LOGBOG	12
BILAG	18

Læsevejledning

Læseflow

Denne rapport vil beskrive problemet ved vores opgave, hvad for en løsning vi vil bruge til at løse problemet, hvordan løsningen fungerer individuelt og samlet, samt tidsplan og log over processen.

Gennemgang af rapporten skal ske i denne ordre, for at opnå det bedste læseflow.

1. Intro.

- Indledning: introduktion til projekt.
- Problemformulering: problemet som projektet vil arbejde med.
- Afgrænsning: Problemer, holdninger og andet som der ikke vil blive taget stilling til.
- Projektplanlægning: Kort beskrivelse af løsning og indhold af løsning.

2. Analyse.

- Problemet: Introduktion til problemet som projektet arbejder med.
- Idéer til løsninger: forskellige overvejelser til eventuelle løsninger.
- Den valgte løsning: løsningen som vi mente der var bedst egnet.
- Hjemmeside – SPA Angular: Beskrivelse af hjemmeside og valg af teknologi.
- Object Detection Software: Begrundelse for valg af teknologi og hvordan det virker.
- Streaming Server – RTMP Module på NGINX: Begrundelse for valg af teknologi og hvordan det virker.
- Node.js – API: Begrundelse for valg af teknologi til API.
- Konklusion:

3. Dokumentation.

- Tidsplan: Den forventede tidsplan hvis alt gik efter planen.
- Realiseret tidsplan: den aktuelle tidsplan.
- Arbejdsfordeling: hvordan arbejdet er fordelt.
- Logbog: kort log over gruppemedlemmernes arbejde.

4. Bilag.

- Bilag: links og andet beskrivelse af anvendt materiale.

Andet

Tekniske termer som ikke er blevet forklaret, kan søges på nettet for betydning.

Links i teksten bliver vist som en reference til det passende link ned i bilaget.

Eksempel: (Link 7) = 7. EksempelLink

Author: Benjamin Østner

Indledning

Vi har udviklet et system som gør det muligt at genkende krybskytter i naturreservater og sende video feed til en hjemmeside. Vagter skal være i stand til at udrede incidents der bliver oprettet ud fra object detection.

For at kunne målrette brugeroplevelse bedst muligt har vi udarbejdet hjemmesiden med brugervenlighed i tankerne. Det viste sig, at det vil være muligt for en vagt bruge hjemmesiden uden ekstern vejledning, dog skulle vagten have hjælp til opsætning af kamera.

Designmetoden er UML/deployment diagram. Genkendelsessoftwaren bliver udviklet i Python med "tensorflow" som back end, og afvikles på operativsystemet Linux Ubuntu 20.04, hjemmesiden er skrevet i angular8, og Arduino/ESP32 koden er skrevet i C, der bliver desuden brugt tilpassede C/C++ biblioteker til de specifikke udstyr, f.eks. et stepmotor bibliotek til styring af stepmotoren. Det udviklede computerprogram fungerer efter hensigten.

Author: Nicklas Pedersen

Problemformulering

Hvordan kan man sikre beskyttelsen af truede dyrearter gennem automatiseret overvågning af områder og genkendelse software, og hvordan kan man gøre dette på en effektiv manér uden at skulle tage højde for de begrænsninger og komprimerende forhold der kan være ved at bruge manuel arbejdskraft, samt hvilke metoder og teknologier kan vi føre i brug for at beskytte dyr og overvåge store områder ligegyldigt vejr- og lysforhold.

Afgrænsning

Projektet har fokus på hvordan automatisering kan lette byrden hos de ansatte ude på reservaterne, og vil derfor ikke gå i dybden med den menneskelige del af projektet.

Beskyttelsen af dyr vil ske igennem område overvågning og holde krybskytterne væk fra dyrene, mens at dyrene skal forstyrres mindst muligt.

Moralen af vores løsning vil ikke der ikke blive skrevet om, da der er skrevet en del hvordan krybskytteri allerede bliver bekæmpet med private lejesoldater, og derfor er vores løsning ikke er et nyt syn på dette problem.

Der vil ikke blive lagt fokus på hvordan vejrforhold og lysforhold kan gøre overvågningen sværere, da løsninger til disse problemer ligger i ekstra hardware, som der ikke var tilgængelig under udvikling.

Dette projekt er stadig kun et koncept som ikke er klar til at blive opsat i den virkelige verden.

Author: Benjamin Østner

Projektplanlægning

Projektet indeholder en hjemmeside som skal vise et live feed fra et kamera som sidder på en Arduino, dette skal gå igennem en server som har en database og hoster hjemmesiden.

Vi brugte noget tid med at teste forskellige ideer og teknologier og kom frem til et bibliotek i Python som indeholdt objekt detektion software, som man kunne koble op til et kamera. Problemet med det var derefter at få vist et live feed fra et kamera på hjemmesiden, men der kom vi frem til at sætte en streaming server op på serveren. Den streaming server endte med at få et live feed fra et kamera som der blev kørt objekt detektion på, hvor resultatet blev sendt over på sin egen stream, som hjemmesiden hentede og viste.

Vi kom frem til at der også skulle oprettes nogle incidents når object detection fandt f.eks en person, hvor der så ville blive taget et billede og et timestamp, som bliver sendt til en database, hvor hjemmesiden så kunne vise disse incidents i en liste.

Vi skulle også have en arduino enhed/tårn med et kamera på, som kunne bevæge sig og holde øje med et område. Vi kom frem til at lave en platform med batteri og kamera, som bliver sat fast på en stepmotor som vil dreje platformen rundt, så kameraet kan se området rundt om hvor den står.

Author: Benjamin Østner

Analyse

Problemet

Der er troede dyrearter rundt omkring på jorden som skal holdes øje med og beskyttes, så racen ikke uddør og jorden mister endnu en dyreart. Dette kan dog være problematisk fordi at der kan være mangel på ansatte som passer på dem, områderne er for store og sidst men ikke mindst, krybskytteri.

Mange dyrearter er truede i dag, fordi at jagt og krybskytteri er meget lukrativt og er svært at gøre noget ved, fordi der netop er mangel på arbejdere som holder øje med områderne som dyrene lever i.

Et eksempel på dette kan findes i denne artikel, hvor der står følgende *"Nogle steder skal en enkelt ranger måske patruljere 100 kvadratkilometer, så han har reelt ikke nogen mulighed for at gøre noget ved problemet."* (link 1). For at putte dette i et perspektiv som alle kan forstå: En fodboldbane er 7140 kvadratmeter, der kan være 14.000 fodboldbaner på 100 kvadratkilometer, dette er ikke muligt at holde øje med, hvis man kun er én person.

Men hvad kan man gøre for at have et bedre overblik over reservaterne og overvåge de store områder, samtidig med at man kan undgå at skulle ansætte flere rangere.

Idéer til løsninger

Problemet kan kort opsummeres til at der er for lidt personer til at holde øje med et alt for stort område, så vi kom frem til at bruge kameraer, men selv dét har sine problemer. Hvis man prøvede at sætte kameraer op, så ville man hurtigt finde ud af at det ville kræve alt for mange kameraer, og at det ville kræve alt for mange ledninger og andet hardware.

En anden ide er at der kunne sættes et dronenetværk op, som fløj efter forskellige gps-pointers, så forskellige ruter kan sættes op. Et problem ved det er at droner larmer, så krybskytteren ville kunne høre dronerne langt før de dukkede op, da der ikke er nogen lyd til at overdøve dronerne ude på f.eks. savannen. Det ville resultere i et netværk af droner der holdte øje med et større område, men krybskytterne ville have en chance for at kunne gemme sig, inden dronerne ville kunne se dem.

En tredje løsning ville være at sætte kameraer direkte på dyrene, men denne ide ville være dårlig på flere punkter. En af grundene til at det er en dårlig løsning, er at de fleste dyr skal fanges og have sat et kamera på, så det ville forstyrre dem at sætte det op. Det er også bare generelt en dårlig ide at sætte ting fast på dyr, fordi at det nemt ville kunne gå i stykker hvis de ramte f.eks. et træ, en sten osv. Det er heller ikke sikkert at det ville hjælpe, da dyrene som regel ikke ser krybskytten før det er for sent, og selv hvis de gjorde, så er det for sent at hjælpe dem, da krybskytten allerede har gjort klar til at skyde dem.

Den valgte løsning

Efter vi havde gjort os nogle tanker om hvad man kan og ikke kan, så kom vi frem til at man kunne sætte nogle få kameraer op på roterende platforme som sidder på et tårn, et stykke oppe i luften, så et enkelt kamera kan holde øje med et lidt større område.

Disse tårne kan man sætte op i en form for grid system, så man kan udnytte deres længere syn og minimere det antal tårne der skal sættes op.

Disse tårne ville sende et live feed til en server over Wi-Fi eller anden trådløs forbindelse til en server, hvor der ville køre Object Detection software på de live feeds serveren modtager.

Det Object Detection software vil så rebroadcast live feedet med de fundne objekter hvilket så er klar til at blive hentet og vist på en hjemmeside, hvor en enkel person vil kunne holde øje med 10-20-50 tårne som strækker sig over store områder, uden at skulle rykke sig.

Når en person, bil eller andet som ikke hører, til ude i reservatet bliver fundet af kameraerne, så vil der blive oprettet et incident som vil blive vist på hjemmesiden, som indeholder et billede af det der blev fundet, et tidspunkt og hvilket kamera der fandt det.

Dette vil gøre overvågningen af et større område væsentligt nemmere, gennem brugen af automatik og AI, så en enkelt person kan udføre arbejdet en del nemmere.

Et problem ved denne løsning kunne være at der skal sættes strøm op i tårnene, det kunne dog løses med en del batterier, men de skal også udskiftes.

Et andet problem ville være at selv om tårnene kan se over større områder, så vil der stadig være områder som er ude for synsvidde, da det stadig ikke kan betale sig at sætte alt for mange tårne op over for små områder.

Hjemmeside – SPA Angular

Vi har valgt at lave hjemmesiden som en SPA (Single Page Application) Angular hjemmeside, fordi at en SPA hjemmeside kun skal indlæses når man går ind på hjemmesiden. En fordel ved det er at alt brug af hjemmesiden kan gøres uden at skulle vente på at de nye side bliver indlæst.

Derimod kan det tage ekstra tid at indlæse siden første gang, da alt på hjemmesiden skal indlæses, men det gør stadig bruget af hjemmesiden til en mere flydende oplevelse.

Ikke nok med at det kan tage ekstra tid første gang, så kan det også tage langt længere tid end en ikke SPA-hjemmeside, da den udover indholdet skal også indlæse en eller anden form for JavaScript for at kunne omforme hjemmesiden til en SPA.

Komponenttankegangen er velunderstøttet af de fleste SPA frameworks, i hvert fald Angular. Komponenttankegangen lyder således at programmet er splittet op i moduler, de moduler kan så genbruges med forskelle inputs så hvis man har noget det har den præcis samme logik, men er måske anderledes i udseende eller andet, behøver man ikke at lave det samme html, css, og JavaScript en gang til.

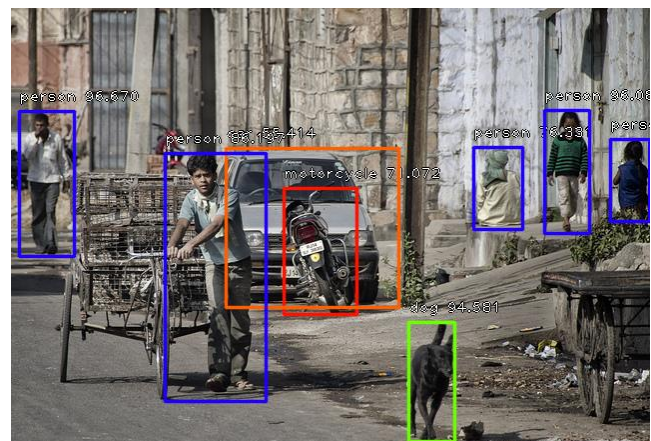
Object Detection Software

Da der kan være tidspunkter hvor der ikke holdes øje med live feedet på hjemmesiden eller det kan være svært at se forskel på de forskellige ting på live feedet, så mente vi at et Object Detection software ville være en god ting at have på live feedet.

Når Object Detection softwaret genkender en person på live feedet, så vil der blive sat en farvet boks rundt om personen, samtidig med at et incident bliver oprettet, som indeholder et timestamp og billede af det der blev genkendt.

Det kunne være at over natten var der blevet fundet 3 personer og oprettet incidents på dem, hvilket gør at man kan have konstant overvågning uden bemanning, fordi at når den næste kiggede på hjemmesiden, så kan man hurtigt se at der er blevet fundet 3 personer, da der er 3 incidents med tidspunkter på.

Vi fandt et bibliotek til Python (link 2) som der havde object detection og image processing, så vi skulle bare finde ud af at bruge det og gøre resultatet tilgængeligt for hjemmesiden at hente, dette bibliotek bruger Tensorflow til selve detektions algorithmerne. Vi mente at object detection er nødvendigt for systemet, da det gør overvågning af en til flere område gennem kameraer nemt og simpelt, da man har AI til at gøre en del af det grove arbejde som samtidig kører 24/7, hvilket gør at der er overvågning, selv om natten.

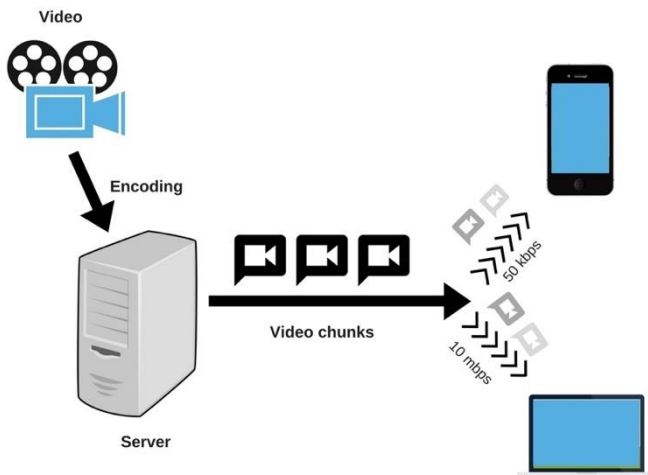


Figur 1 | Eksempel på object detection

Streaming Server – RTMP Modul på NGINX

Til at modtage video input har vi kompileret NGINX med et RTMP-modul, som gør det muligt for serveren at modtage video-input over RTMP (Real-Time-Messaging-Protocol) små video-filer kaldet fragments bliver opbevaret midlertidigt på serveren, de sidste 4-5 fragments bliver opbevaret så klienter som er faldet et stykke bagud f.eks. på grund af ringe forbindelse o. lign.

stadig kan følge med i streamen og får fornemmelsen af at de ser "live" med selvom de i realiteten kan være minutter bagud.



Figur 2 | Overblik over video til hjemmeside forbindelse

En Python applikation kører 24/7 som analyserer de opbevarede videofragmenter vha. vores AI Object Detection, resultatet bliver opbevaret på serveren med et specielt suffiks "_proc" navnet skal antyde behandlede videofragmenter, disse fragmenter kan tilgås som en stream over HTTP via en HLS (HTTP Live Streaming) player som er inkluderet på vores SPA Angular side.

Node.js – API

Til at håndtere de forskellige kald til databasen har vi valgt at udvikle en Node.js API.

Vi har valgt Node.js da den integrerer let med vores Angular front-end og da den med meget få libraries har mulighed for at skabe forbindelse til diverse former for databaser og sende data på korrekt vis så databasen forstår det.

API'en kører som en REST'ful API og kører derfor 24/7 og lytter bare på en specifik port på ip-adressen maskinen den ligger på er tildelt.

At vores API er REST'ful betyder at vi kan beholde vores front-end og vores database modulære så længe de ved i hvilket format de skal sende data, dette gør også hele vore løsning mere fleksibel.

Konklusion

Under projektet blev der lavet et automatisk tårn med kamera som kan sende et live feed til en server, som derefter kører object detection på live feedet, som så bliver sendt videre til en hjemmeside hvor det bliver vist.

Krybskytteri er et problem som kan være svært at løse af mange grunde, da der er mange underproblemer som der skal tages hensyn til, som f.eks. de store områder som dyrene befærder sig på og mangel på ressourcer.

Under dette projekt kom vi frem til en løsning som har potentialet til at formindske det antal dyr som der dør på grund af krybskytteri, samtidig med at manglen på rangere bliver reduceret og området som en enkelt ranger kan holde øje med bliver udvidet.

Produktet har nogle problemer som der skal tages højde for, da manglen på ressourcer stadig vil påvirke produktets resultater i form af lavere kvalitet på kameraet, hvilket kan give false positives i object detection når den leder efter mennesker, eller at den slet ikke opfanger mennesker på det den kan se.

Ved den rigtige implementering af produktet med et ordentligt opsat grid system, hvor afstand, position, højde og kamera kvalitet er rigtigt, så kan antallet af krybskytter som bliver opdaget og derefter samlet op af en ranger, eventuelt skudt af tårnet, forøges. Dette vil resultere i at antallet af dyr som bliver skudt falder.

Hvis ressourcerne tillader det, så er det også muligt at sætte FLIR (link 3) op på kameraet, så det har såkaldt night-vision (termisk funktionalitet) om natten, hvilket der også vil reducere det antal dyr der dør, da krybskytterne bliver opdaget og fanget, selv om natten.

Efter alt dette er gjort, er det stadig muligt at krybskytterne kan slippe afsted med at skyde nogle dyr, uden at blive opdaget, da det vil være umuligt at have 100% overvågning af områder af denne størrelse, med de ressourcer der er tilgængelige.

Tidsplan

4 uger

1. Første uge

- Mandag – Fredag
 - Få færdiggjort kravsspecifikation
 - Dele projekt ud i mindre bidder

2. Anden uge

- Mandag
 - Fastsat design på hjemmeside – Interface mockups
 - Fastsat design på database – ER diagram
 - Fundet nødvendige komponenter til arduino
- Tirsdag
 - Opsættelse af ubuntu/linux på server

- Begynd opsættelse af hjemmeside
 - Opsættelse af test database
 - Indsæt mock data
 - Design og opbyggelse v.1 af AOBE model/tårn
 - Onsdag
 - Begynd arbejde på API mellem database og hjemmeside
 - Fundet løsning til streaming af kamera fra AOBE på hjemmesiden
 - Basic layout af hjemmeside færdig
 - Side 1
 - a. Live feed/streaming boks på hjemmeside
 - b. Felt/liste til unresolved incidents
 - Side 2
 - a. Felt/liste til alle incidents
 - Overvågnings kode til horisontal stepper opsat
 - Torsdag
 - Opsættelse af streaming service på server færdig
 - Eksperimentering med object detection
 - Opsættelse transmitter/receiver på arduino
 - Fredag
 - API opsat og begynd opsætning af hjemmeside – API – database forbindelse
 - Object detection virker på pc kamera
 - Kamera sat på AOBE og sende live feed/stream til webserver hosted af kamera wifi modul
 - Deployment diagram lavet
3. Tredje uge
- Mandag
 - Begynd opsættelse af object detection via kamera på AOBE
 - Opsættelse af vertikal step motor på arduino og kamera holder på vertikal step motor
 - Hjemmeside – API – database forbindelse virker
 - Tirsdag
 - Login modul på hjemmeside lavet
 - Begynd opsættelse af tcp forbindelse fra kode til arduino
 - Sende bevægelses commands over til arduino
 - Centrering af objects når de bliver fundet
 - Forsæt overvågnings mode
 - Object detection på AOBE kamera færdiggjort
 - Begyndt på at få det færdige resultat sendt videre til hjemmesiden
 - Begynd dokumentation
 - Onsdag
 - Begynd på at skrive procesrapport
 - Tcp modul til arduino færdiggjort
 - Begynd på at lave test rapport
 - Færdig version af AOBE model/tårn bygget

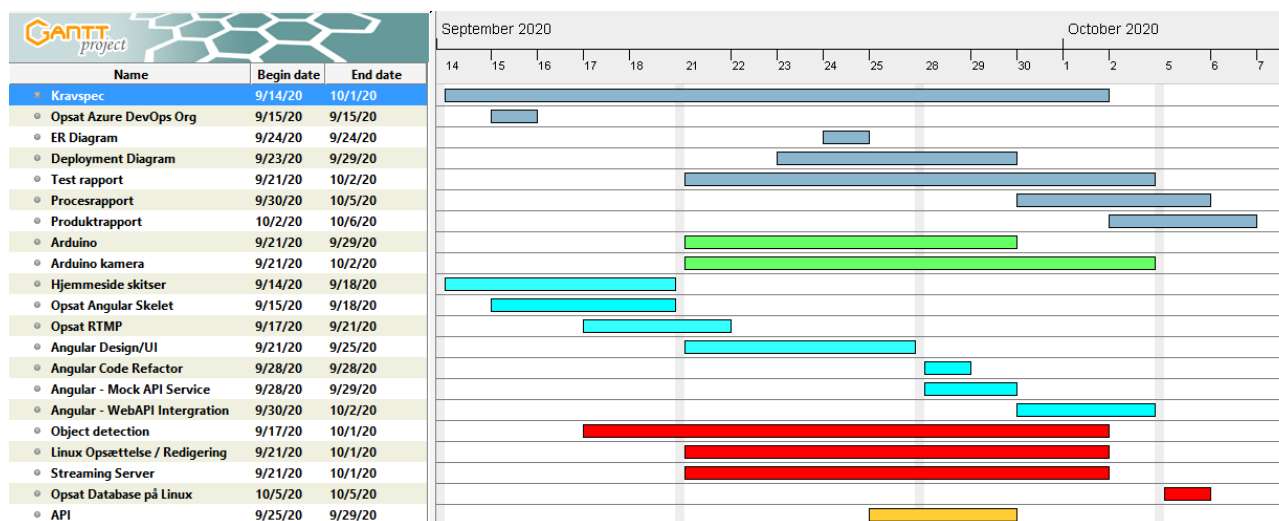
- Torsdag
 - Hjemmeside skal være færdig
 - Object detection feed til hjemmeside skal være færdig
 - Test rapport færdiggøres
 - Begynd på at skrive produkt rapport
- Fredag
 - Test rapport skal være færdig
 - Fortsæt med proces rapport
 - Fortsæt med produkt rapport

4. Fjerde uge

- Mandag
 - Fortsæt med proces rapport
 - Fortsæt med produkt rapport
- Tirsdag
 - Proces rapport skal være færdig
 - Produkt rapport skal være færdig

Author: Benjamin Østner

Realiseret tidsplan



Figur 3 | Den realiserede tidsplan

Arbejdsfordeling

- Kravsspecifikation
 - Benjamin
 - Bjørn
 - Nicklas
 - Nicky

- Procesrapport
 - Benjamin
 - Bjørn
 - Nicklas
 - Nicky
- Produktrapport
 - Benjamin
 - Bjørn
 - Nicklas
 - Nicky
- Deployment diagram
 - Benjamin
 - Nicky
- ER diagram
 - Bjørn
 - Nicky
- Testrapport
 - Bjørn
- Database
 - Bjørn
- API
 - Bjørn
- Arduino
 - Benjamin
 - Bjørn
 - Nicklas – kamera opsættelse
- Streaming server
 - Nicklas
 - Nicky
- Object detection
 - Benjamin – fronten for research småting
 - Nicklas
 - Nicky
- Hjemmeside
 - Nicky

Logbog

Mandag d. 14. september 2020:

- Nicky Hansen
 - Arbejdet på kravspecifikation FURPS+

- Arbejdet på skitser af hjemmeside på iPad.
- Nicklas Pedersen
 - Arbejdet på kravspecifikation FURPS+
- Bjørn Deleuran
 - Arbejdet på kravspecifikation FURPS+
- Benjamin Østner
 - Arbejdet på kravspecifikation FURPS+

Tirsdag d. 15. september 2020:

- Nicky Hansen
 - Arbejdet videre med kravspecifikation FURPS+
 - Begyndt så småt at sætte Angular projekt og Azure DevOps Organisation op.
 - Også opsat alle de andre projekter.
 - Arbejdet på skitser af hjemmeside på iPad.
- Nicklas Pedersen
 - Arbejdet på kravspecifikation FURPS+
- Bjørn Deleuran
 - Arbejdet på kravspecifikation FURPS+
- Benjamin Østner
 - Arbejdet på kravspecifikation FURPS+

Onsdag d. 16. september 2020:

- Nicky Hansen
 - Arbejdet på kravspecifikation FURPS+
 - Arbejdet på skitser af hjemmeside på iPad.
- Nicklas Pedersen
 - Arbejdet på kravspecifikation FURPS+
- Bjørn Deleuran
 - Arbejdet på kravspecifikation FURPS+
- Benjamin Østner
 - Arbejdet på kravspecifikation FURPS+

Torsdag d. 17. september 2020:

- Nicky Hansen
 - Tilføjet Bootstrap/jQuery/popper.js til Angular
 - Tilføjet Angular Router
 - Tilføjet komponenter: Dashboard, Logout, Messages, Navbar, Profile, Settings.
 - Undersøgt krav til livestream video feed sammen med Nicklas.
 - Arbejdet på skitser af hjemmeside på iPad.
- Nicklas Pedersen
 - Undersøgt muligheder for objekt detektion

- Anmodet om maskine til brug af server
- Bjørn Deleuran
 - Arbejdet på kravspecifikation FURPS+
- Benjamin Østner
 - Arbejdet på kravspecifikation FURPS+

Fredag d. 18. september 2020:

- Nicky Hansen
 - Arbejdet på kravspecifikation FURPS+
 - Tilføjet JavaScript HLS-player til hjemmesiden. (Den kan afspille video stream fra et testlink)
 - Arbejdet på skitser af hjemmeside på iPad.
- Nicklas Pedersen
 - Afprøvet muligheder for objekt detektion, på egen maskine
- Bjørn Deleuran
 - Arbejdet på kravspecifikation FURPS+
- Benjamin Østner
 - Arbejdet på kravspecifikation FURPS+

Mandag d. 21. september 2020:

- Nicky Hansen
 - Opsat RTMP NGINX Server.
 - Arbejdet på Angular Cases component.
 - Installeret grafikkort i fysisk server.
- Nicklas Pedersen
 - Installeret grafikkort i fysisk server.
 - Installeret Ubuntu 20.04 på fysisk server.
 - Installeret og konfigureret nginx på fysisk server.
- Bjørn Deleuran
 - Arbejdet med Arduino ESP 32 cam
- Benjamin Østner
 - Kigget på FURPS+
 - Bygget første version af tårn til Arduino
 - Arbejdet med stepper arduino kode

Tirsdag d. 22. september 2020:

- Nicky Hansen
 - Hentet Ethernet-kabel
 - Arbejdet på Cases table design.
 - Arbejdet på Cases modal design.
- Nicklas Pedersen

- Kæmpet med afhængigheder (tensorflow og keras) på fysisk server.
 - Dokumentation
- Bjørn Deleuran
 - Arbejdet med Arduino ESP 32 cam
- Benjamin Østner
 - Kigget på FURPS+
 - Arbejdet med Arduino stepper kode

Onsdag d. 23. september 2020:

- Nicky Hansen
 - Arbejdet på Delete knap og Edit knap.
- Nicklas Pedersen
 - Installerede "Anaconda" for at separere python miljø og få tensorflow til at du på en (meget gammel, fuck den er gammel, sikke noget lort) gpu.
- Bjørn Deleuran
 - Arbejdet med Arduino ESP 32 cam
- Benjamin Østner
 - Ændret på Arduino tårn
 - Arbejdet med Arduino stepper kode

Torsdag d. 24. september 2020:

- Nicky Hansen
 - Arbejdet på View-Case-Modal component.
 - Lavet Databasediagram sammen med Bjørn.
 - Tilføjet Data models til Angular projekt.
- Nicklas Pedersen
 - Grafikkortet er lort, fuck hvor er den lort.
 - Jeg fik den til at virke med en anden model til objekt detektion.
- Bjørn Deleuran
 - Udarbejdet Databasediagram med Nicky.
 - Oprettet lokal database ud fra diagram
 - Begyndt udvikling af Node.js REST API
- Benjamin Østner
 - Arbejdet med Arduino kode
 - fik Camilla til at bestille FTDI oversættere

Fredag d. 25. september: 2020

- Nicky Hansen
 - Arbejdet på Case service.
 - Arbejdet på Cases component.
 - Arbejdet med Edit Case Modal

- Nicklas Pedersen
 - Jeg fik den til at kunne hente billeder samtidig med at den processerer dem, ved hjælp af tråde, dette skal til fordi at man vil helst have det nyeste billede.
- Bjørn Deleuran
 - Arbejdet med REST API
- Benjamin Østner
 - Arbejdet med Arduino transmitter/receiver

Mandag d. 28. september 2020:

- Nicky Hansen
 - Refactoret en del i Angular koden, renamet Case til Incident.
 - Oprettet Mock services til kommunikation med API.
 - Begyndt på Cameras component (som skal refactors til AOBes)
 - Tilføjet Search input, table samt AOB E modals til Cameras component.
- Nicklas Pedersen
 - Fik kameraet/AOB E'en koblet op så serveren (i gennem ffmpeg, et program) kan hente billeder ned fra den.
- Bjørn Deleuran
 - Arbejdet med REST API
 - Test af API mod lokal mock database
- Benjamin Østner
 - FTDI oversætter ankom og kamera virker
 - Skrevet dokumentation
 - Lavet holder til kamera på tårn

Tirsdag d. 29. september 2020:

- Nicky Hansen
 - Arbejdet på Modal components i Cameras/AOB E component.
 - Omskriv IncidentStatusService til at bruge Promises.
 - Arbejdet på "Opret Ny" modale komponenter.
- Nicklas Pedersen
 - Bruger programmet "ffmpeg" til at tage billeder fra kameraet og sende det til rtmp serveren.
- Bjørn Deleuran
 - Test af API
- Benjamin Østner
 - Testet threading på Arduino
 - Ombygget tårn så et større batteri kunne side I det
 - Kamera sat op på tårn og virker
 - Sat mig ind i hvordan streaming serveren virkede, hvordan object detection fik et live feed og sendte det til streaming serveren.

Onsdag d. 30. september 2020:

- Nicky Hansen
 - Arbejdet på at integrere API med Angular Projektet.
- Nicklas Pedersen
 - Konfigureret serveren til at man nu kan hente video ned med HLS (HTTP Live Streaming.)
- Bjørn Deleuran
 - Arbejde med databasen (mock data)
 - Opsætning af FTP-server på Arduino ESP32 cam
- Benjamin Østner
 - Arduino kan ikke køre threading eller events
 - Vertikal motor fjernet, da der ikke er tid til at rode mere med det
 - Laver deployment diagram
 - Tidplan til procesrapport er skrevet
 - Arbejdsfordeling til procesrapport er skrevet

Torsdag d. 1. oktober 2020:

- Nicky Hansen
 - Oprettet Use Cases til Kravspecifikation.
- Nicklas Pedersen
 - Fik Python programmet til at lave et underprogram som kalder ffmpeg til at hente billederne ned på.
- Bjørn Deleuran
 - Tandlæge (løgn ifølge benjamin)
 - Rettelse af bugs i API
- Benjamin Østner
 - Skrevet procesrapport

Fredag d. 2. oktober 2020:

- Nicky Hansen
 - Lavet ER-Diagram over AOB E-system.
 - Opdateret mine domæne-modeller i Angular-Frontend projektet og udredet/løst forskelligheder mellem API og Frontend med Bjørn.
- Nicklas Pedersen
 - Python programmet kan nu hente billedere direkte ned fra kameraet i stedet for at den skal gå igennem RTMP serveren.
- Bjørn Deleuran
 - Rettelse af bugs i API
- Benjamin Østner
 - Skrevet procesrapport

Mandag d. 5. oktober 2020:

- Nicky Hansen
 - Arbejdet på Realiseret Tidsplan i Processrapport
- Nicklas Pedersen
 - Brugt Bjørns database script til at oprette og opsætte en database.
 - Skrevet dokumentation.
- Bjørn Deleuran
 - Arbejdet med Testrapport
- Benjamin Østner
 - Skrevet procesrapport

Tirsdag d. 6. oktober 2020:

- Nicky Hansen
 -
- Nicklas Pedersen
 -
- Bjørn Deleuran
 - Færdiggjort Testrapport
- Benjamin Østner
 - Skrevet produktrapport

Bilag

1 [Artikel omkring krybskytteri](#)

2 [Object Detection Library](#)

3 [Forward-Looking infrared](#)