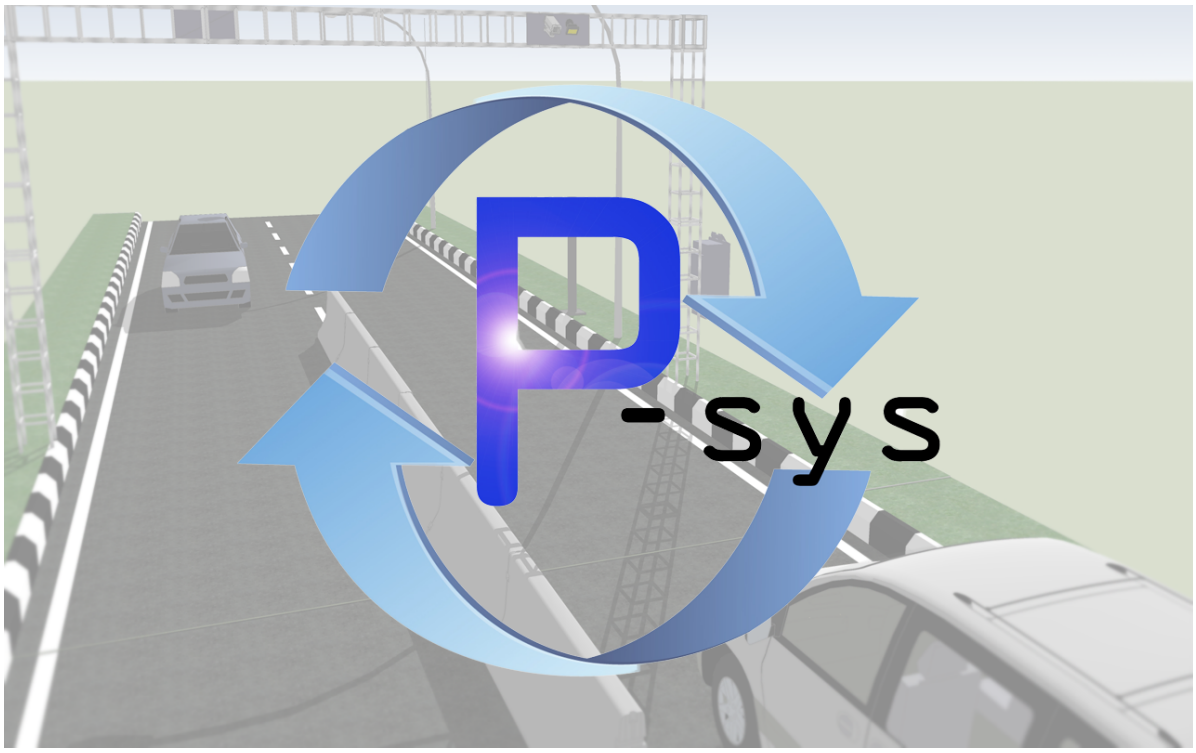
BACHELOROPPGAVE

P-sys, parkeringssystem med automatisk skiltgjenkjenning



Av
(11) Magnus Pihlstrøm
(15) Lars Fredrik Tvinde
(16) Øystein Berge
(9) Lars Magnus Kaasa

P-sys, parking system with automatic number plate recognition

Avdeling for Ingeniør- og naturfag
HO2-300 Hovedprosjekt
23.05.2014

HØGSKULEN i SOGN OG FJORDANE

1.0 Referanseside

TITTEL	RAPPORTNUMMER	DATO
HO2-300 Hovedprosjekt		23.05.2014
PROSJEKTTITTEL	**TILGJENGELIGHET**	**ANTALL SIDER**
P-sys, parkeringssystem med automatisk skiltgjenkjenning	Åpen	62 (uten vedlegg)
FORFATTERE	**STYRINGSGRUPPE**	
Magnus Pihlstrøm, prosjektleder Lars Fredrik Tvinde Øystein Berge Lars Magnus Kaasa	Joar Sande, prosjektansvarlig Kjell Otto Gjesdal, kontaktperson oppdragsgiver	
OPPDRAGSGIVER		
Førde lufthamn Bringeland		

SAMMENDRAG

Dette er et prosjekt i faget *HO2-300 – Hovedprosjekt* ved HiSF-AIN våren 2014. Prosjektet er et samarbeid mellom prosjektgruppen og Førde lufthamn Bringeland.

Oppdragsgivers ønske er å erstatte det nåværende parkeringsavgiftsystemet med et automatisk avgiftssystem. Avgiftssystemet skal regne ut hver enkelt parkeringstid basert på automatisk registrering av ankomst og avreise. Registreringen foregår med bildetaking ved inn- og utkjøring til parkeringsanlegget. Ved avreise regnes totaltid. Totaltiden knyttes mot bruker i en brukerdatabase.

Det er laget en prototype som tilfredsstiller oppdragsgivers spesifikasjoner. Programmet registrerer bilskilt og knytter dette opp mot en eier for så å regne ut parkeringstid.

Gruppen har bygget en prototype som er blitt testet i høgskolens garasje. I denne rapporten dokumenteres hvilke løsninger som er valgt, resultat med de valgte løsninger og hvordan en kom frem til resultatet.

EMNEORD

Skiltgjenkjenning, Tegngjenkjenning, OpenCV, JavaCV, Java

2.0 Forord

I løpet av vårsemesteret 3. året ved automasjonslinjen på Høgskulen i Sogn og Fjordane (HiSF), skal studentene gjennomføre et prosjekt i forbindelse med faget *HO2-300 – Hovedprosjekt vår 2014*. Prosjektet er et fordypningsemne som skal være praktisk og/eller teoretisk. Det oppfordres til å finne et prosjekt selv, eller finne en oppdragsgiver. Det er krav om at prosjektet skal innbefatte prosjektbeskrivelse, forprosjekt, midtveispresentasjon, plakat, pressemelding og muntlig presentasjon.

Prosjektgruppen kom selv opp med ideen om automatisk registrering av parkeringstid. Etter samtale med oppdragsgiver Kjell Otto Gjesdal, lufthavnsjef ved Førde lufthamn Bringeland, ble oppgaven formet og spesifisert. Oppdragsgivers ønske er å erstatte det nåværende parkeringsavgiftsystemet med et automatisk avgiftssystem. Avgiftssystemet skal regne ut parkeringstid basert på automatisk registrering ved ankomst og avreise. Registreringen foregår med bildetaking ved inn-/utkjøring til parkeringsanlegget. Ved avreise regnes totaltid. Totaltiden knyttes mot bruker i en brukerdatabase, for så å kunne sende faktura til kundene. Det var opp til prosjektgruppen å finne en løsning som tilfredsstiller spesifikasjonene.

Prosjektet har resultert i en prototype som registrerer biler ved ankomst og avreise. Prototypen knytter bil mot eier og regner ut tid bilen har vært på parkeringsanlegget. I rapporten dokumenteres oppsett/metode, beskrivelse av prototypen, utvikling av prototypen og konklusjon.

Det rettes en stor takk til:
Marcin Fojcick for god rettledning ved programmering.
Preben Nes for gode råd under rapportskriving.
Joar Sande for å komme med god rettledning under statusmøter.
Inger Margrethe Hove Ås for lån av høyskolens garasje til test av systemet.
Kjell Otto Gjesdal for gode retningslinjer for et ferdig system.

Gruppen består av:

Magnus Pihlstrøm	Prosjektleder
Lars Magnus Kaasa	Student
Lars Fredrik Tvinde	Student
Øystein Berge	Student

Styringsgruppe:

Joar Sande	Prosjektansvarlig
Kjell Otto Gjesdal	Kontaktperson oppdragsgiver

Førde, 23.05.2014

Magnus Pihlstrøm

Lars Magnus Kaasa

Lars Fredrik Tvinde

Øystein Berge

3.0 Sammendrag

Denne bacheloroppgaven er utført av fire avgangsstudenter ved Høgskulen i Sogn og Fjordane våren 2014. Gruppen kom selv opp med en idé til automatisk registrering av parkeringstid. Førde lufthamn Bringeland ble kontaktet og oppgaven ble spesifisert.

Målet for prosjektet er å utvikle og teste en prototype for automatisk skiltgjenkjenning som kan videreutvikles til å erstatte det nåværende parkeringsavgiftsystemet ved Førde lufthamn Bringeland. Programmet skal registrere tidspunkt for ankomst og avreise. Registreringen foregår ved bildetaking. Basert på dette skal tid hver bruker har vært parkert på området regnes ut. Førde lufthamn krever at registrering av biler inn/ut foregår uten at kjøretøyet må stoppe.

I forprosjektperioden skaffet prosjektgruppen en oversikt over programmer og metoder som kunne brukes. Dette ble gjort ved undersøkelser og test av enkeltprogrammer. Testperiode ble planlagt og det ble avtalt mellom gruppen og høgskolen at testing skulle foregå i skolens garasje.

For registrering av bil inn/ut brukes et IP-kamera aktivert av to ultralydsensorer, programmert i Arduino. Farten til bilene reguleres ved hjelp av lysindikasjon. Rødt lys indikerer at farten må senkes til maksimalt 10 km/t. Et Java-program gjennomfører en rekke bildebehandlinger og finner skiltrammen. Når skiltet er detektert ved innkjøring, leses det av med et tegngjenkjenningsprogram via Java. Dette registreringsnummeret lagres i en parkeringsdatabase konstruert i MySQL. Avlest registreringsnummer kobles mot eier og faktureringsadresse i brukerdatabasen. Brukerdatabasen fyller brukeren inn selv ved å registrere seg ved første besøk. Ved avreise blir nytt bilde tatt og registreringsnummer registrert. Tidspunkt for avreise lagres og total parkeringstid regnes ut.

Brukergrensesnittet ble testet med 20 testpersoner, som ga en tilbakemelding på forbedring i det grafiske og funksjonelle.

Testfasen av bildebehandlingsdelen avdekket at lysforholdene og bildekvalitet er en viktig faktor. Dette er testet ved å sammenligne bildebehandlingsresultat av bilder tatt ute mot bilder tatt i garasjen. Bildene tatt ute er under bedre lysforhold og med bedre bildekvalitet enn bildene i garasjen.

Resultatet er en fungerende prototype. For å teste systemet videre anbefaler gruppen å bruke et kamera som gir bedre bildekvalitet. Videre testing må foregå ute under bedre lysforhold. For å inn- og utregistrere biler i raskere hastighet må det brukes bedre sensorer og et raskere kretskort.

4.0 Innhold

5.0 Innledning

I dette kapittelet forklares det hva oppgaven går ut på, hvorfor dette er aktuelt for Førde lufthamn, hvordan oppgaven løses, resultatet og rapportens oppbygging. Til slutt er det klargjort avgrensninger gruppen må ta hensyn til.

5.1 Oppgave

Prosjektgruppen skal på oppdrag for Førde lufthamn Bringeland utvikle og teste en prototype med automatisk skiltgjenkjenning. Prototypen skal registrere tidspunkt for ankomst og avreise. Basert på disse opplysningene skal det regnes ut hvor lang tid hver bruker har vært parkert på området. Under møte med oppdragsgiver, se Vedlegg 12, kreves det at ingen biler må stoppe for å bli registrert.

Lufthavnsjefen ved Førde lufthamn, Kjell Otto Gjesdal, mener det blir brukt for mye ressurser på systemet som er i dag. Et automatisk system vil kunne kutte ned på tid brukt til klagehåndtering som følge av bøter og billettkontroll.

Systemet skal registrere tidspunkt for ankomst av biler ved bildetaking. Bildet skal bildebehandles slik at registreringsnummeret er leselig for et tegngjenkjenningsprogram. Avlest skilt skal knyttes opp mot en bruker i en brukerdatabase. Ved avreise skal nytt bilde bli tatt og samme behandling utføres. I parkeringsdatabasen knyttes de avleste registreringsnumrene sammen og en får basert på dette parkeringstid. Denne tiden knyttes så mot bruker i brukerdatabasen.

Det er i løpet av prosjektperioden utviklet et program som fungerer som over. Brukerdatabasen fylles ut ved at brukeren registrerer seg første gang. Gruppen har ved testing funnet ut at programmet er avhengig av riktige lysforhold og god bildekvalitet.

I denne rapporten er det først beskrevet arbeidsmetoder og oppsett av prototype. Videre vil prototypen beskrives. Det avsluttes med en drøfting av utviklingsprosessen av prototypen. Til slutt er det skrevet en konklusjon basert på observasjoner underveis.

5.2 Avgrensinger

Programmet skal gjenkjenne norskregistrerte biler med rektangulære skilt med bokstaver og siffer på en enkelt linje. Skiltene har sort skrift på hvit bakgrunn.

Av økonomiske årsaker kan det ikke brukes en eksisterende kjøretøysdatabase. Derfor må en brukerdatabase bygges ved at brukeren registrerer seg selv ved førstegangsbesøk. Brukerdatabasen er tilpasset for norske registreringsnummer med to bokstaver og fem siffer.

Det er ikke tatt hensyn til biler med tildekket registreringsskilt, som snø eller skitt.

Det vil i programmet ikke klargjøres for sending av faktura annet enn utregning av totaltid og registrering av brukerens adresse.

I prosjekt i faget *Prosjektstyring med prosjekt*, høsten 2013 (Vedlegg 1) er det tatt opp utfordringer i hvilken grad dette er overvåking, vil kreve utvidelse av vei eller om det er behov for montering av mekanisk oppheng. Problemstilling omkring sletting av bilder etter hvert for opprettholdelse av krav i *Personopplysningsloven* vil ikke løses i denne prototypen.

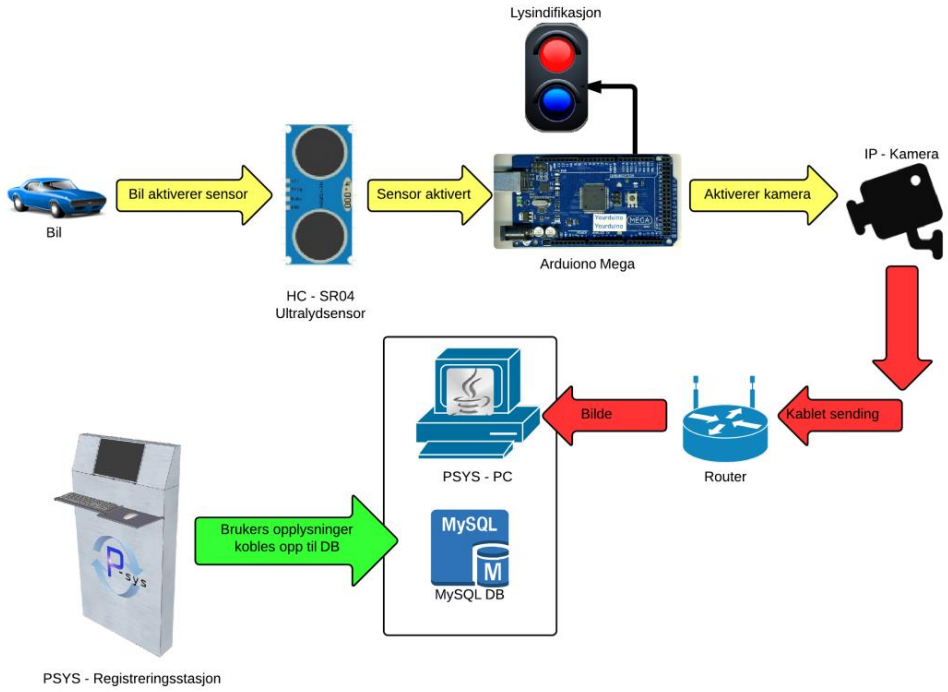
5.3 Fagterminologi og forkortelser

HiSF	Høgskulen i Sogn og Fjordane
HiSF-AIN	Avdeling for Ingeniør og Naturfag ved Høgskulen i Sogn og Fjordane
JAVA	Programmeringsspråk og metode
JAVA-prosjekt	En programdel av totalprogrammet
SWING	Grafisk verktøysett for JAVA
Skew	Skråstilling/vridning av bilde
Piksel	Et enkeltpunkt i bildet
Netbeans	Utviklingsverktøy for JAVA, php, C/C++
Bibliotek	Samling av funksjoner og klasser. Her brukt Newping og OpenCV
Algoritme	Gjenbrukbar funksjonalitet for programmering
Åpen kildekode	Software hvor kildekoden er åpen til offentligheten
Wrapper	Grensesnitt som muliggjør bruk av bibliotek på tvers av programmeringsspråk
C++	Programmeringsspråk
Database	Strukturert samling av relaterte data
MySQL	Program for databasekonstruksjon
OCR	Optical Character Recognition (No. Optisk Tegn Gjenkjenning).
Språkpakke	Mal med opplærte tegn brukt av tegngjenkjenningsprogram
Virtuell tabell	Eng; View. Viser data i utvalgte kolonner fra basistabeller
Databehandler	Den som behandler personopplysninger på vegne av Førde lufthamn Bringeland
Primærnøkkel	ID som binder sammen tabeller i en database. Eng; Primary key
Registreringsnummer	Tegn som skal avleses, bestående av to bokstaver og fem siffer
Skiltplate	Område som kun inneholder registreringsnummer
Skiltramme	Rammen rundt skiltplaten
Registreringsskilt	Avgrenset område på bil som inneholder skiltramme, skiltplate og registreringsnummer
Sekvens (ultralydsensor)	Tid fra sending til mottak av puls
Trigger	Puls som utløser kamera

6.0 Arbeidsmetode og oppsett

Prosjektgruppen beskriver i dette kapittelet utstyr og programmer som brukes for å konstruere en prototype for automatisk skiltgjenkjenning. Disse delene ble valgt av gruppen ved undersøkelser og testing under forprosjektperioden våren 2014, lagt til som Vedlegg 2. Eneste avviket fra det som ble valgt da, er at gruppen har benyttet MySQL fremfor Wampserver til konstruering av databaser.

Figur 6.1 viser hvordan av deler i prototypen er satt sammen.



Figur 6.1 Oppsett av prototype

6.1 Programmering

Programmeringsspråket som brukes er Java. Gruppen bruker Netbeans til å programmere i Java. Det importeres også eksterne bibliotek.

6.1.1 OpenCV

OpenCV er et åpent bibliotek som brukes, både faglig og kommersielt, til bildebehandling.

CV (Computer Vision) er et fellesnavn for programmer som tilegner, prosesserer, analyserer og forstår bilder. OpenCV er en åpen kildekode innenfor denne kategorien.

OpenCV er skrevet i C++, men har også grensesnitt for bruk i Java og Python [1].

6.1.2 JavaCV

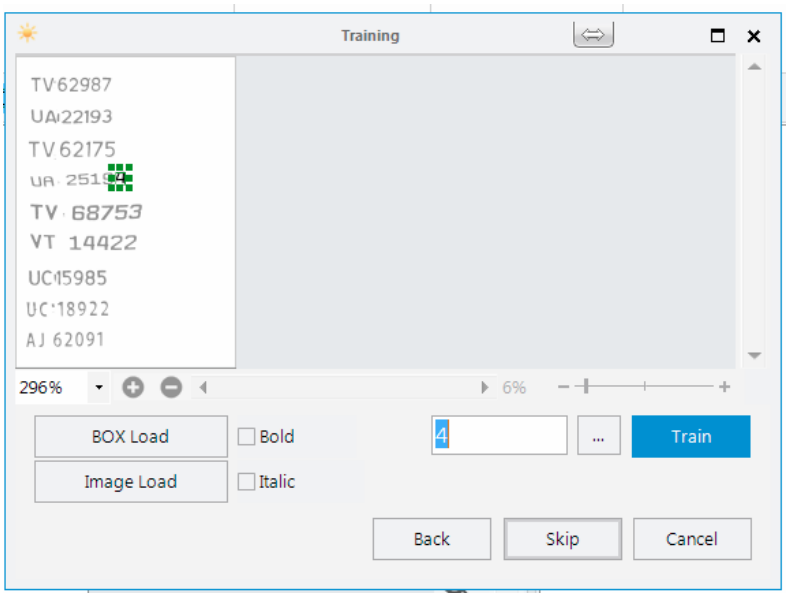
JavaCV er en *wrapper*. Dette gjør at det er mulig å overføre metoder og kommandoer fra OpenCV til Java-biblioteket [5].

6.2 MySQL

MySQL er en av verdens mest brukte SQL-databaseprogram [3]. SQL gir muligheter til å enkelt kunne få tilgang til og endre data i databasen [3]. Denne brukervennligheten er et viktig kriterium for at gruppen gikk bort fra Wampserver, som ble valgt i forprosjektperioden.

6.3 Tegngjenkjenning

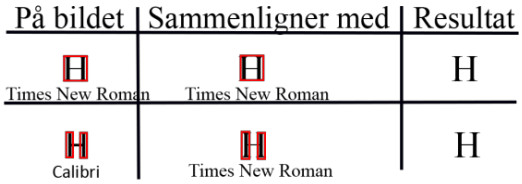
Tesseract OCR er en gratis tegngjenkjenningsmotor som er forhåndsopplært med enkelte språkpakker. Eldre registreringsnummer blir ikke lest fordi disse bruker en skrifttype som ikke er en del av språkpakkene [3]. For å kunne lese denne skrifttypen, må en lære opp motoren til å gjenkjenne tegnene. Dette gjøres med SunnyPage. I prosjektet brukes SunnyPage kun som grensesnitt for opplæringen. Selve avlesingen gjøres ved at Tesseract OCR, med egendefinert språkpakke, legges til i Java. For å bruke Tesseract OCR i Java, benyttes wrapperen Tess4j.



Figur 6.2 SunnyPage under opplæring av Tesseract OCR

Figur 6.2 viser opplæringen i SunnyPage. På denne måten legges tegn til i en språkpakke for bruk i avlesingen.

Metoden som Tesseract OCR bruker er Feature Extraction. Denne metoden ignorerer de mest variable delene i et tegn. Et eksempel er bokstaven "H" i Figur 6.3. "H" består alltid av to parallelt vertikale linjer og en horisontal linje. I algoritmen vil den se etter likheter med den i språkpakken [4]. Med Feature Extraction får en resultat selv om skriften i malen er litt forskjellig fra avlest bilde.



Figur 6.3 Feature Extraction

6.4 Bilregistrering

Utstyret gruppen bruker for å registrere ankomst og avreise av bilene, er i dette kapittelet.

6.4.1 Ultralydsensor

For å detektere bil inn/ut er det brukt en sensor av typen HC-SR04 illustrert i Figur 6.4. Sensoren har en sender og en mottaker. Sensoren virker best innenfor en vinkel på 15 grader og har en rekkevidde på opp til 5,00m. Detaljerte spesifikasjoner er gitt i Vedlegg 3.

Figur 6.4 HC-SR04

6.4.2 Arduino

Arduino er en utviklingsplattform med åpen kildekode. Det brukes ofte til å motta signal fra sensorer og kontrollere lyskilder [6].

6.4.3 Kamera

Kameraet gruppen har valgt, er et IP-kamera av typen Vivotek IP8332. Det er klassifisert med IP66 og tåler temperaturer ned til -20 grader. Med HD-oppløsning på 1200x800 piksler og en lav lysfølsomhet er det mulig for bruk under mørke omgivelser. Flere opplysninger er i Vedlegg 4.

Figur 6.5 Vivotek IP8332

7.0 Beskrivelse av prototypen

I dette kapittelet beskrives hvordan systemet fungerer. Det forklares hvordan en finner registreringsskiltet og klargjør denne for avlesing av tegngjenkjenningsprogrammet. Hvordan bildet blir tatt og knyttes opp mot en bruker er også beskrevet her. Databasenes funksjoner og struktur er en del av resultatet. Algoritme for prototypen er gitt i Vedlegg 5. Programmene med installasjonsinstrukser for prototypen er vedlagt på egen CD.

7.1 Finne registreringsskiltet

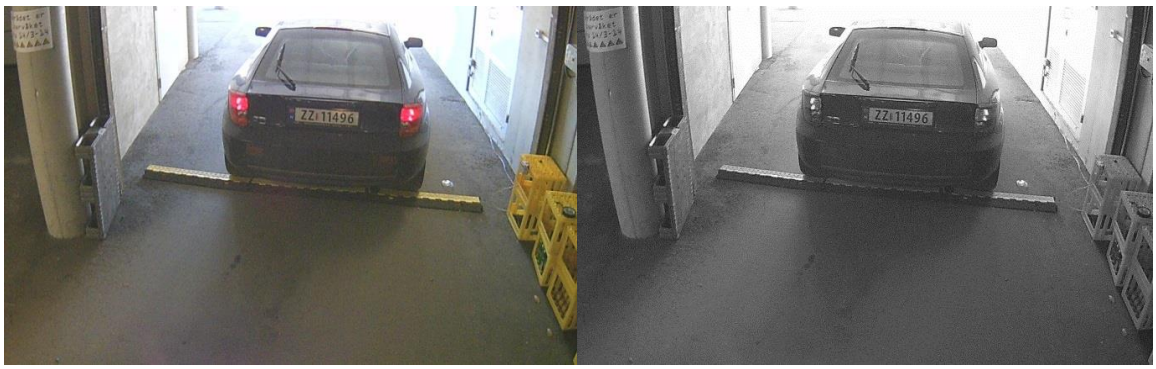
I bildet som blir tatt, er det første målet til programmet å finne registreringsskiltet på bildet. Dette gjøres ved å bruke rektangelgjenkjenning. Det letes etter konturer i bildet som passer innenfor et rektangel, på størrelsen med et registreringsskilt. Konturene i bildet finnes etter bildebehandlingene Greyscale, Gaussian Blur og Adaptive Threshold.

7.1.1 Greyscale

Det første som gjøres når et bilde skal bildebehandles er å fjerne farger. Dette gjøres med funksjonen Greyscale. Denne funksjonen gråskalerer bildet alt etter hvor mye lys det er i den enkelte piksel [5].

Figur 7.1 viser nivåforskjellene i grensene som blir gjort ved gråskalering. I Figur 7.2 ser en til venstre originalbildet tatt med kamera og til høyre det samme bildet etter gråskalering.

Figur 7.1 Eksempel på skala for gråskalering



Figur 7.2 Bilde gråskalert

Dette gjøres for å finne lysintensiteten i bildet. Prosjektgruppen har erfart at neste steg, Gaussian Blur, dermed blir gjennomført på en bedre måte enn hvis det hadde vært gjort direkte fra originalbildet.

7.1.2 Gaussian Blur

Gaussian Blur fjerner støy og reduserer detaljer i bildet. Bildet blir mindre skarpt.

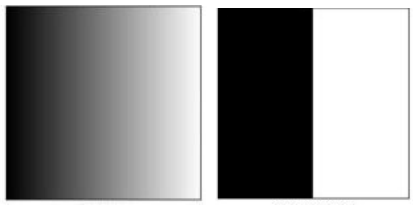
Figur 7.3 Bilde behandlet med Gaussian Blur

I Figur 7.3 ser en til høyre bildet etter Gaussian Blur – behandlingen. Til venstre er bildet gråskalert.

Med Gaussian Blur blir det mindre detaljer i bildet. Dermed blir det færre konturer å finne ved neste steg.

7.1.3 Adaptive Threshold

Endring av terskelverdi er en bildebehandlingsteknikk der en velger ut pikselverdier innenfor en grense. Teknikken finner objekter med en viss lysstyrke innenfor et interessant område . Dette blir gjort for å omforme gråskalerte bilder til binære bilder [6]. Teknikken brukt i programmet er Adaptive Threshold. Piksler som holder seg innenfor terskelverdiene blir hvitt, mens pikslene utenfor blir sort. Figur 7.4 viser et eksempel på dette.



Figur 7.4 Eksempel omgjøring gråskalert til binært

I Figur 7.5 ser en hvilken forskjell teknikken gjør med et gråskalert bilde. Denne behandlingen er nødvendig for å detektere registreringsskiltet.

Figur 7.5 Bilde behandlet med Adaptive Threshold

7.1.4 Lokalisere registreringsskiltet

Når Adaptive Threshold er gjennomført, kan programmet lete etter konturer formet som et rektangel. Siden det er bestemt et fotoområde med en fotolinje for plassering av registreringsskiltet, se *8.1.1 Fotolinje*. Dermed vil størrelsen på registreringsskiltene være tilnærmet lik på alle bildene. Størrelsen til rektanglene må være 30 – 50 piksler i høyde og 120 – 180 piksler i bredde for å bli gjenkjent. På denne måten unngår en å finne rektangler som ikke er relevante.

Figur 7.6 Nederste og øvre grense for størrelse registreringsskilt

I Figur 7.6 ser en grensene for gjeldene rektangler som er tilpasset fotoområdet. Rektangler som er mindre enn det røde eller større enn det sorte rektangelet, er ikke gjeldene.

Når registreringsskiltet er funnet markeres dette i en kopi av originalbildet. Denne delen klippes ut og lagres som et eget bilde for neste behandling.

Figur 7.7 Bilde med markert registreringsskilt

I Figur 7.7 er området markert hvor det er funnet et rektangel. Området innenfor markeringen lagres som et eget bilde.

7.2 Bildebehandle registreringsskiltet

Når registreringsskiltet er funnet, må dette behandles. Støy og unødvendige detaljer må fjernes, før en spesifiserer enda mer nøye hva som skal avleses.

7.2.1 Median Filter

Programmet henter bildefilen som ble lagret for videre bildebehandling. Bildet som hentes er vist i Figur 7.8.

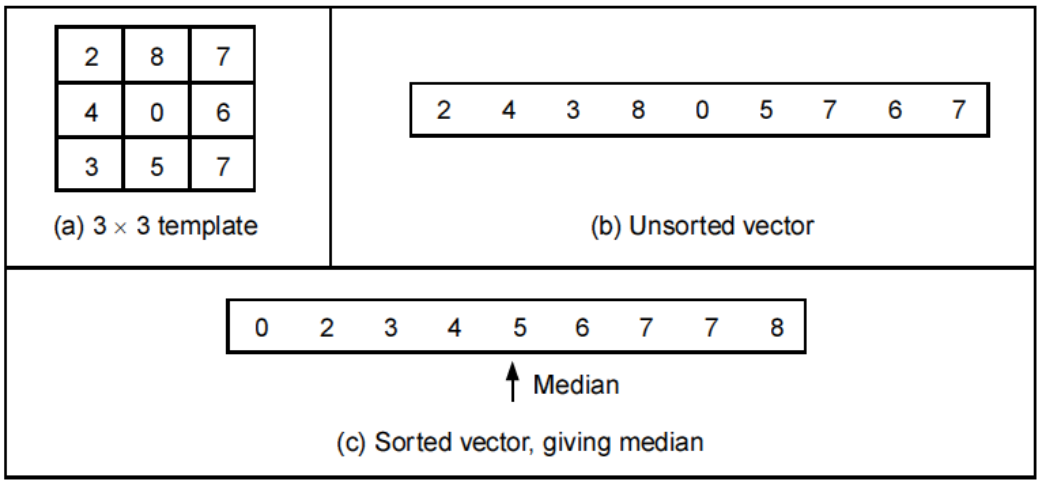
Figur 7.8 Bilde hentet for behandling

Bildet gjøres sort/hvitt, ved pikselverdier og grense for sort og hvitt. Det blir konvertert til et binært bilde. RGB-verdier over 140 blir hvit og under blir sort. I Figur 7.9 ser en bildet etter konvertering.

Figur 7.9 Bilde gjort sort/hvitt

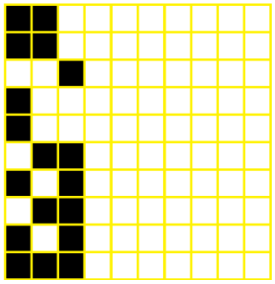
Median Filter brukes til å filtrere ut støy på sort/hvitt-bilde. På denne måten fjernes små punkter som ikke er av interesse i bildet.

I Median Filter blir pikslenes RGB-verdi sammenlignet med pikslene rundt. Metoden finner medianen og setter dette som verdi for området [8].



Figur 7.10 Mal Median Filter

Figur 7.10 viser et eksempel på Median Filter med en 3 x 3 mal. Medianen i området som er testet er fem. Dermed vil alle punktene rundt bli satt til fem. I vår bildebehandling er fargene enten sort eller hvit. I Figur 7.11 er de sorte pikslene i undertall, så dette området blir satt til hvitt.



Figur 7.11 Eksempel på 10x10 mal

I Figur 7.9 ser en et registreringsskilt før behandling med Median Filter. Figur 7.12 er etter behandlingen. En ser at støy har blitt fjernet fra bildet.

Figur 7.12 Registreringsskilt behandlet med Median Filter

7.2.2 Fjerne skiltrammen

For at tegngjenkjenningsprogrammet skal gi riktig resultat, må det meste av skiltrammen fjernes. Dette gjør gruppen med linjedeteksjon.

For å finne området som skal beskjæres, finner en først øvre grense. En teller hvite punkter horisontalt i bildet, pikselrad for pikselrad, ovenfra og ned til midten. En sammenligner disse tellingene mot hverandre og finner maksimalt antall hvite punkter i en rad. Det maksimale antallet blir så brukt til å regne ut et søkeområde for grense. Øvre grense blir den første pikselraden som inneholder 80% hvite punkter av det som er maksimalantallet. Denne grensen blir markert på bildet med rød linje. Nedre grense blir funnet ved å telle fra nederst og oppover. Nedre grense blir markert med grønn linje.

Å finne sidegrensene startes på samme måte som for øvre og nedre grense. Denne gangen blir derimot grensen 30% av maksimal antall hvite punkter. Men denne gangen er grensen avhengig av antall hvite punkter i pikselraden før. For å sette en pikselrad som sidegrense, må to krav opprettholdes:

1. Pikselraden må inneholde 30% hvite punkter mot raden med flest hvite punkter.
2. Endring av antall hvite punkter i raden må være større enn en. Altså første linje som følger utregning i Formel 1:

$$\frac{Antall\ hvite\ punkter\ nåværende\ linje}{Antall\ hvite\ punkter\ forrige\ linje} > 1{,}0$$

Formel 1

I Figur 7.13 ser en beskjæringslinjene markert.

Figur 7.13 Grenser satt etter telling av hvite piksler

Programmet finner koordinatene for krysningspunktene mellom beskjæringslinjene. Med disse koordinatene regnes bredde og høyde ut. Beskjæringen foregår i henhold til disse målene, med utgangspunkt i krysningspunktet oppe til venstre. I Figur 7.14 ser en bildet beskjært og klart for avlesing.

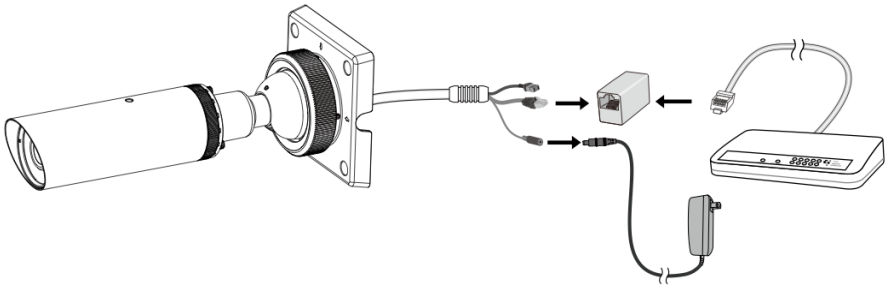
Figur 7.14 Bilde klart til tegngjenkjenning

7.3 Bilregistrering

I dette kapittelet beskrives hvordan registrering av bilene foregår. Det forklares prosesser som foregår og handlinger dersom feil forekommer.

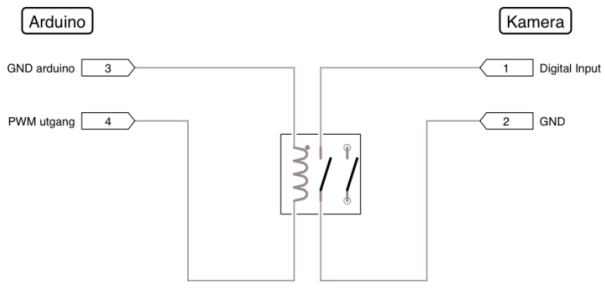
7.3.1 Registrering inn og ut

IP-kameraet er tilkoblet en router som danner et lokalt nettverk. Kommunikasjonen mellom kamera og router går via en kabel. Figur 7.15 viser hvordan kamera og router er koblet sammen. Kameraet er installert og konfigurert på PC ved hjelp av softwaren som fulgte med kameraet. Under konfigurasjonen er installasjonsanvisningen i Vedlegg 6 fulgt. Etter installasjonen er kameraet tilgjengelig i det lokale nettverket med egen IP-adresse.



Figur 7.15 Kobling mellom kamera og router

Kameraet er avhengig av å ha en trigger for å kunne ta stillbilder. Det er ved bruk av softwaren som følger med kameraet konfigurert en hendelse. Hendelsen er at når den digitale inngangen på kameraet går fra "lav" til "høy" tar kameraet bilde. Det er sensorene som brukes til å aktivere et rele til å styre denne hendelsen. Når sensorene aktiveres, aktiveres også releet. Da går den digitale inngangen i kameraet til "høy". Det er satt opp en forsinkelse på ett sekund fra en hendelse er startet til den tidligst kan registrere neste hendelse. Grunnen til dette er for å ikke ta unødvendig mange bilder dersom kameraet mottar flere triggersignaler på samme bil. Figur 7.16 viser koblingsskjema mellom rele og kamera.

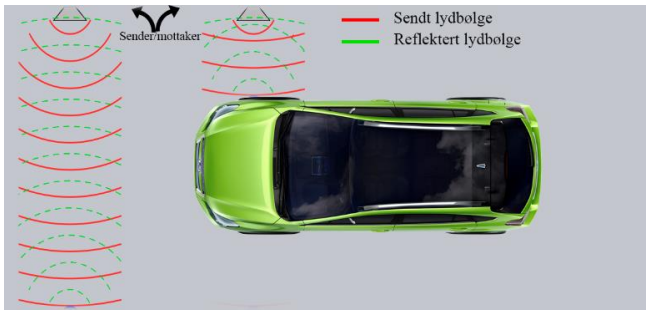


Figur 7.16 Oppkobling mellom arduino og IP-kamera

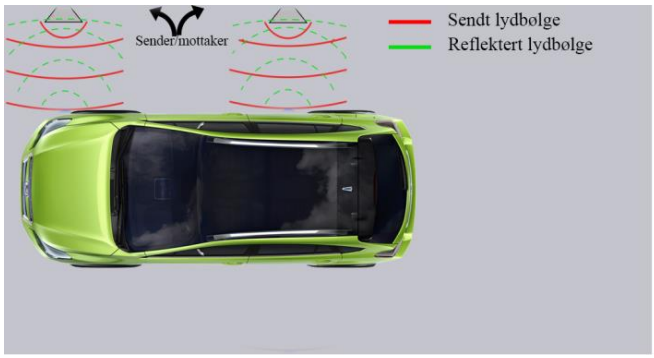
I prototypen har det blitt brukt to sensorer, dette er for å kunne skille bil som kjører inn fra bil som kjører ut. Arduino-programmet er laget slik at når begge sensorene blir aktivert innenfor et tidsrom på fem sekunder, vil kamera ta bilde. Rekkefølgen på aktivering av sensorene, gir informasjon om hvilken retning bilen kjører. Dersom sensor 1 aktiveres før sensor 2, kjører bilen inn, og motsatt dersom sensor 2 aktiveres før sensor 1.

Figur 7.17 Oppsett system ved ankomst garasje del 1

I Figur 7.17 ser en hvordan systemet er satt sammen ved ankomst garasje. Sensorene er markert med sirkler merket "1" og "2". Markering "3" er koblingsboks med Arduino – kort og styring for sensorene, lysindikasjon, markert "4", og kamera.



Figur 7.18 Bil på vei inn/ut. En sensor aktivert.



Figur 7.19 Bil på vei inn/ut. To sensorer aktivert.

I Figur 7.18 aktiveres en sensor, bilen har da fem sekunder på seg til å aktivere neste sensor, Figur 7.19. Da vil bilde bli tatt og informasjon om ankomst eller avreise blir sendt til javaprogrammet. Bildet lagres på en nettverksdisk. Denne disken er på samme PC som Java-programmet ligger på. Programmet finner siste bildet som er blitt tatt og leser siste melding sendt fra Arduino. Dermed vet programmet om siste bildet er tatt i forbindelse med ankomst eller avreise.

I Figur 7.20 er det markert plassering av lysindikasjon("4") og kamera("5").

Figur 7.20 Oppsett system ved ankomst garasje del 2

For å kunne ta bildene når bilene er ved riktig plassering, må farten være maksimum 10km/t. For at brukeren skal vite når farten skal senkes, er det brukt lysindikasjon. Når første sensor aktiveres, tennes et rødt lys. Dette vil være tent andre sensor er aktivert. Bilde blir tatt og et blått lys tennes. Ved blått lys kan farten økes.

7.3.2 Bildehåndtering

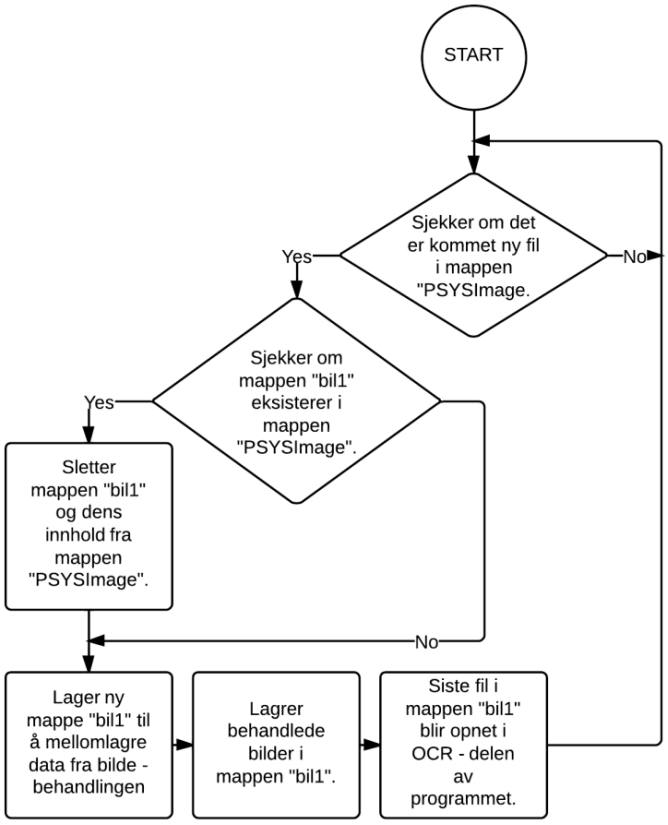
Når en ny bil blir tatt bilde av, blir dette originalbildet lagret i en mappe, "PSYSImage". Dette er hovedmappen for bildene som blir brukt i bildebehandlingen.

Programmet søker til enhver tid etter et nytt bilde som er lagt til. Når et nytt bilde er detektert, er det fordi en bil har ankommet. Dette nye bildet blir gråskalert og lagt inn i en mappe som lages automatisk og heter "bil1". Denne nye mappen er provisorisk og brukes kun under bildebehandlingen for å finne registreringsnummeret.

Originalbildet ligger fortsatt i "PSYSImage". Det gråskalerte bildet blir lagt inn i "bil1"-mappen. Når bildebehandlingen gjennomgås steg for steg som forklart tidligere, blir bildet for hvert steg lagret i denne mappen. Det betyr at mappen til slutt inneholder syv bilder hvor det siste blir sendt videre til tegngjenkjenningsprogrammet.

Når denne behandlingen er ferdig, søker programmet på nytt etter nye bilder i mappen "PSYSImage". Når det detekteres nytt bilde, blir "bil1" slettet. Dette gjøres for å fjerne bildene fra behandlingen av den forrige bilen. Ny mappe som også heter "bil1" blir så laget, og den samme behandlingen gjøres på nytt. Grunnen til at de ferdigbehandlede bildene blir slettet, er for å ikke fylle opp lagringsplassen på datamaskin. Disse bildene har man ikke behov for så lenge man har originalbildet.

I Figur 7.21er algoritmen for bildehåndtering i mappen "PSYSImage".



Figur 7.21 Algoritme for bildehåndteringen

Når behandlingene av bildet er utført i "bil1"-mappen, har en syv bilder. Den siste behandlingen som gir ut bilde uten støy og detaljer rundt rammen, skal avleses i tegngjenkjenningsprogrammet. Når den siste behandlingen i mappen "bil1" er utført, søker tegngjenkjenningsprogrammet etter det nyeste bildet. Originalbilder fra behandlede bilder hvor programmet ikke avleser riktig registreringsnummer, kopieres til en egen mappe utenfor "PSYSImage", "ResizedImage". Bildene i denne mappen halveres i størrelse. Dette bildet vises i vinduet som kommer opp på databehandlerens dataskjerm ved svikt i tegngjenkjenningsprogrammet.

7.3.3 Parkeringsdatabase

For database for parkeringsregistrering er det laget fire tabeller. Denne parkeringsdatabasen er kun for å håndtere inn- og utregistreringer, i tillegg til å regne ut totaltid. Det er først når totaltiden er utregnet at denne infoen blir knyttet opp mot brukerdatabasen.

Den viktigste tabellen i parkeringsdatabasen er "Parkering". Dette er fordi tidspunkt for ankomst og avreise for bilene blir fylt inn her. Når en bil ankommer, blir det laget en ny rad for dette registreringsnummeret med info om ankomst. Ved avreise blir info om dette tidspunktet supplert på gjeldende rad. I Figur 7.22 ser en tabellen "Parkering".

Parkering_ID	Parkering_bilskilt	Tid_inn	Dato_inn	Tid_ut	Dato_ut
1	ZZ 11496	12:19:18	16.4.2014	12:27:32	16.4.2014
NULL	NULL	NULL	NULL	NULL	NULL

Figur 7.22 Tabell "Parkering"

Det er laget en tabell for feilregistreringer. Dette er en tabell for å samle feil, slik at det er lett for en databehandler å ha oversikt dersom det blir spørsmål vedrørende mangelfull informasjon om parkering. Feilene det er snakk om er mangelfull registrering inn/ut og svikt i tegngjenkjenningsprogrammet. Tidspunkt når feilen oppdages blir notert. Her er det også en kolonne som må fylles inn for å bekrefte at feilen er registrert av en databehandler. Figur 7.23 viser tabellen for feilregistreringer.

Feilmeldingstxt	Prioritet	Tid	Dato	Bekreftet
OCR KLARTE IKKE LESE FRA BILDE AUTOMATISK	1	19:00:21	2014-05-19	nei
BILEN VAR IKKE UTREGISTRERT FRA TIDLIGERE PARKERING	2	19:01:59	2014-05-19	nei
BILEN VAR IKKE INNREGISTRERT, BLE IKKE UTREGISTRERT	3	19:02:22	2014-05-19	nei

Figur 7.23 Tabell for feilmeldinger

De to siste tabellene er for utregning av total parkeringstid og registreringsnummer for ansatte og andre som ikke skal betale. Utregning av totaltid er lagt i egen tabell. Slik at det senere kan gjøres enklere med utvidelse av en fakturafunksjon. Det er blant annet en kolonne for bekreftelse om regning er betalt.

7.3.4 Utregning totaltid

Når en rad i tabellen for registrering inn/ut er utfylt, regnes totaltiden ut i en egen tabell. Det er laget et Java-prosjekt som henter tid og dato fra tabellen for biler registrert inn og ut. Når en bil er registrert ut, starter utregningen. Differansen oppgis i hele timer. Etter utregning fylles tabellen "Regning" ut. Her overføres primærnøkkel og registreringsnummeret fra parkering inn/ut-tabellen for å ha kontroll på hvilken bil totaltiden tilhører. Det fylles også inn i en kolonne den utregnede tiden og det fylles inn "nei" i kolonne for bekreftelse. Denne kolonnen er for bekreftelse om regning er betalt eller ikke.

Parkering_ID	Parkering_bilskilt	Email	DiffHours	Bekreftet
1	ZZ 11496	magnus@hisf.no	0	nei

Figur 7.24 Tabell "Regning"

I Figur 7.24 ser en tabell "Regning" etter at totaltid er regnet ut.

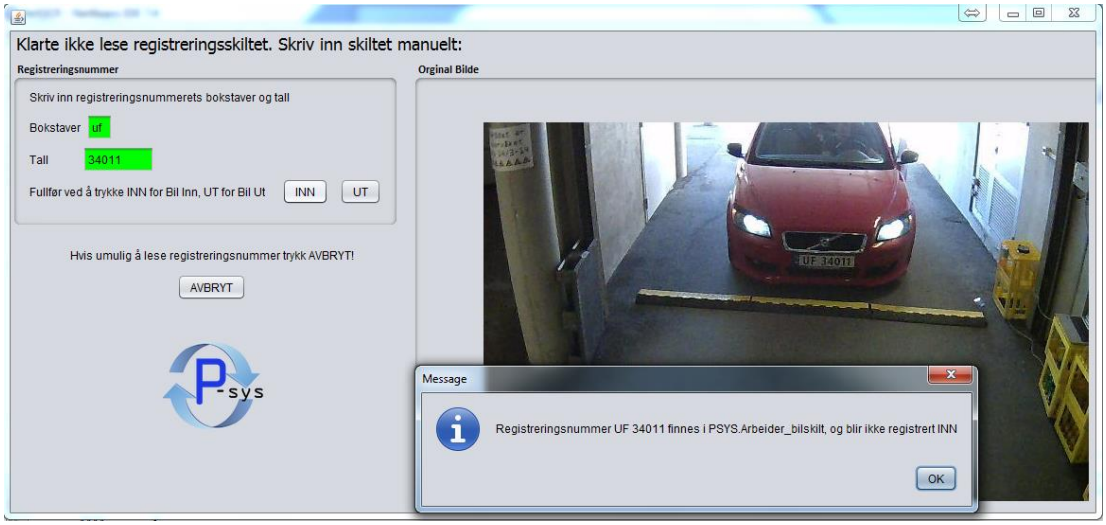
7.3.5 Ansatte

Tabellen "Ansatte" inneholder informasjon om registreringsnummer for personer som ikke skal betale for parkeringstid. Programmet vil ved registrering av ankomst/avreise av bil kontrollere denne databasen. Dersom avlest registreringsnummer ved ankomst/avreise finnes i denne tabellen, vil ingenting skje. Informasjon vil ikke legges til noen steder. I Figur 7.25 ser en et eksempel på tabellen. Registreringsnumre som er lik de i tabellen, vil ikke være gjeldene.

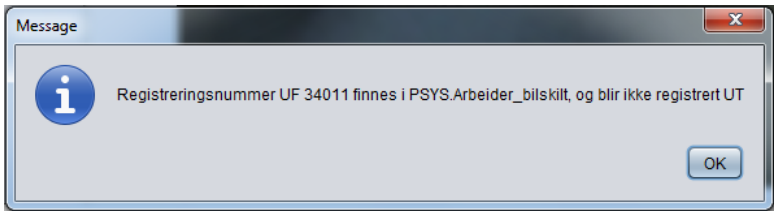
Ansatte_ID	Ansatte_bilskilt
1	AA 12345
2	BB 12345
3	CC 12345
4	DD 12345
5	EE 12345
6	FF 12345
7	GG 12345
8	UF 34011
NULL	NULL

Figur 7.25 Eksempel på tabell "Ansatte"

Dersom tegngjenkjenningsprogrammet ikke har lykkes med å tyde registreringsnummeret, vil det som ellers komme et vindu hvor en databehandler må taste inn informasjonen manuelt. Tastes det inn et registreringsnummer som er registrert i tabellen "Ansatte", vil en få en meldingsboks som i Figur 7.26 og Figur 7.27.



Figur 7.26 Manuell innregistrering av ansattes registreringsnummer



Figur 7.27 Manuell utregistrering av ansattes registreringsnummer

7.3.6 Feilregistreringer

En bil som registreres inn på parkeringsanlegget, må være registrert ut ved forrige besøk. Det er konstruert en metode i Java som kontrollerer dette. Denne metoden finner siste gang registreringsnummeret var registrert inn, og sjekker at det er fylt inn informasjon i kolonnene for dato og tid ut.

	Parkering_ID	Parkering_bilskilt	Tid_inn	Dato_inn	Tid_ut	Dato_ut
	1	ZZ 11496	12:19:18	16.4.2014	NULL	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL

Figur 7.28 Tabell for biler ankommet og reist

Dersom bil "ZZ 11496" ankommer, ser en i Figur 7.28 at den allerede ligger inne i rad 1 men er ikke registrert ut. Metoden finner registreringsnummeret allerede registrert inn. Derfor slettes rad 1 og ny rad opprettes med den nye registreringen. Det vil opprettes en rad i tabellen for feil. Programmet setter inn feilmeldingstekst og tidspunkt for oppdagelse av feilen. Blir en bil registrert ut, men ikke ligger innregistrert, vil det samme skje. Programmet leter etter rad i

tabellen for å fylle inn tidspunkt for avreise, men registreringsnummeret ligger ikke innregistrert. Feilmelding blir fylt inn i tabell som vist i Figur 7.29

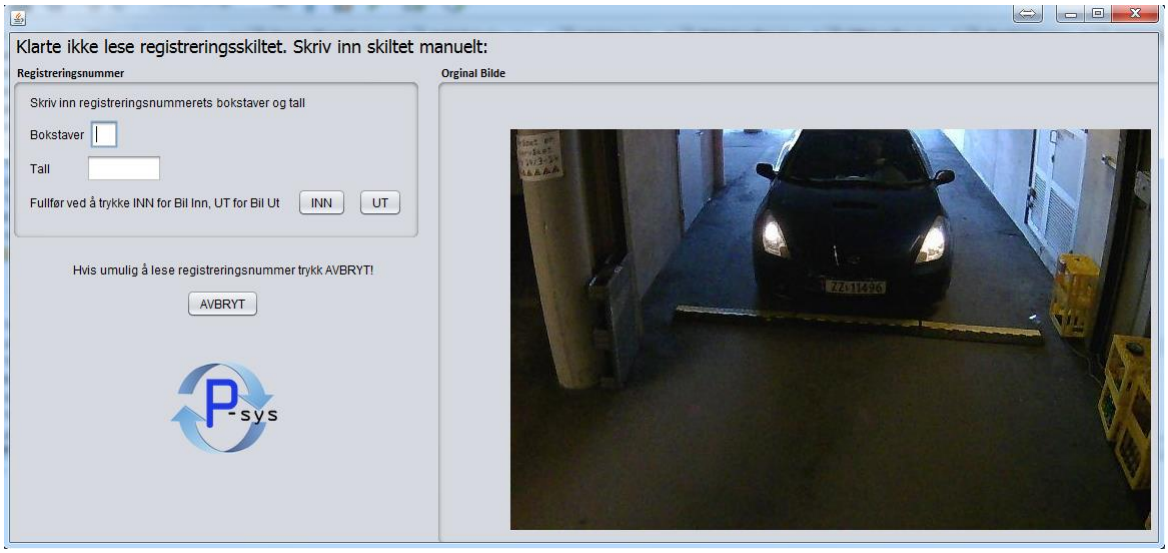
Feilmeldingstxt	Prioritet	Tid	Dato	Bekreftet
BILEN VAR IKKE UTREGISTRERT FRA TIDLIGERE PARKERING	2	19:01:59	2014-05-19	nei
BILEN VAR IKKE INNREGISTRERT, BLE IKKE UTREGISTRERT	3	19:02:22	2014-05-19	nei

Figur 7.29 Feilmelding ved manglende inn- eller utregistrering

Grunnen til at feilregistrerte biler fjernes fra "Parkering" og legges inn i tabellen for feilregistreringer, er for å forhindre flere feil. Hadde en ikke fjernet feilregistrerte biler fra tabellen "Parkering", risikerer en at ny feil kan oppstå og ny registrering knyttes opp mot gammel registrering. Totaltid kan da bli feil, eksempelvis dersom en feilregistrering for et år siden knyttes opp mot en ny feilregistrering. Parkeringstiden blir da et år. Tabellen "Feil" er konstruert for at databehandleren skal kunne følge en logg med feilregistreringer.

7.3.7 Svikt i tegngjenkjenningsprogrammet

Dersom tegngjenkjenningsprogrammet ikke klarer å lese registreringsnummeret, registreres feilen i egen tabell. I tillegg vil det dukke opp et vindu på databehandlerens PC. Dette vinduet er kun for databehandleren. Det inneholder bilde av bilen og et felt hvor en skriver inn registreringsnummeret manuelt.



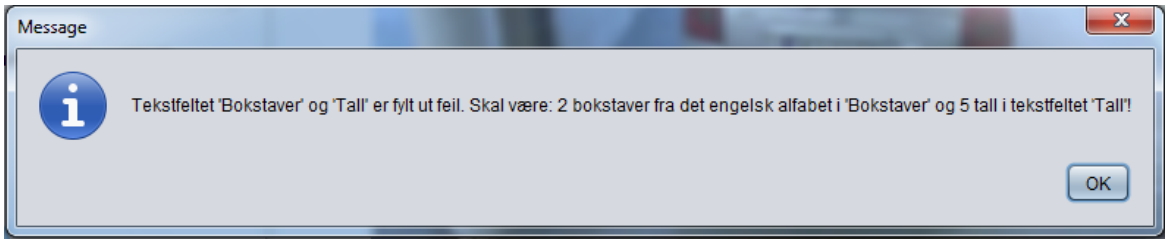
Figur 7.30 Vindu for manuell inntasting

Figur 7.30 viser vinduet som kommer opp. Bildet til høyre er bilen som systemet ikke klarte å avlese. Her må en etter å ha tastet registreringsnummeret også trykke «INN» eller «UT» ettersom hvilken retning bilen kjører. Dersom det ikke er mulig å lese registreringsnummeret, trykker en «AVBRYT».

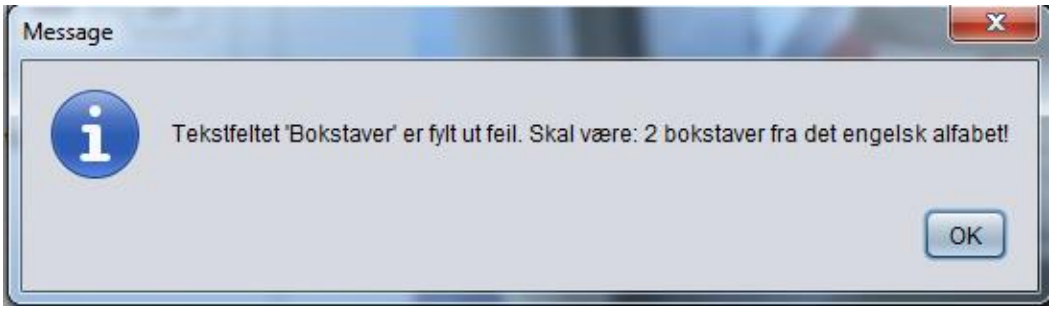
Dersom bildet i vinduet ikke er godt nok, kan en holde musepekeren over bildet. Da får en opp en hjelpelinje med plasseringen til bildet på datamaskinen. Da kan databehandleren finne originalbildet og prøve å tyde registreringsnummeret ut fra dette, ved for eksempel bedre zoom. I Figur 7.31 ser en denne hjelpeteksten.

Figur 7.31 Hjelpetekst til plassering av bilde

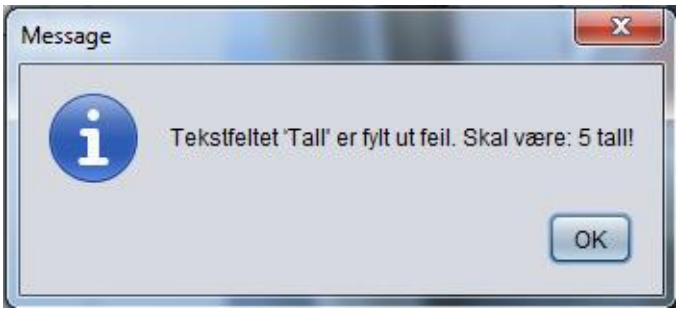
Det er kun mulig å registrere biler dersom en taster inn to bokstaver og fem siffer. Prøver en likevel å registrere uten å oppfylle disse kravene, vil en få opp feilmelding. Denne feilmeldingen er tilpasset det som eventuelt mangler.



Figur 7.32 Feil antall bokstaver og siffer



Figur 7.33 Feil antall bokstaver

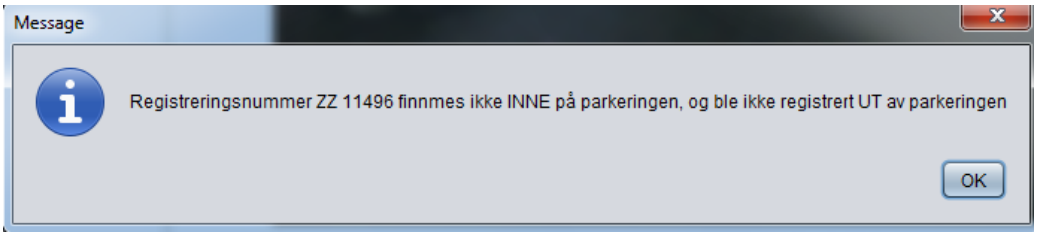


Figur 7.34 Feil antall siffer

Når informasjonen er tastet inn korrekt, vil inntastingsfeltene bli grønne og det kommer opp en melding som bekrefter at informasjonen er lagt til, som i Figur 7.35.

Figur 7.35 Antall bokstaver og siffer stemmer

Dersom det har skjedd en feil i systemet ved innkjøring, vil det komme opp melding om dette, som i Figur 7.36.



Figur 7.36 Informasjon fylt manuelt, men bilen er ikke registrert inn

7.4 Brukerregistrering

Her beskrives hvordan en bruker opprettes. Feilmeldinger blir forklart og brukerdatabasens oppbygging gjennomgås.

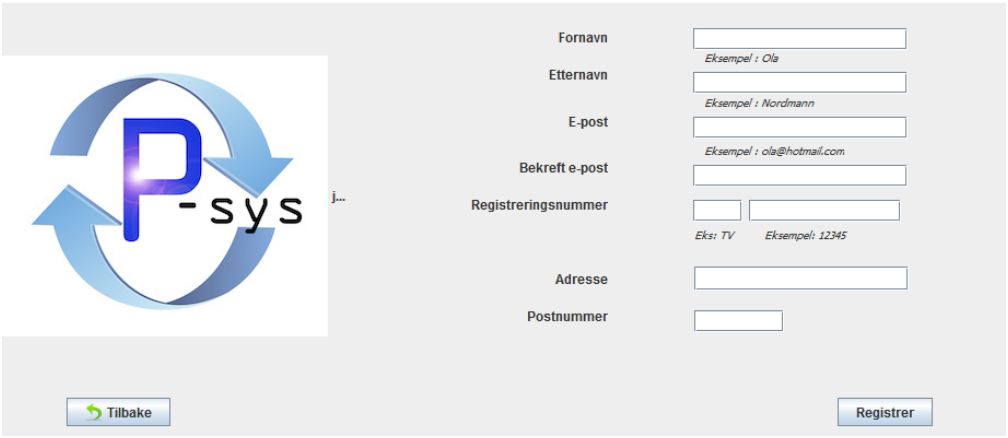
7.4.1 Registrering

Det er nødvendig med et enkelt registreringssystem som gjør det brukervennlig for alle. Det kreves det at brukeren selv registrerer seg ved første besøk.

For å registrere kundene til brukerdatabasen, er det laget et javaprogram med et enkelt brukergrensesnitt. Grensesnittet er utviklet i Java Swing, men informasjonen som skrives inn blir sendt til brukerdatabasen konstruert i MySQL. Det er laget en framside der man velger om det skal registreres en ny kunde eller om det er en databehandler som skal inn i brukerdatabasen for å hente informasjon. Figur 7.37 viser startsiden. På denne siden kan en velge mellom å registrere ny kunde eller logge seg inn som ansatt.

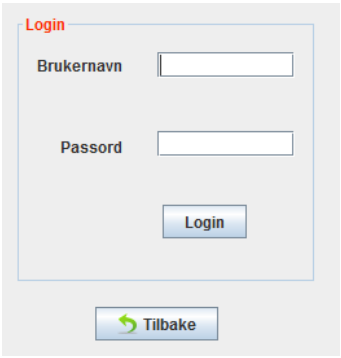
Figur 7.37 Startside brukerregistrering

Velger man «Ny kunde», vil vinduet som i Figur 7.38 vises. Når man trykker på «Registrer» vil data bli lagt inn i brukerdatabasen. Her skal det skal tastes inn informasjon om identitet, registreringsnummer og fakturaadresse. Med denne informasjonen opprettes en bruker. Registreringsnummeret til denne brukeren brukes til sammenligning med resultatet i *7.3 Bilregistrering*. Det er ikke mulig å registrere seg dersom registreringsnummeret allerede finnes i brukerdatabasen. Dermed kan ikke samme bil registreres flere ganger på ulike navn. Dersom bilen har skiftet eier må en databehandler endre brukerinformasjon. Dette gjøres ved å gå inn i MySQL. Der finner man vedkommende i tabellene, og endrer det som er nødvendig.



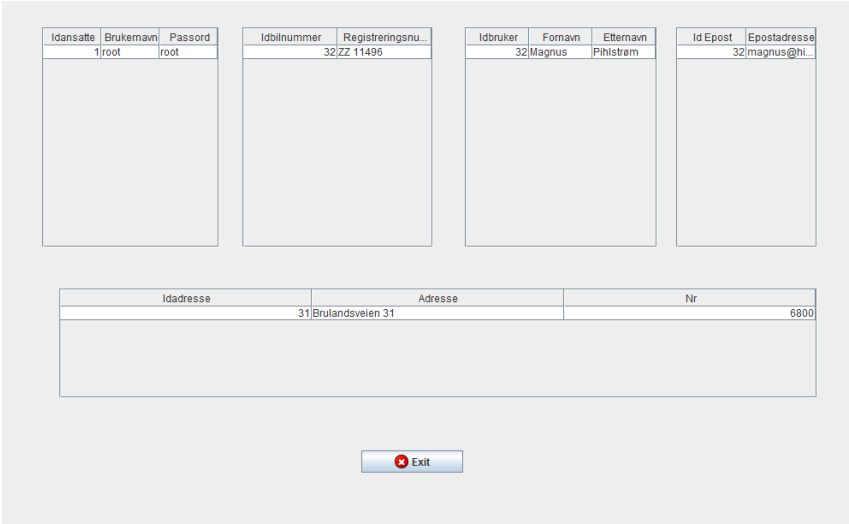
Figur 7.38 Registreringsvindu brukerregistrering

Når en i første vindu (Figur 7.37) trykker på «For ansatte» vil en komme til et vindu som i Figur 7.39. Av sikkerhetsmessige årsaker må den ansatte taste inn brukernavn og passord, for å nå fram til ønsket informasjon. Slik vil man kunne unngå at uvedkommende kan komme seg inn i brukerdatabasen.



Figur 7.39 Innlogging for ansatte

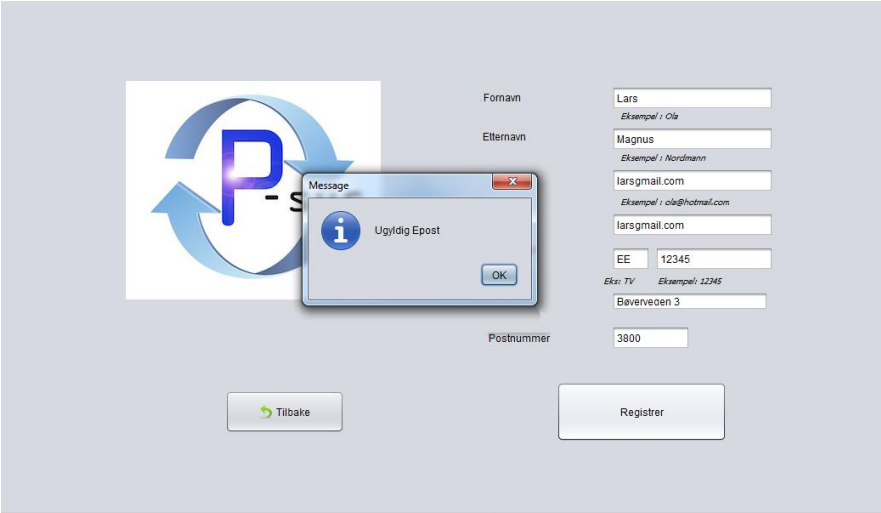
Har man korrekt brukernavn og passord kommer man inn i brukerdatabasens oversikt som ser ut som i Figur 7.40. Her har man oversikt over hvem som er registrert i brukerdatabasen.



Figur 7.40 Oversikt over utvalgte tabeller med virtuell tabell

7.4.2 Feilmeldinger

Dersom en bruker prøver å registrere seg med feil informasjon vil det komme opp en feilmeldingsboks på skjermen. Teksten i denne boksen er tilpasset hva som er feilkilden.



Figur 7.41 Ugyldig e-post

I Figur 7.41 er det prøvd å registrere ugyldig e-post. Dette betyr at et eller begge tekstfeltene er tomme, innholdet i tekstboksene er ulikt eller adressen inneholder ikke "@".

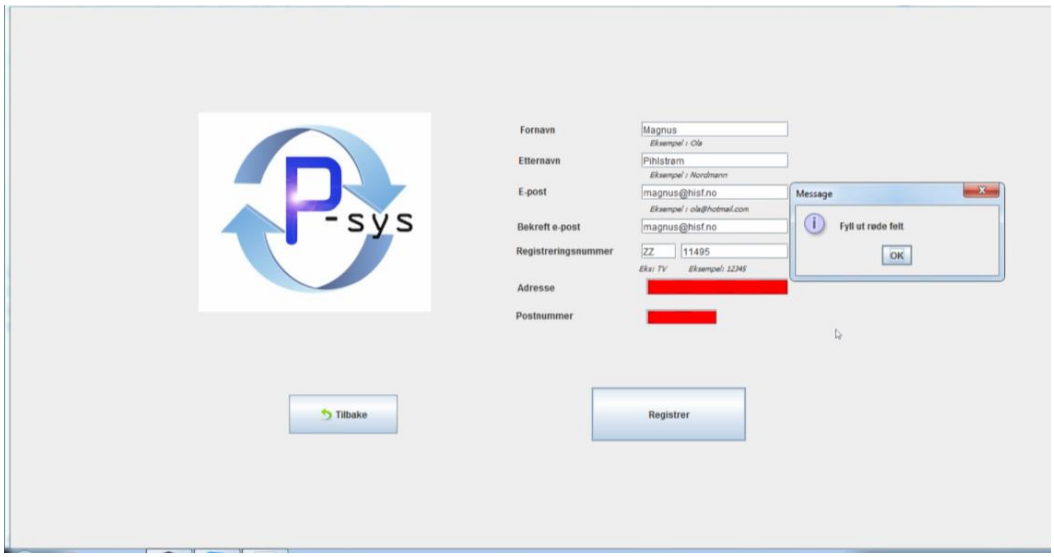
Figur 7.42 Registreringsnummer er allerede i brukerdatabasen

Feilmeldingen i Figur 7.42 kommer opp dersom registreringsnummeret allerede ligger i brukerdatabasen. Skjer dette under førstegangsregistrering, kan årsaken være at bilen nettopp har skiftet eier. Registreringsnummeret er da mest sannsynlig registrert på forrige bileier, og databehandler må inn i databasen for å endre brukerinformasjon.

Figur 7.43 Ugyldig registreringsnummer

Vinduet i Figur 7.43 kommer opp fordi en bruker har forsøkt å registrere et registreringsnummer som ikke inneholder to bokstaver og fem siffer.

Dersom noen felt står tomme når en trykker «Registrer», vil disse feltene merkes med rødt. Det vil også komme en feilmelding som i Figur 7.44



Figur 7.44 Uutfylte felt

Dersom uvedkommende forsøker å logge seg inn i brukerdatabasen eller databehandleren skriver inn feil brukernavn eller passord, kommer en feilmelding som i Figur 7.45

Figur 7.45 Brukernavn eller passord er feil

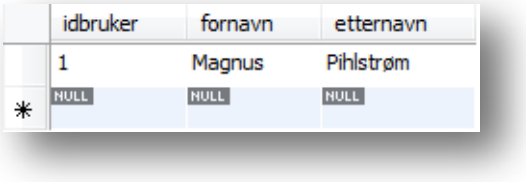
7.4.3 Brukerdatabase

I brukerdatabasen er det laget fem tabeller. Oppdelingen er gjort på denne måten for å gjøre det enklere ved utvidelse av flere funksjoner, eller for en databehandler å rette opp i feil. I alle tabellene er det informasjon som er viktig for å vite hvem som eier hvilken bil og hvordan en kan få sendt faktura. En får oversikt over informasjonen for brukere ved en virtuell tabell, "Registrerte kunder". Denne virtuelle tabellen viser navn, adresse, e-post og registreringsnummer.

Når man i brukerregistreringen trykker «Registrer», vil informasjonen i de forskjellige feltene legges inn i en tilhørende tabell. Fornavn og etternavn i "Bruker", e-post i "Epost", registreringsnummer i "Bilnummer" og adresse og postnummer i "Adresse". Siden alle felt må være fylt ut og informasjonen legges inn samtidig, vil en kunne knytte dette sammen i den virtuelle tabellen ved sammenligning av primærnøkkel i tabellene.

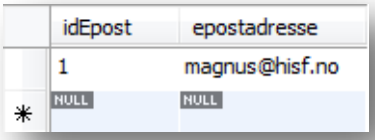
I Figur 7.46 er det en bruker som er ferdig med utfylling av brukerinformasjonen. I Figur 7.47 – 7.50 ser en hvordan informasjonen legges inn i tabellene. I Figur 7.51 er informasjonen fra disse tabellene samlet i den virtuelle tabellen "Registrerte kunder".

Figur 7.46 Bruker har fylt ut informasjon



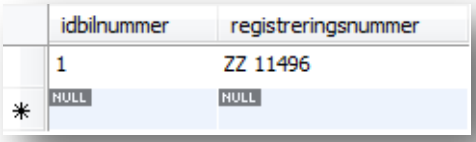
	idbruker	fornavn	etternavn
	1	Magnus	Pihlstrøm
*	NULL	NULL	NULL

Figur 7.47 Tabell "Bruker"



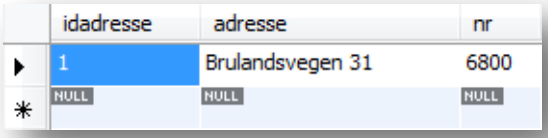
idEpost	epostadresse
1	magnus@hisf.no
NULL	NULL

Figur 7.48 Tabell "Epost"



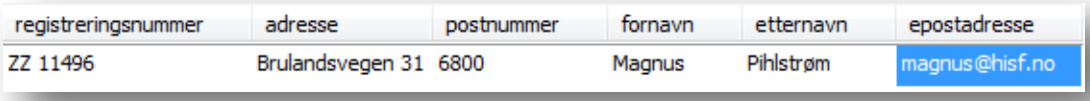
idbilnummer	registreringsnummer
1	ZZ 11496
NULL	NULL

Figur 7.49 Tabell "Bilnummer"



idadresse	adresse	nr
1	Brulandsvegen 31	6800
NULL	NULL	NULL

Figur 7.50 Tabell "Adresse"



registreringsnummer	adresse	postnummer	fornavn	etternavn	epostadresse
ZZ 11496	Brulandsvegen 31	6800	Magnus	Pihlstrøm	magnus@hisf.no

Figur 7.51 Virtuell tabell "Registrerte kunder"

8.0 Utviklingen av prototypen

I dette kapittelet grunngis valg av løsninger. Det gjennomgås hvordan utviklingen av prototypen har foregått. Gruppen forklarer hva som hender ved de forskjellige stegene i prosessen og diskuterer fordeler/ulemper. Det blir også fremlagt forslag til forbedringer og alternative løsninger.

Under utviklingen av prototypen har gruppen latt seg inspirere av eksisterende skiltgjenkjenningsprogram med ulike teknikker. Disse er brukt til å utvikle egne løsninger.

8.1 Bilregistrering

Her drøftes løsningene ved å registrere ankomst og avreise ved bildetaking. Det diskuteres hvordan gruppen har løst problemer vedrørende fart og plassering av bil på bildet.

8.1.1 Fotolinje

Gruppen ville fastsette en plassering av bilene ved bildetaking. Fotolinje ble plassert slik at den ble midt i bildene som skulle bildebehandles. Søkeintervallene i Java-programmet ble tilpasset for å finne registreringsskiltet ved denne plasseringen. Deretter ble det testet ved å flytte en bil i intervaller på 15cm. Bilen ble flyttet mot kamera, deretter bort fra kamera. Dette ble gjort for å finne et fotoområde hvor det er mulig å finne rektangler i de forhåndsbestemte størrelsene, 30 – 50 piksler i høyde og 120 – 180 piksler i bredde. Gruppen fant på denne måten et fotoområde som strakk seg fra 30cm mot eller fra fotolinjen.

Figur 8.1 viser testintervallene. Det ble satt en fotolinje og deretter tegnet merker med 15cm avstand. Når støtfangeren av bilen var plassert på linje med hvert enkelt merke, ble bilde tatt. Deretter testet gruppen bildebehandling for å sjekke ved hvilke merker registreringsskiltet ble funnet.

Figur 8.1 Finne fotoområde

Gruppen fant ut at ved innkjøring ble registreringsskiltet funnet fra fotolinje og 30cm inn i garasjen. Ved utkjøring er området fra fotolinjen og 30cm ut fra garasje.

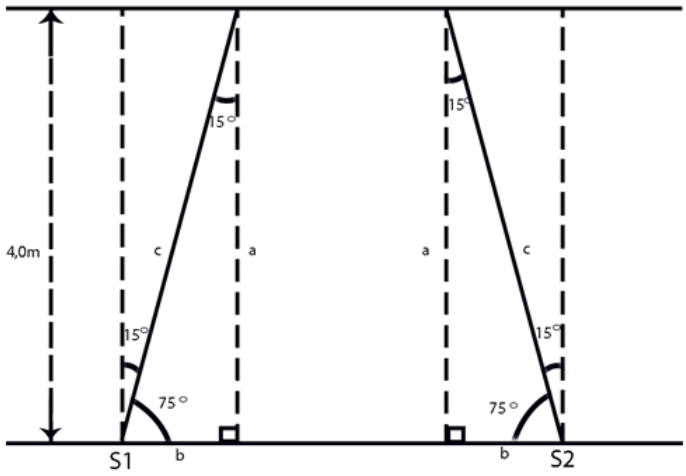
Grunnen til at fotoområdet er forskjellig fra inn- og utkjøring mener gruppen er på grunn av høydeforskjell på registreringsskiltene foran og bak. Dermed må det bakre skiltet være lengre unna for å få samme oppløsning som skiltet i front.

8.1.2 Sensorer

I prosjektet er det brukt to sensorer. Grunnen til det er at det skulle kunne skille bil som kjører inn i fra bil som kjører ut. Dette kunne ikke blitt gjort med bare en sensor. Ved å bruke *Arduino Uno*, ble det enkelte feilregistreringer fordi kortet ikke taklet to sensorer. Det ble derfor skiftet over til *Arduino Mega*.

Ultralydsensorene sender et signal for så å vente på at dette blir reflektert. Sekvensen er anbefalt større enn 50ms. For å forhindre at sensorene skal bruke for lang tid på å sende og motta, og dermed gjøre programmet tregt igjen, ble det også lagt inn 0,10m som nedre grense og 2,00m som øvre grense. Dette betyr at alle objekter innenfor 0,10m eller utenfor 2,00m ikke vil bli registrert.

Dersom sensorene står for nær hverandre, vil det oppstå duplisering. Dette er fordi signalene går ut i en vinkel på 15 grader.I Figur 8.2 er skissen for utregningen av avstand mellom sensorene. Utregningene i Formel 2 viser at avstanden mellom sensorene må være minst 2,2m. Under testingen har det vært en vegg på andre siden av veibanen, 4,00m unna sensorene. Derfor er 4,00m brukt som utgangspunkt i utregningen.



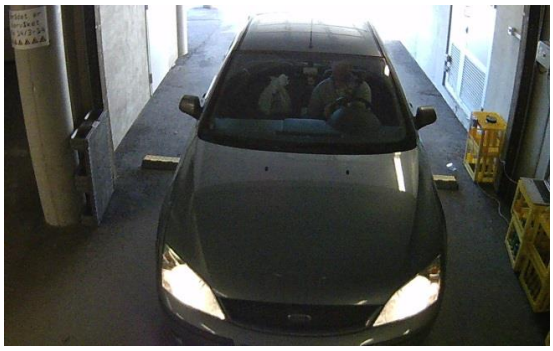
Figur 8.2 Skisse utregning avstand mellom sensorer

$$c = \frac{a}{\sin x} = \frac{4{,}0\ m}{\sin 75°} = 4{,}14m$$

$$b = c \cdot \cos x = 4{,}14m \cdot \cos 75° = 1{,}1m$$

Formel 2 Utregning for avstand mellom sensorer

Når begge sensorene var "lav" altså når ingen biler passerte, gikk de første Arduino-programmene som ble testet tregt. Konsekvensen med dette kan være at en sensor aktiveres for sent, da risikerer en at bilen er utenfor fotolinjen når bildet blir tatt. Dette ser en et eksempel på i Figur Det var først ved innføringen av biblioteket Newping [1] at det ble bedre flyt i målingene. Newping behandler signalene mer effektivt enn de første bibliotekene det ble gjort forsøk med. I følge testing fungerer Newping like raskt med en sensor som med to.



Figur 8.3 Bilde av bil utenfor fotolinje 1

Programmet er laget slik at når begge sensorene blir «høy» innenfor fem sekunders mellomrom, vil kamera ta bilde. Dette fungerer bra når bilene kjører inn, siden registreringsskiltet vil være på samme sted hver gang. Ved avreise er ikke dette en optimal løsning. Bilenes bakre skilt vil være på forskjellig sted alt etter hvor lang hver enkelt bil er. Det er testet med å la systemet ta bildet i det den innerste sensoren slipper, altså melder "lav" fordi bilens bakre del passerer, mens den ytterste sensoren er "høy". Dette har ikke fungert fordi sensorene melder "lav" innimellom, selv om de skulle vært "høy", og dermed blir det tatt bilde. Da er plasseringen av bilens bakside ofte i en posisjon hvor bildebehandling eller avlesing ikke er mulig å gjennomføre.

Figur 8.4 Bilde av bil utenfor fotolinje 2

Figur 8.4 viser bilde tatt ved at den ytterste er "høy" og den innerste skiftes fra "høy" til "lav". Her har sensoren av ukjent grunn meldt "lav" selv om den skulle meldt "høy" og bildet blir tatt på feil tidspunkt.

Det har også vært problemer ved lave temperaturer. I en periode når temperaturen lå rundt 0^{o}C, måtte systemet startes på nytt for å rette opp feilregistreringer som forekom hyppig.

Gruppen mener at i denne prototypen hadde det vært bedre med et annet sensorsystem. Det foreslås å erstatte Arduino med Raspberry Pi på grunn av høyere ytelse og lav pris [10]. Det anbefales å bruke sensorer som har mulighet til å måle bilenes fart.

8.1.3 Kamera

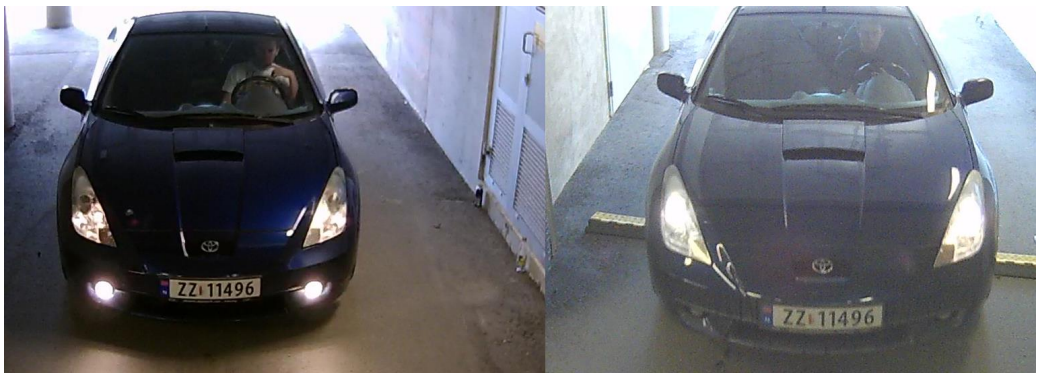
Det ble forsøkt med å stille på innstillingene til kameraet for å bedre bildekvalitet og rette opp i plassering på bilene i bildene som ble tatt. Etter hvert som innstillingene på kameraet ble justert, ble endringene i bildekvalitet og bilplassering kontrollert.

Følgende innstillinger ble justert:

- Detaljnivå
- Endre fra farge til svart/hvitt
- Lukkertid

Ved å endre på detaljnivå og/eller endre farge til sort/hvitt, var det ingen eller liten effekt. Både i bilplassering og bildekvalitet.

Endring av lukkertid viste seg å gjøre en stor forskjell. Bildet ble tatt raskere, men kvaliteten på bildet sank. I Figur 8.5 vises forskjell på bildekvalitet med endring i lukkertid. Bildet til høyre, med lavest lukkertid, har mye mer støy i bildet.



Figur 8.5 Forskjell i bildekvalitet ved endring av lukkertid

Det negative med å ha støy i bildet, er at det blir veldig synlig når en skal finne konturene i Adaptive Threshold. Dette gjør arbeidet med å finne registreringsskiltet mye vanskeligere. Årsaken er at støyen vil danne mønstre som programmet vil oppfatte som et registreringsskilt. I tillegg vil selve skiltplaten inneholde støy rundt tegnene som skal leses av. En risikerer da å fjerne deler av tegnene med metoden Median Filter.

Gruppen kom frem til at for å få best mulig bildebehandling, trengs lang lukkertid. Problemet da er at bilene er feilplassert på bildene. For å få riktig plassering på bilene, men uten at bildekvaliteten forringes, er en avhengig av at bilene hoder lav fart.

8.1.4 Fartsregulering

For å kunne øke lukkertiden på kameraet, men samtidig ha riktig plassering av bilene på bildene, er lav fart en avgjørende faktor. Det ble gjort forsøk med en provisorisk fartsdump. Denne hadde liten effekt.

Det ble satt opp en lysindikasjon ved innkjøringen til garasjen. Denne ble laget ved hjelp av to lyskilder, en i rød og en i blå. Når en bil bryter en sensor, lyser det røde lyset. Da må farten senkes til 10km/t til neste sensor brytes og det blå lyset tennes. Under testing har gruppen erfart at med en fart på 10km/t vil plasseringen på bilene bli innenfor fotoområdet.

Figur 8.6 Bil følger lysindikasjon

I Figur 8.6 ser man et eksempel hvor hastigheten blir fulgt ved lysindikasjon. Kvaliteten på bildet er optimalt og bilen er innenfor et området hvor programmet klarer å lese av registreringsnummeret.

Lysindikasjonen er bygget av besøkende fra en lokal ungdomsskole. Den er limt sammen av hvite hylleplater. I fronten er det boret to hull for montering av lyskilder, en rød og en blå. Kassen er klargjort for montering på stativ ved å borre hull i bunnen. Under testing i garasjen ble lyskildene forsynt med en provisorisk strømforsyning. Figur 8.7 viser lysindikasjonen.

Figur 8.7 Lysindikasjon

Når lysindikasjonen følges, er resultatene av bildebehandlingen bedre. Problemer med lysindikasjonen er å få brukerne til å følge oppfordringen med å senke farten ved rødt lys.

8.2 Brukerregistrering

Her er det skrevet om en registreringsstasjon som ble bygget i prosjektperioden. Det er også utført en test av brukerregistreringsprosessen for å finne detaljer prosjektgruppen kan ha oversett underveis.

8.2.1 Registreringsstasjon

Det ble i prosjektperioden bygget en registreringstasjon for bruk under testperioden i garasjen. Denne er ikke brukt like mye som ønsket, grunnet en ikke fikk gjennomført en fullverdig test i høgskolens garasje.

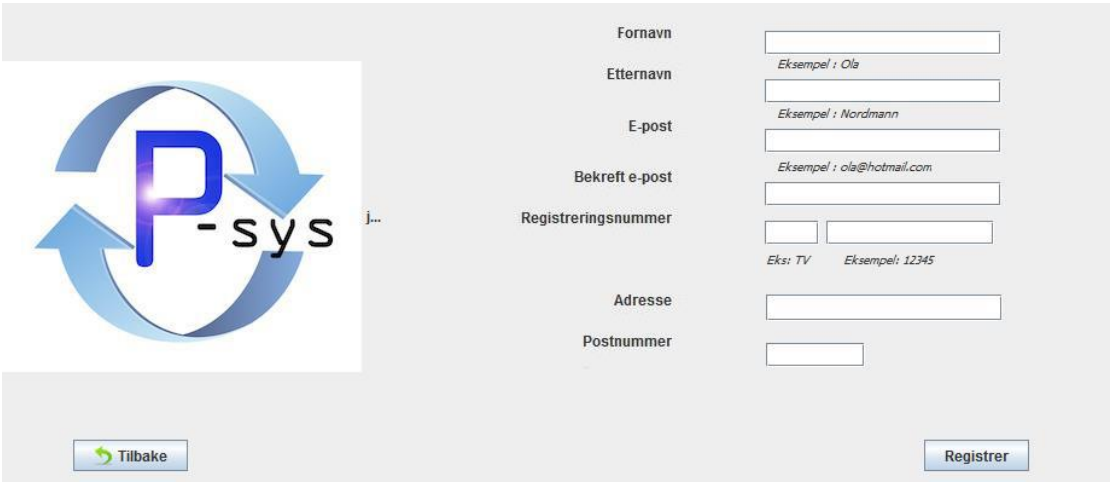
Rammeverket til stasjonen er bygget med lekter satt sammen av vinkelbeslag. Platingen er utført med hvite hylleplater som er limt på rammeverket. Det er montert hjul på stasjonen for mobilitet. På innsiden er en hylle hvor en PC kan plasseres, denne kobles mot skjerm, datamus og tastatur som er integrert i registreringstasjonen. Stasjonens mål er 79x42x146cm. I Figur 8.8 ser en registreringsstasjonen som gruppen har bygget

Figur 8.8 Brukeregistreringsstasjon

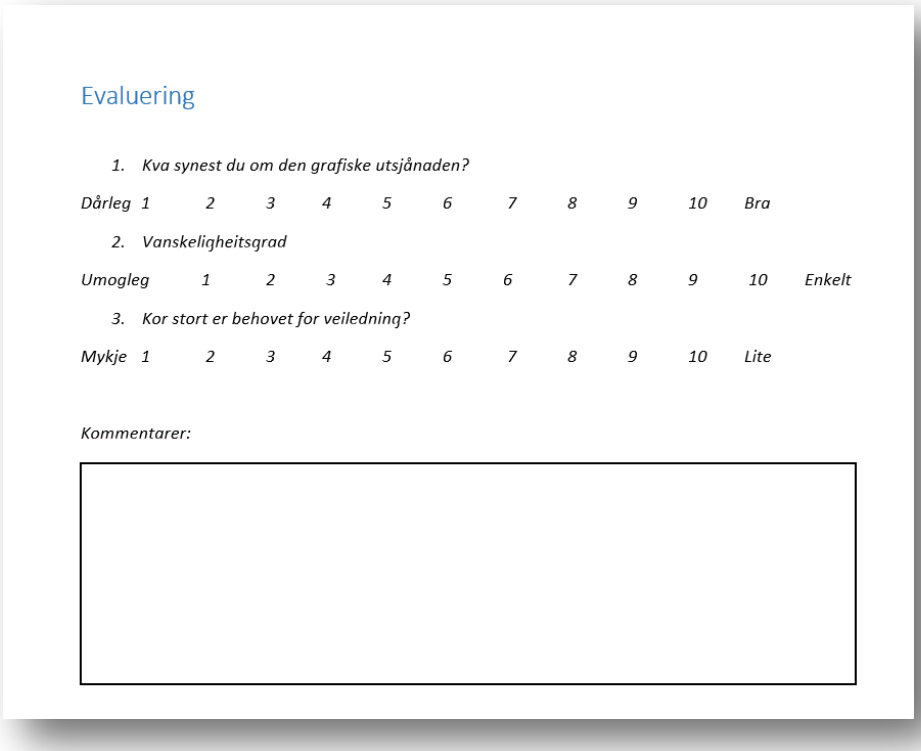
Modellen er brukt under demofilm for systemet og under testing av brukergrensesnittet.

8.2.2 Brukergrensesnitt

Brukergrensesnittet er laget med Java Swing. Brukergrensesnittet ble testet av 20 personer i aldersgruppen 18 – 42 år. Disse fikk i oppgave å registrere en bruker i brukerdatabasen. Det ble hengt opp instruksjoner på registreringsstasjonen, se Vedlegg 7. Disse instruksjonene skulle være tilstrekkelig for å kunne registrere en ny bruker på egenhånd. Deretter skulle de fylle ut et skjema med evaluering av design, vanskelighetsgrad og behov for veiledning. Skalaen var gradert med 1 som dårligst og 10 som best. Om ønskelig kunne skjemaet utdypes med kommentarer. Figur 8.9 viser grensesnittet da undersøkelsen ble foretatt og Figur 8.10 viser evalueringsskjemaet.



Figur 8.9 Brukergrensesnitt ved undersøkelse



Evaluering

1. *Kva synest du om den grafiske utsjånaden?*

Dårleg 1 2 3 4 5 6 7 8 9 10 Bra

2. *Vanskeligheitsgrad*

Umogleg 1 2 3 4 5 6 7 8 9 10 Enkelt

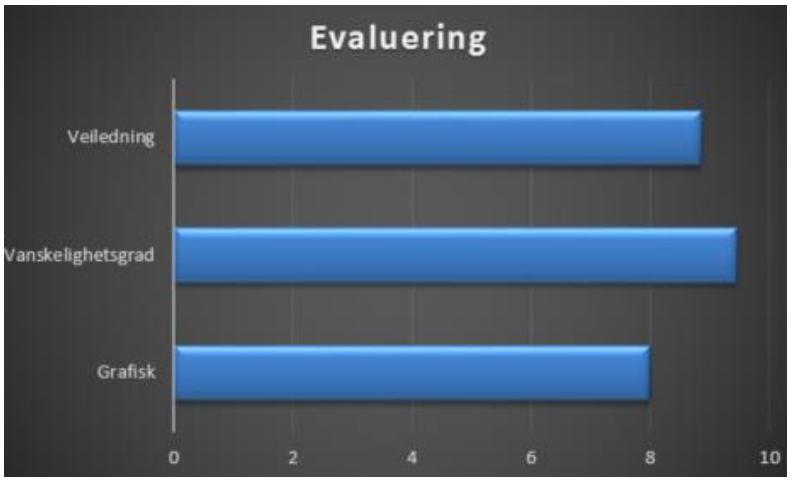
3. *Kor stort er behovet for veiledning?*

Mykje 1 2 3 4 5 6 7 8 9 10 Lite

Kommentarer:

Figur 8.10 Evalueringsskjema for undersøkelse

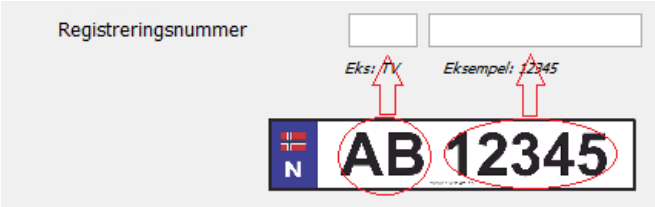
Tilbakemeldingene fra undersøkelsen var positive. Vurderingene varierte fra 5 til 10. Figur 8.11 viser en fremstilling av gjennomsnittet fra vurderingene.



Figur 8.11 Fremstilling av vurdering

På bakgrunn av tilbakemeldingene fra evalueringen, var det forbedringspotensiale på det grafiske. Informasjonen var feil plassert i forhold til boksene som skulle fylles ut. Det ble også uttrykt ønsker om bedre forklaring på hva som skulle fylles inn, noe som trakk ned på poengsummen for behov for veiledning. Det ble avdekket feil ved utfylling av e-postadresser. Adresser som inneholdt spesialtegn som punktum, understrek og bindestrek foran alfakrøllen fikk feilmelding. En adresse som inneholdt to punktum etter alfakrøllen ble heller ikke registrert.

Forbedringer som ble gjort etter testen var å få informasjonen på linje med feltene og gjøre det mulig å bruke spesialtegn foran alfakrøll med flere etterfølgende punktum i e-postadressen. Det er ønsket utbedring med å legge ved et eksempelbilde under felt for registreringsnummer, som i Figur 8.12, for bedre beskrivelse.



Figur 8.12 Forslag for forbedring av brukergrensesnitt

Gruppen er fornøyd med tilbakemeldingene og viser at registrering er enkel å gjennomføre. Det grafiske må forbedres noe. Det kunne vært utvidet med flere feilmeldinger tilpasset feilene som forekommer under brukerregistrering.

8.3 Bildebehandling

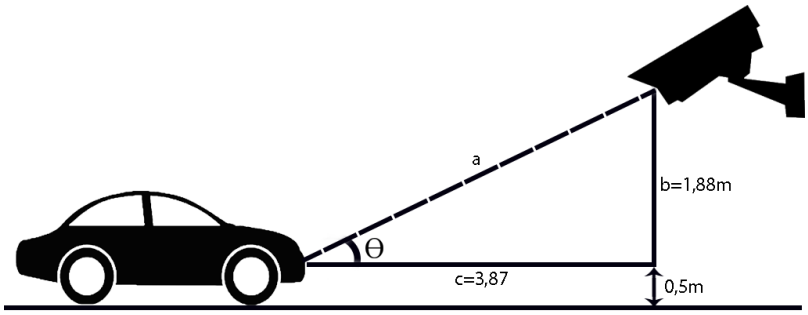
Her dokumenteres hvordan gruppen er kommet frem løsningen av bildebehandlingene i *7.1 Finne registreringsskiltet* og *7.2 Behandle registreringsskiltet*. Det fremvises først hva som har vært et problem under testing. Så diskuteres konsekvenser dette gir, forslag til alternative løsninger.

8.3.1 Test

Ved testing av systemet i garasjen, ble det tidlig mistenkt at lysforholdene ikke var optimale. Det kunne også være vinkelen mellom kamera og registreringsskilt, men siden bildene som ble sendt til tegngjenkjenning var med mye støy, ble bildekvaliteten prioritert testet.

Skulle det vise seg å fortsatt være et problem med bedre forhold, var en av de tidlige løsningene for å klargjøre registreringsnumrene for avlesing av tegngjenkjenningsprogrammet å "vri" skiltrammen (Eng; "skew"). Slik blir bildet som om det var tatt rett forfra istedenfor på skrå.

Å bruke blitz for bedre bildekvalitet i garasjen er ikke et alternativt. Dette kan blende eller skremme sjåføren er det derfor ikke noen bilder tatt med blitz, verken inne eller ute. Det ble forsøkt med IR-blitz, men uten forbedring. Alle bildene i testen er tatt i en vinkel på 25,84^{o} på registreringsskiltene. I Figur 8.13 ser en skisse for utregning av vinkelen i garasjen. Kamera er montert 3,87m unna fotolinjen og 2,87m over bakkenivå. Det er tatt utgangspunkt i at registreringsskiltene er 0,50m over bakken. Dette gir en høydeforskjell mellom kamera og registreringsskilt på 1,88m. Ved å ta bildene ute med kompaktkamera 2,00m unna bilen og 1,47m over bakken, får en samme vinkel. I Formel 3 ser en utregningen av vinkelen.



Figur 8.13 Skisse for utregning av vinkel mellom kamera og registreringsskilt

$$a_1 = \sqrt{3{,}87^2 + 1{,}88^2}\text{m} = 4{,}30$$

$$\theta_1 = \cos^{-1}\theta = \cos^{-1}\frac{3{,}87}{4{,}30} = 25{,}84^{o}$$

Formel 3 Utregning vinkel mellom kamera og registreringsskilt

Gruppen bestemte seg for å forsøke med å bildebehandle og avlese bilder av ti biler tatt med IP-kamera i garasjen, for så å sammenligne resultatet av ti bilder tatt ute med kompaktkamera. Bildene tatt med kompaktkamera har en oppløsning på 4288x3216 piksler, og bildene fra IP-kamera er på 1280x800 piksler. Det er ikke gjennomført forsøk med kompaktkamera i garasjen. Dette på grunn av at det ble for dårlig kvalitet på bildene uten blitz.

Av de ti bildene som ble tatt med IP-kamera i garasjen, ble alle bildebehandlet riktig. Derimot ble ingen av disse avlest riktig av tegngjenkjenningsprogrammet. Syv registreringsskilt ga ingen resultat etter avlesing. På de tre en fikk resultat, var resultatene feil. Hele denne behandlingen av bildene tok mellom fem til ti sekunder. Bildene som ble sendt til tegngjenkjenning, har for lav oppløsning til at forskjellen på hvert tegn blir synlig. Gruppen mener at tegnene består av så få piksler at tegnene kan bli oppfattet som støy. Figur 8.14 viser et bilde som ble sendt til avlesing, men ga ingen resultat.

Figur 8.14 Bilde med oppløsning 165x26

Av de ti bildene som ble tatt med kompaktkamera, ga alle resultat av tegngjenkjenningen. Syv registreringsskilt ga riktig resultat. På grunn av høyere oppløsning på bildene ble flere lest av riktig. Oppløsningen førte også til at bildebehandlingen tok lengre tid. Behandlingene tok mellom 35 til 50 sekunder. I Figur 8.15 ser en et ferdigbehandlet registreringsskilt som ga riktig registreringsnummer etter tegngjenkjenningen.

Figur 8.15 Bilde med oppløsning 594x117

Konsekvensen med for lang behandlingstid er dersom flere biler passerer mens bildebehandling av forrige bil pågår. Programmet henter siste bilde i mappen "PSYSImage" for bildebehandling. Hele behandlingen må gjennomføres, med tegngjenkjenning, før programmet igjen leter etter siste lagrede bilde. Dersom det skulle ankomme/reise flere biler mens en slik behandling foregår, er det kun siste bil som vil bildebehandles.

Gruppen foreslår som en løsning på dette å konstruere programmet til å ikke hente siste lagrede bilde i "PSYSImage" til bildebehandling. Istedenfor burde første bilde som er forskjellig fra siste behandlede bilde sendes til bildebehandling. En vil dermed lage et køsystem for bildebehandlingen.

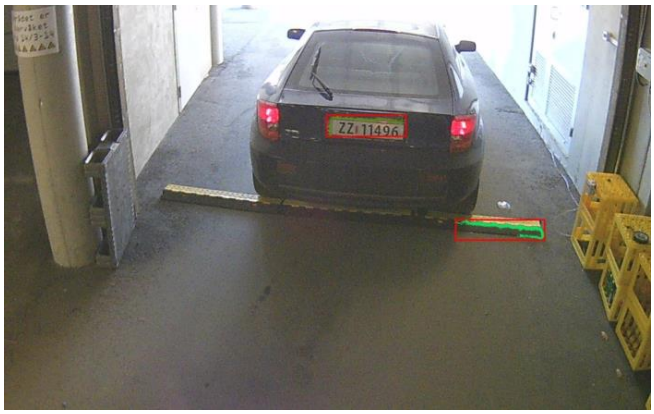
Det ble under testingen bekreftet at med bedre lysforhold og bildekvalitet, var resultatene gode. Men bildebehandlingen tar lengre tid. Gruppen mener at denne testen viser at vinkelen som er brukt ikke er avgjørende for tegngjenkjenningen av registreringsnummeret.

8.3.2 Finne registreringsskilt

Et annet problem med feil lysforhold og dårlig bildekvalitet, var mengden av støy i bildet. Det ble dermed vanskelig for programmet å detektere kun registreringsskiltet og ingen andre rektangler. Dette mener gruppen er årsak til resultatet fra testing over. I garasjen ble det brukt et kamera som gir bilder med 1280x800 piksler. Dette har gruppen observert gir dårlige resultater når en skal gjenkjenne rektangel. Under testing ved bruk av kompaktkamera, med 4288x3216 piksler, har denne behandlingen gått bedre. I Figur 8.16 ser en forskjellen mellom bildekvaliteten med IP – kamera (venstre) og den ønskede kvaliteten (høyre).

Figur 8.16 Forskjell mellom bildekvalitet

Et annet problem som ble oppdaget, var avvik mellom ønsket og faktisk plassering av bil på bildet. Det ble satt en fotolinje tidligere i kapittelet. Siden sensorene og Arduino ikke har vært stabile nok, har det vært nødvendig med et stort søkeintervall når en skal finne registreringsskiltet. Det søkes etter rektangler som er 30 – 50 piksler i høyde og 120 – 180 piksler i bredde. Hadde bildene blitt tatt på samme sted hver gang kunne søkeintervallet vært mindre. En kunne da funnet kun registreringsskiltet og ikke andre rektangler i bildet. En ville heller ikke fått med så mye detaljer rundt skiltet. Figur 8.17 viser at programmet har funnet registreringsskiltet, men i tillegg er det funnet et annet rektangel på grunn av intervallet.



Figur 8.17 To rektangler funnet

8.3.3 Fargebehandling av registreringsskilt

Når bildet skal gjøres sort/hvitt, er det RGB – verdiene som avgjør hvilken farge pikslene ender opp med, *7.2.1 Median Filter*. Med dårlig lys og kontrast vil pikslene i bildet før bildebehandlingen være av en lav RGB – verdi. Dette resulterer i at områder som skulle vært hvitt, blir sort.

I Figur 8.18 ser en resultat av bildebehandling av et bilde med for dårlig lys og kontrast. Figur 8.19 er samme behandlingen, men av et bilde tatt under andre forhold med bedre bildekvalitet. På disse Figurene er terskelgrensen satt til 240. Med bildet tatt under bedre forhold med bedre bildekvalitet, Figur 8.19, blir resultatet av behandlingen riktig. I Figur 8.18 er de punktene som skal være hvite på bildet i en slik RGB - verdi at alle punktene blir sorte etter behandlingen. En løsning på dette var å endre terskelgrensene til 140.

Figur 8.18 Behandling med terskelgrense 240 på bilde med dårlig kvalitet

Figur 8.19 Behandling med terskelgrense 240 på bilde med god kvalitet

Resultatet med å senke grensen, er at områder utenfor skiltplaten kan oppfattes hvite. Dette kan vanskeliggjøre den videre prosessen. I Figur 8.20 er terskelverdien senket til 140. Derfor er det områder rundt skiltrammen som er blitt hvite.

Figur 8.20 Behandling med terskelgrense 140 på bilde med dårlig kvalitet

8.3.4 Median Filter

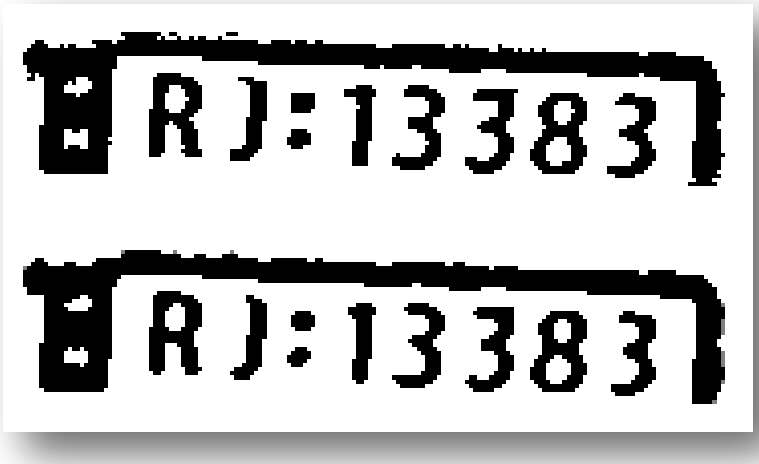
Behandlingen med Median Filter er nødvendig for å kunne skille ut skiltplate og sende bilde uten støy videre til tegngjenkjenning. Støy kan bli gjenkjent som deler av eller egne tegn. Dette gir unødvendige feilavlesinger. Median Filter fjerner støy i bildet, og gjør informasjonen tydeligere og enklere å hente ut senere i programmet

Under behandling med Median Filter, er malen som brukes viktig. Under utviklingen av programmet har en oppdaget at større mal fjerner mer støy. Oppløsningen på bildet bestemmer størrelsen på malen som kan brukes. Under bildebehandling av bilder tatt med IP-kamera, er det brukt 3x3-mal. Bilder tatt med kompaktkamera ute er behandlet med en 10x10-mal.

Under behandling med for stor mal, kan tegnene avrundes. Dette kan gi feilavlesninger i tegngjenkjenningsprogrammet. Bruker en for liten mal, har behandlingen for liten effekt til at støy blir fjernet.

Figur 8.21 Registreringsnummer før og etter Median Filter - kompaktkamera

I Figur 8.21 ser en registreringsnummer før og etter behandling med Median Filter. En ser at støy er fjernet fra bildet. Når tegngjenkjenningen ble utført, viste det seg at malen var for stor. Registreringsnummeret som ble avlest var"22 11490". Dette fordi de to bokstavene er for mye avrundet. Behandlingen fjerner også for mye støy, slik at sekstallet blir "0".



Figur 8.22 Registreringsnummer før og etter Median Filter – IP-kamera

Figur 8.22 viser behandlingen før og etter Median Filter – behandling av et bilde tatt med IP-kameraet. En ser at effekten av denne behandlingen er mindre enn på bildet fra kompaktkamera i Figur 8.21.

For et godt resultat er en avhengig av høy oppløsning av bildet med en tilpasset mal.

8.3.5 Fjerne skiltrammen

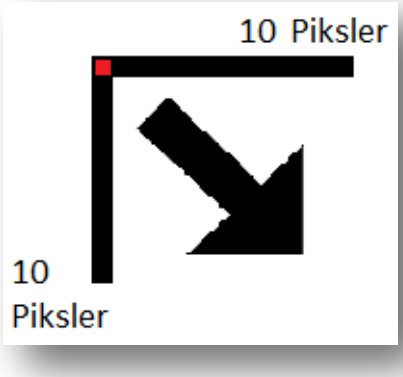
Gruppen har gjort forsøk med to andre metoder for å fjerne skiltrammen enn den som ble valgt. Den ene er å skille ut hvert tegn for seg og den andre er å detektere hjørnene på skiltplaten.

En løsning gruppen har gjort forsøk med, har vært å skille ut hvert tegn i registreringsnummeret. Ved å prøve ut denne teknikken, fant programmet flere rektangler enn det var tegn. Støy i bildet, som fra oblatet eller rammen, ble ofte markert som egne rektangler. Å sette sammen rektanglene med bokstavene og sifferne til ett bilde, var et problem. Dette fordi rektanglenes størrelse varierte og fordi rekkefølgen ikke stemte. I Figur 8.23 ser en markering rundt gjenkjente tegn ved denne teknikken.

Figur 8.23 Hvert enkelt tegn detektert for seg

Prosjektgruppen avgjorde etter noen prøver å finne hele skiltplaten. Ved å finne hele platen er en sikker på at rekkefølgen på tegnene i registreringsnumrene alltid vil være riktig.

Gruppen forsøkte å detektere hjørnene på skiltplaten. Dette gjøres ved en 10x10-mal som teller hvite punkter på et område. Malen beveger seg på skrått på bildet. Når malen teller kun hvite punkter ved to tellinger på rad, blir dette krysningspunktet satt som en koordinat. Dette gjøres til en har fire koordinater, en i hvert hjørne. Disse koordinatene var sjelden på linje med hverandre, noe som er et krav for at programmet skal "klippe" ut mellom punktene. I Figur 8.24 ser en eksempel på en slik mal. Malen har kun talt hvite punkter, beveget seg i pilens retning og deretter talt ti nye hvite punkter. Dermed blir punktet i rødt merket som en koordinat.



Figur 8.24 Mal for hjørnedeteksjon

Gruppen endte opp med løsningen som i *7.2.2 Fjerne skiltrammen*. Da finner en hele linjer som grense istedenfor enkeltpunkt.

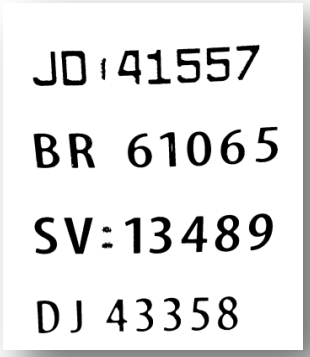
8.4 Tegngjenkjenning

Gruppen forklarer her hvordan tegngjenkjenningsprogrammet har blitt opplært. Det er også gjennomført en test for å sjekke om denne opplæringen har hatt en effekt.

8.4.1 Opplæring tegngjenkjenningsprogram

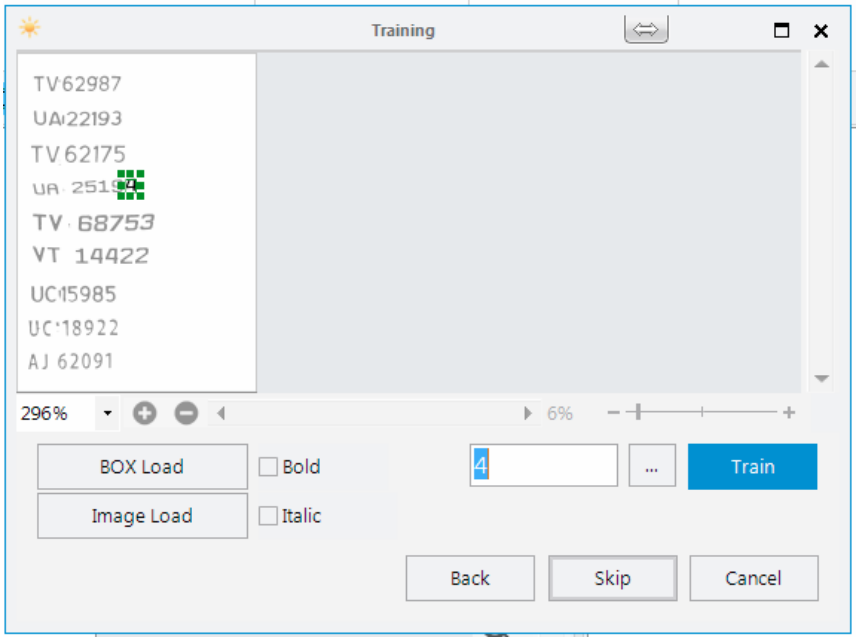
For å unngå feilregistrering, har opplæring av tegngjenkjenningsprogrammet vært viktig. Dette grunnet at registreringsnumrene har en annen skrifttype enn det som er i språkpakkene som følger med tegngjenkjenningsprogrammet, *6.3 Tegngjenkjenning*. Uten slik opplæring opplæringen risikerer en at tegn som "6" kan bli lest som "b", "7" som "T" etc.

For å starte opplæringen må en importere inn en bildefil med ønsket skrift. Dette bildet har blitt forhåndsbehandlet slik at det skal ligne mest mulig på skiltplatene som kommer ut av bildebehandlingen i *7.2.2 Fjerne skiltrammen*. Måten dette er løst på er ved å ta en rekke med bilder av registreringsskilt og endre terskelverdien i *Photoshop*. For å unngå å importere unødvendig mange bildefiler, plasseres flere utklipp på ett bilde, slik som i Figur 8.25.



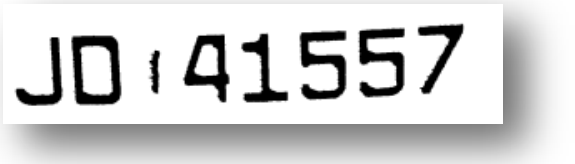
Figur 8.25 Bilde til opplæring

I SunnyPage markeres ønsket tekst og opplæring kan starte. Tegn som er lest vil bli markert med en grønn maske, og vises i tekstrute. Her får en mulighet til å rette tegnet om dette ikke skulle stemme, og justere masken dersom det er nødvendig. Når lest tegn samsvarer med tegnet i tekstboksen trykker man «train» for å lære dette.



Figur 8.26 Opplæring via SunnyPage

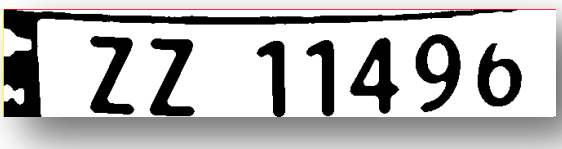
I Figur 8.26 ser en programmet under opplæring. I denne figuren er det tallet fire som skal læres. I tekstboksen ser en at programmet har markert denne som et firetall. Hadde dette vært feil kan det rettes til riktig tegn. Når tegnet er riktig, trykker en «train» og går videre til neste.



Figur 8.27 Registreringsnummer med oblatmerke

En ser i Figur 8.27 et merke mellom bokstavene og sifferne. Dette er merket etter oblatet. Det er gjort forsøk med å lære programmet til å overse dette merket i avlesingen. Dette har gruppen ikke lykkes med. Derfor ble det i bildebehandlingen viktig å fjerne alle merker som ikke skal tydes.

I Figur 8.28 er et registreringsnummer som ble avlest feil av tegngjenkjenningsprogrammet. Avlest nummer ble "22 11490". De to bokstavene som skulle vært "ZZ" har blitt for avrundet under behandling med Median Filter. Dermed ligner bokstavene mer på to-tall enn "Z" i språkpakken. Med mer opplæring av tegngjenkjenningsprogrammet kunne programmet oversett avrundingen og dermed avlest "ZZ" istedenfor "22".



Figur 8.28 Bilde som ga avlesing "22 11490"

8.4.2 Test tegngjenkjenningsprogram

Gruppen har utført en test av tegngjenkjenningsprogrammet for å finne ut om programmet har blitt bedre av opplæringen. Det ble avlest 50 registreringsnummer med totalt 350 tegn, to bokstaver og fem siffer per skilt. Avlesingen foregikk en gang med den originale språkpakken og en gang slik språkpakken var etter opplæring. Antall feilleste tegn og feilleste registreringsnummer ble notert. Det ble også notert hvilke tegn det oftest var feil på.

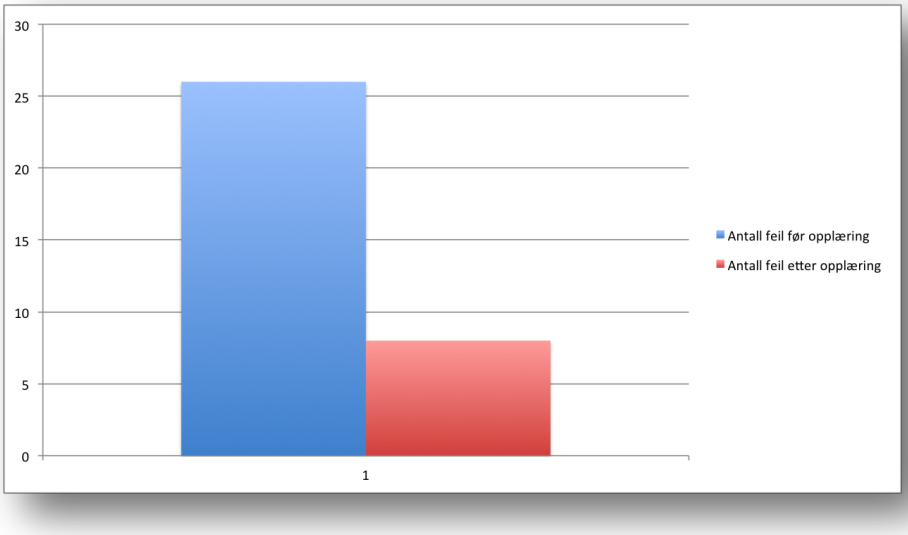
Før opplæring:

- 37 registreringsnummer avlest vellykket
- 13 registreringsnummer med totalt 26 feil

Etter opplæring:

- 43 registreringsnummer avlest vellykket
- 7 registreringsnummer med totalt 8 feil

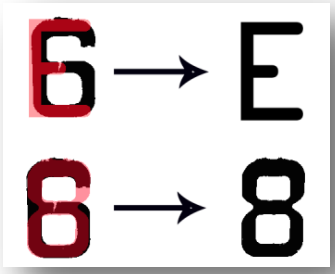
I Figur 8.29 ser en grafisk fremstilling av antall feil før og etter opplæring.



Figur 8.29 Grafisk fremstilling av antall feilleste tegn

Figur 8.30 Eksempel på registreringsnummer med typiske lesefeil

Figur 8.30 viser et skilt med tegn som typisk gikk igjen og ga feil avlesing før opplæring. Dette skiltet ble avlest som "TV E8882". Grunnen til dette er at programmet ser etter likheter med tegn som ligger i språkpakken. Programmet ser en større likhet mellom det første sekstallet og "E" enn et sekstall i språkpakken. Det har også tolket det siste sekstallet som "8". Figur 8.31 viser likhetstrekkene mellom "6" og "E", og "6" og "8".



Figur 8.31 Illustrasjon av likhetstrekk

Da registreringsnummeret i Figur 8.30 ble lest etter opplæring, var det ingen feil. Under opplæringen har disse tegnene blitt lagt til i språkpakken. Tegngjenkjenningsprogrammet vil da se en større likhet mellom tegnene i registreringsnummeret, og tegnene i språkpakken.

Det er brukt 150 registreringsnummer, totalt 1050 tegn i opplæringsfasen. Opplæringen har vært tidkrevende. Dette er fordi en må manuelt kontrollere hvert enkelt tegn. Med mer tid for opplæring av flere tegn, mener gruppen at programmets tegngjenkjenning hadde resultert i flere vellykkede avlesinger.

9.0 Konklusjon

Oppgaven for prosjektgruppen har vært å utvikle og teste en prototype med automatisk skiltgjenkjenning som kan videreutvikles til å erstatte det nåværende parkeringsavgiftsystemet ved Førde lufthamn Bringeland. Prototypen oppfyller de spesifikasjonene som ble satt når prosjektet startet. Ankomst av biler blir registrert ved bildetaking. Bildebehandling blir utført slik at registreringsnummeret er leselig for et tegngjenkjenningsprogram. Registreringsnummeret blir avlest og lagret i en tabell. Ved avreise blir et nytt bilde tatt og den samme bildebehandlingen utføres. Registreringsnummeret blir avlest og knyttet mot registreringsnummeret fra innregistrering. Basert på dette regnes parkeringstid ut. Tiden knyttes opp mot en bruker i brukerdatabasen.

Gruppen har tatt bilder under forskjellige lysforhold. Bildene er tatt i høgskolens garasje med IP-kamera og ute med kompaktkamera. Ved å sammenligne resultater fra bildebehandling ser en at prototypen er avhengig av riktig lysforhold og god bildekvalitet. Med bedre bildekvalitet, øker derimot bildebehandlingstiden. Bildehåndtering må derfor utvides med et køsystem for bildene som skal behandles. Vinkelen på 25,84^{o}, som er testet, er en god vinkel for prototypen. Under bildebehandling må hele skiltplaten finnes og sendes samlet til tegngjenkjenningsprogrammet.

For gode bilder som kan bildebehandles, er plassering av bilene på bildet viktig. Det må brukes sensorer som kan måle fart for videre utvikling. Lysindikasjon til å begrense kjørehastighet er ikke godt nok.

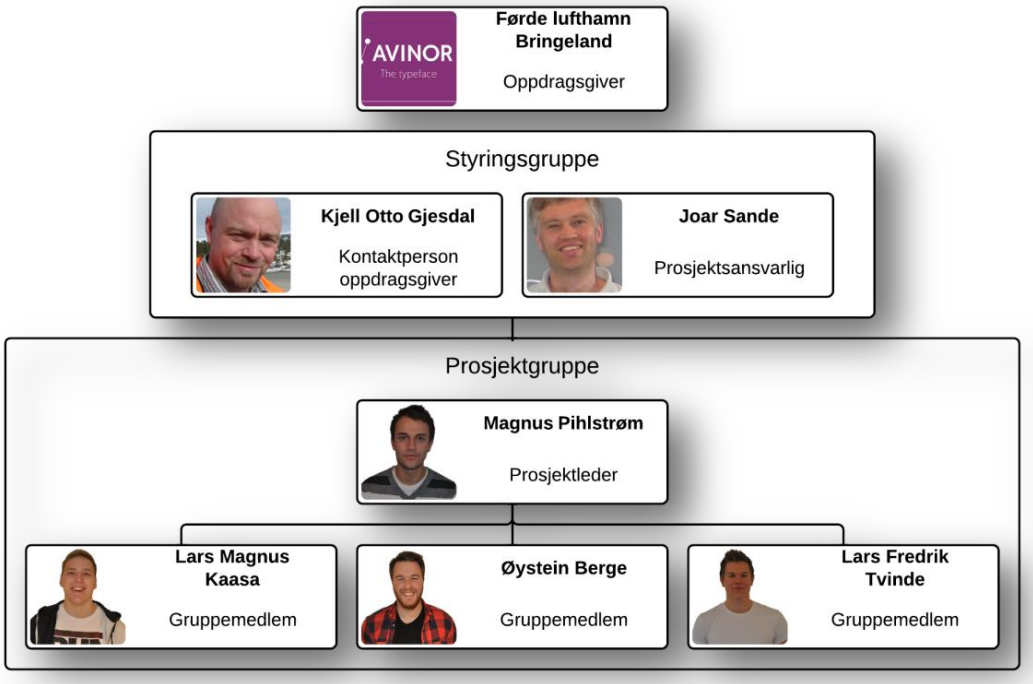
Brukerdatabasen bygges av brukerne ved registrering en gang. Grensesnittet er testet av brukere og forbedret etter testpersonenes tilbakemeldinger. Ansatte ved Førde lufthamn får oversikt over brukerdatabasen ved å logge seg inn med brukernavn og passord. Grensesnitt med mulighet for endring av informasjon i brukerdatabasen er mangelfull og må utbedres. Brukerregistrering må også utvides med flere feilmeldinger tilpasset feil som kan forekomme.

Resultatet er en fungerende prototype. For å teste systemet videre anbefaler gruppen å bruke et kamera som gir bedre bildekvalitet. Videre testing må foregå ute under bedre lysforhold. For å registrere ankomst og avreise av biler i høyere hastighet, må det brukes bedre sensorer og et raskere kretskort.

10.0 Prosjektadministrasjon

Her kommer oversikt over prosjektet. Det er med økonomi og hvordan prosjektgruppen har jobbet gjennom prosjektperioden.

10.1 Organisering



Figur 10.1 Organisasjonskart

Oppdragsgiveren vår er Førde lufthamn Bringeland. Førde lufthamn er en regional flyplass i Gaular kommune. Flyplassen har 87 000 besøkende i året og gjennomsnittlig syv avganger pr. dag. Førde lufthavn har i dag 35 ansatte. 17 fra Avinor, 9 fra Widerøe og 9 fra Securitas.

Styringsgruppen har det overordnede ansvaret under prosjektperioden. Den består av Joar Sande som er ansatt ved HiSF og Kjell Otto Gjesdal som er lufthavnsjef ved Førde lufthamn.

Prosjektgruppen består av avgangsstudenter ved HiSF. Medlemmene har forskjellig erfaring fra både arbeidslivet og drift av organisasjoner.

10.2 Økonomi

Dette prosjektet ble budsjettert med utgift på kr 6100,-. Dette var høyere enn resultatet som endte med totalutgift på kr 3.899,-.

Driftsinntekter	2014	Budsjett 2014	2013
Støtte HiSF	699	1 500	
Sponsorinntekter		800	
Bringeland	2 800	2 800	
Div	400	1 000	
Sum driftsinntekter:	**3 899**	**6 100**	

Figur 10.2 Inntekter/støtte

Figur 10.2 viser driftsinntekter for prosjektet. Sponsorinntektene var planlagt. Med en sponsor kunne gruppen sikre innkjøp av bedre utstyr. Dette fikk ikke gruppen på plass, men det ble fremarbeidet en god rabatt ved kjøp av kamera hos Kontorsenteret AS. Div er utstyr som kabler, koblingsutstyr og annet gruppen ellers skulle betalt for, men var gratis tilgjengelig på skolen. Gruppen har bestemt å ikke ta dette med i regnskapet fordi det kan skape misforståelser ved tilbakebetaling, men det er fortsatt i budsjettet. De kr 400,- som er inn på div, er en bok om OpenCV som gruppen skulle kjøpe, men som biblioteket anskaffet.

Driftskostnader		
Kamera	1 940	1 000
Skjerm		1 800
Innkapsling		1 000
Detektor	191	500
Kabler		150
Arduino-kort		300
Bilgodtgjørelse		500
Modell registrering	1 188	850
Div	580	
Sum driftskostnader:	**3 899**	**6 100**

Figur 10.3 Utgifter

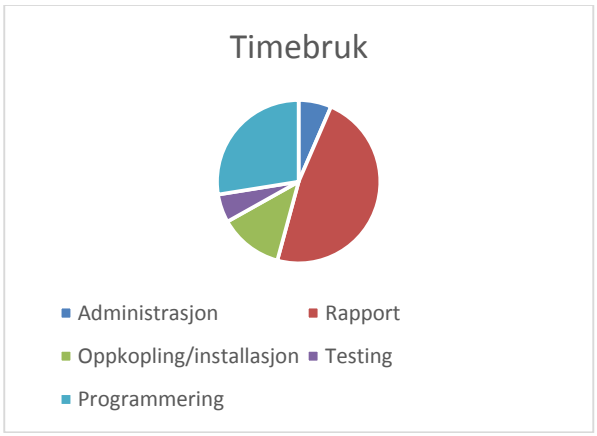
Fordi det ble spart en del penger på flere poster i budsjettet, gikk gruppen til innkjøp av et dyrere kamera, Figur 10.3. Kameraet var da allerede vanntett, så en slapp å gå til innkjøp av innkapsling til dette. Skjerm, kabler og Arduino-kort var tilgjengelig på skolen. Div er kr 400,- for boken om OpenCV som biblioteket støttet med og kr 180,- for lysindikasjon som ble bygget av ungdomsskoleelever på besøk.

Materielliste ligger i Vedlegg 8.

Regnskap er ført i gratis regnskapsprogram fra Landbruksdata Voss AS og er gitt i Vedlegg 9. Dette har gjort økonomien oversiktlig gjennom prosjektperioden.

10.3 Timer brukt

Prosjektgruppen har fordelt det totale timeantallet blant medlemmene som i Figur 10.4.



Magnus Pihlstrøm	503 timer
Lars Fredrik Tvinde	428,5timer
Øystein Berge	506,5timer
Lars Magnus Kaasa	442,5timer

Figur 10.4 Timefordeling totalt

Grunnet sportsarrangementer i mai, ble det mot slutten en skjevfordeling av timeantall totalt mellom medlemmene i prosjektgruppen.

10.3 Arbeidsgjennomføring

Prosjektperioden startet 6. januar. Da begynte gruppen å sette opp budsjett, planlegge fremdrift, skrive prosjektbeskrivelse og skaffe en best mulig oversikt over programdeler som trengtes for å gjennomføre prosjektet. De første to ukene ble det også brukt en del tid på å prøve å skaffe sponsorer til utstyr og område til testing. Forprosjektperioden var viktig for å skaffe god oversikt over hva som trengtes av programdeler og hvilke programmer som kunne brukes sammen. Dette for at prosjektet ikke skulle ende i en "blindgate" ved testing. Testing av systemet ble tidlig avklart kunne starte mandag 17.03.

I begynnelsen av mars ble det jobbet mer spesifikt mot testingen i garasjen. Registreringsmodellen ble bygget og sensorsystem for registrering av bil inn/ut ble planlagt. Når testingen startet, følte hele gruppen at prosjektet var under kontroll.

Det ble tidlig klart at testperioden i garasjen ville dra ut. Dette var noe som var påberegnet, med planlagt testtid til 16.05, altså to måneder. Dette innebar tre uker med feilsøking, tre uker med fullverdig test og tre uker med gjennomgang, analyse og bekreftelse av testdataene. Starten av testtiden gikk med til å sørge for at bildene ble tatt når bilene var på samme punkt. Her ble programmet gjennomgått og forbedret og etter hvert ble Arduino Uno byttet ut mot Arduino Mega. Når plassering av bilene på bildene var tilfredsstillende, ble det oppdaget problemer med lysforholdene. Det gikk mye tid bort når gruppen forsøkte å forbedre bildekvaliteten.

I mai var det sportsarrangementer som førte til at to av gruppemedlemmene ikke fikk jobbet like mye med prosjektet som planlagt i Gantt – skjema i Vedlegg 10. Dette gjaldt spesielt medlemmet som hadde ansvaret for brukerdatabasen. Dette førte på slutten til at finjustering og optimalisering av databasen ikke ble gjennomført. Fravær grunnet uttak til landslag var en del av vår risikovurdering i Vedlegg 11.

Det ble i stedet for en fullstendig test, laget en film om prototypens funksjon. Prototypen er deltestet. En test for brukergrensesnitt, en for bildebehandlingsprogrammet og en for tegngjenkjenningsprogrammet. Gruppen er fornøyd med gjennomføringen av testene. Testene avklarte spørsmål gruppen hadde rundt prototypen.

Gruppen er meget fornøyd med arbeidsfordelingen, hvor alle medlemmene har tatt mye ansvar for ferdigstilling av prototypen.

10.4 Møter

Prosjektgruppen har hatt interne statusmøter hver tirsdag kl. 09.00, hvor fremdriften i prosjektet har blitt gjennomgått. Det ble også satt opp ukemål på hvert møte.

Prosjektgruppen og styringsgruppen så ikke nødvendigheten med å ha møte hver 14. dag, og bestemte seg for å ha møte én gang i måneden i stedet.

Innkallelser og referat fra møtene ligger i Vedlegg 12.

10.5 Nettside

Gruppen har i løpet av prosjektperioden laget og driftet en hjemmeside. Siden er laget med hjelp av Wordpress med grunnlag i en gratis mal. I tillegg er den utvidet med slideshow av bilder og visning av siste nyheter.

Underveis har gruppen jevnlig oppdatert siden med nyheter, bilder og videoer fra arbeidet.

Adressen til hjemmesiden er:

http://studprosjekt.hisf.no/~14psys/

11.0 Figurliste

12.0 Referanseliste

[1] itseez, «OpenCV About,» OpenCV, [Internett]. Available: http://opencv.org/about.html. [Funnet 13. februar 2014].

[2] S. Audet, «JavaCV,» [Internett]. Available: https://code.google.com/p/javacv/. [Funnet 13. februar 2014].

[3] Oracle Corporation, «MySQL Reference Manual,» 20114. [Internett]. Available: http://dev.mysql.com/doc/refman/4.1/en/what-is-mysql.html. [Funnet 19 mai 2014].

[4] K. Toft Hansen og T. Mallaug, «Databaser,» i *Databaser*, Oslo, Gyldendal Norsk Forlag AS, 2011, p. 27.

[5] T. Eng, «Uryddig vei til nye bilnummerskilt,» Typografi i Norge, [Internett]. Available: http://www.typografi.org/bilskilt/bilskilt.html. [Funnet 13. februar 2014].

[6] OCRWizard, «How OCR Software Works,» OCRWizard, 2011. [Internett]. Available: http://ocrwizard.com/ocr-software/how-ocr-software-works.html. [Funnet 13. februar 2014].

[7] Robonor, «Hva er Arduino,» Robonor, [Internett]. Available: http://www.robonor.no/mage/default/arduinoinfo. [Funnet 22 Mai 2014].

[8] QuinStreetEnterprise, «webopedia,» QuinStreet Inc., 2014. [Internett]. Available: http://www.webopedia.com/TERM/G/gray_scaling.html. [Funnet 3 april 2014].

[9] E. Wolfart og S. Perkins, «HIPR2 Explore with Java,» 2003. [Internett]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/median.htm. [Funnet 22 mai 2014].

[10] T. Eckel, «arduino.cc,» Arduino, 15 august 2012. [Internett]. Available: http://playground.arduino.cc/Code/NewPing#History. [Funnet 30 april 2014].

[11] AOL Tech, «Engadget,» AOL Inc., [Internett]. Available: http://www.engadget.com/products/raspberry-pi/a-b/specs/. [Funnet 22 mai 2014].

[12] M. Knoff og J.-A. Saarnak, Interviewees, *Bachelorprosjekt 2013/14*. [Intervju]. 31 oktober 2013.

[13] Ubuntu, «OCR - Optical Character Recognition,» Ubuntu, 2013. [Internett]. Available: https://help.ubuntu.com/community/OCR. [Funnet 13. februar 2014].

[14] Sourceforge.net, «Tess4J,» Sourceforge.net, [Internett]. Available: http://tess4j.sourceforge.net/. [Funnet 13. februar 2014].

[15] R. Smith, «An Overview of the Tesseract OCR Engine,» Google Inc.

[16] Redcorn Ltd. E.L.V & Nuisance Vehicle Specialist, «Redcorn,» [Internett]. Available: http://www.redcorn.co.uk/content/view/14/31/1/0/. [Funnet 9 mars 2014].

[17] Worldwide Independent Inventors Association, «Worldwideinvention,» 22 november 2009. [Internett]. Available: http://www.worldwideinvention.com/articles/details/272/Automatic-number-plate-recognition-Inventions-for-sale-Patent-for-sale-Sell-patents-Sell-invention-.html. [Funnet 9 mars 2014].

[18] M. Nixon og A. Aguado, «Feature Extraction & Image Processing,» 2002.

13.0 Vedlegg

Vedlegg 1: Rapport fra OR2-300 Prosjektstyring med prosjekt (ZIP)

Vedlegg 2: Forprosjekt HO2-300 (ZIP)

Vedlegg 3: Datablad ultralydsensor

Vedlegg 4: Vivotek IP8332 spesifikasjoner

Vedlegg 5: Algoritme prototype

Vedlegg 6: Vivotek IP8332 installasjonsanvisning (ZIP)

Vedlegg 7: Instruksjoner gitt ved test av brukergrensesnitt

Vedlegg 8: Materielliste

Vedlegg 9: Regnskap (ZIP)

Vedlegg 10: Gantt-skjema

Vedlegg 11: Risikovurdering

Vedlegg 12: Møteinnkallelser og møtereferat (ZIP)

Vedlegg 13: Fullmaktsskjema

Ultrasonic ranging module : HC-SR04

Specifications:

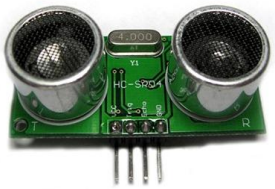
power supply :5V DC

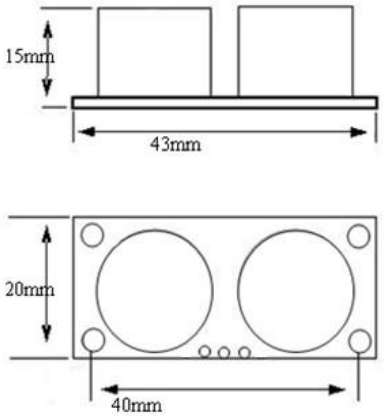
quiescent current : <2mA

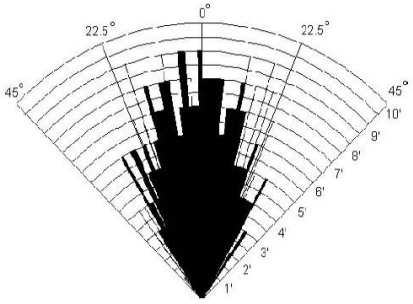
effectual angle: <15°

ranging distance : 2cm – 500 cm

resolution : 0.3 cm

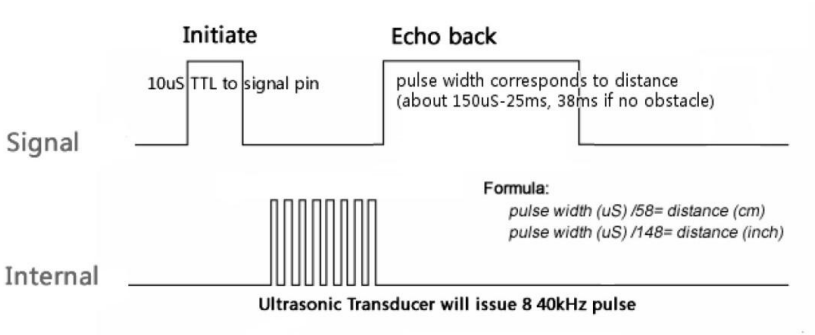






Practical test of performance,
Best in 30 degree angle

Sequence chart



A short ultrasonic pulse is transmitted at the time 0, reflected by an object. The senor receives this signal and converts it to an electric signal. The next pulse can be transmitted when the echo is faded away. This time period is called cycle period. The recommend cycle period should be no less than 50ms. If a 10μs width trigger pulse is sent to the signal pin, the Ultrasonic module will output eight 40kHz ultrasonic signal and detect the echo back. The measured distance is proportional to the echo pulse width and can be calculated by the formula above. If no obstacle is detected, the output pin will give a 38ms high level signal.

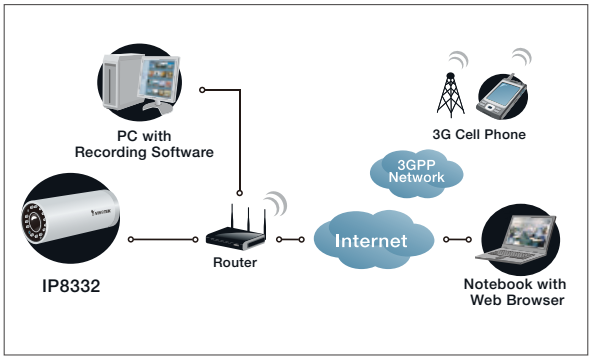
Library:
http://iteadstudio.com/store/images/produce/Robot/HCSR04/Ultrasonic.rar

Specifications

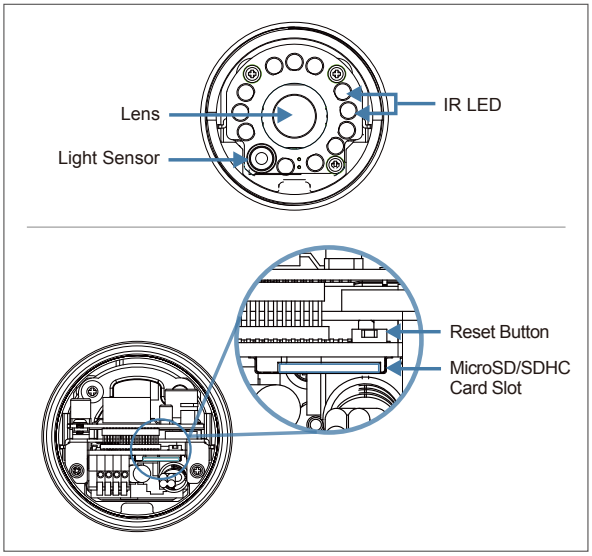
Ver 1.3

System	· CPU: TI DM365 SoC · Flash: 128MB · RAM: 256MB · Embedded OS: Linux 2.6
Lens	· Board lens, f = 3.6 mm, F1.8, Fixed · IR corrected · Removable IR-cut filter for day & night function
Angle of View	· 56° (horizontal) · 41° (vertical) · 71° (diagonal)
Shutter Time	· 1/5 sec. to 1/25,000 sec.
Image Sensor	· 1/4" CMOS sensor in 1280x800 resolution
Minimum Illumination	· 0 Lux / F1.8 (IR LED on)
IR Illuminators	· Built-in IR illuminators, effective up to 15 meters · IR LED*12
Video	· Compression: H.264, MJPEG & MPEG-4 · Streaming: Simultaneous multiple streams H.264 streaming over UDP, TCP, HTTP or HTTPS MPEG-4 streaming over UDP, TCP, HTTP or HTTPS MPEG-4 multicast streaming MJPEG streaming over HTTP or HTTPS · Supports activity adaptive streaming for dynamic frame rate control · Supports 3GPP mobile surveillance · Frame rates: H.264: Up to 30/25 fps at 1280x800 MPEG-4: Up to 30/25 fps at 1280x800 MJPEG: Up to 30/25 fps at 1280x800
Image Settings	· Adjustable image size, quality and bit rate · Time stamp and text caption overlay · Flip & mirror · Configurable brightness, contrast, saturation, sharpness and white balance · AGC, AWB, AES · Automatic, manual or scheduled day/night mode · Supports privacy masks
Networking	· 10/100 Mbps Ethernet, RJ-45 · Onvif Support · Protocols: IPv4, IPv6, TCP/IP, HTTP, HTTPS, UPnP, RTSP/RTP/RTCP, IGMP, SMTP, FTP,DHCP, NTP, DNS, DDNS, PPPoE, CoS, QoS, SNMP and 802.1X
Alarm and Event Management	· Triple-window video motion detection · Tamper detection · One D/I for external sensor · Event notification using HTTP, SMTP or FTP · Local recording of MP4 file
On-board Storage	· MicroSD/SDHC card slot · Stores snapshots and video clips
Security	· Multi-level user access with password protection · IP address filtering · HTTPS encrypted data transmission · 802.1X port-based authentication for network protection
Users	· Live viewing for up to 10 clients
Dimension	· Ø 60 mm x 170 mm · Cable length: 400 mm · Cable diameter: Ø 12 mm; Ring: Ø 18 mm
Weight	· Net: 702 g
Power	· 12V DC · 24V AC · Power consumption: Max. 4 W · 802.3af compliant Power-over-Ethernet (Class 2)
Housing	· Weather-proof IP66-rated housing
Approvals	· CE, LVD, FCC, VCCI, C-Tick, UL
Operating Environments	· Temperature: -20 ~ 50 ° C (-4 ~ 122 ° F) · Humidity: 90% RH
Viewing System Requirements	· OS: Microsoft Windows 7/Vista/XP/2000 · Browser: Mozilla Firefox, Internet Explorer 7.x or above · Cell phone: 3GPP player · Real Player: 10.5 or above · Quick Time: 6.5 or above
Installation, Management, and Maintenance	· Installation Wizard 2 · 32-CH ST7501 recording software · Supports firmware upgrade
Applications	· SDK available for application development and system integration
Warranty	· 24 months

System Overview



External and Internal View

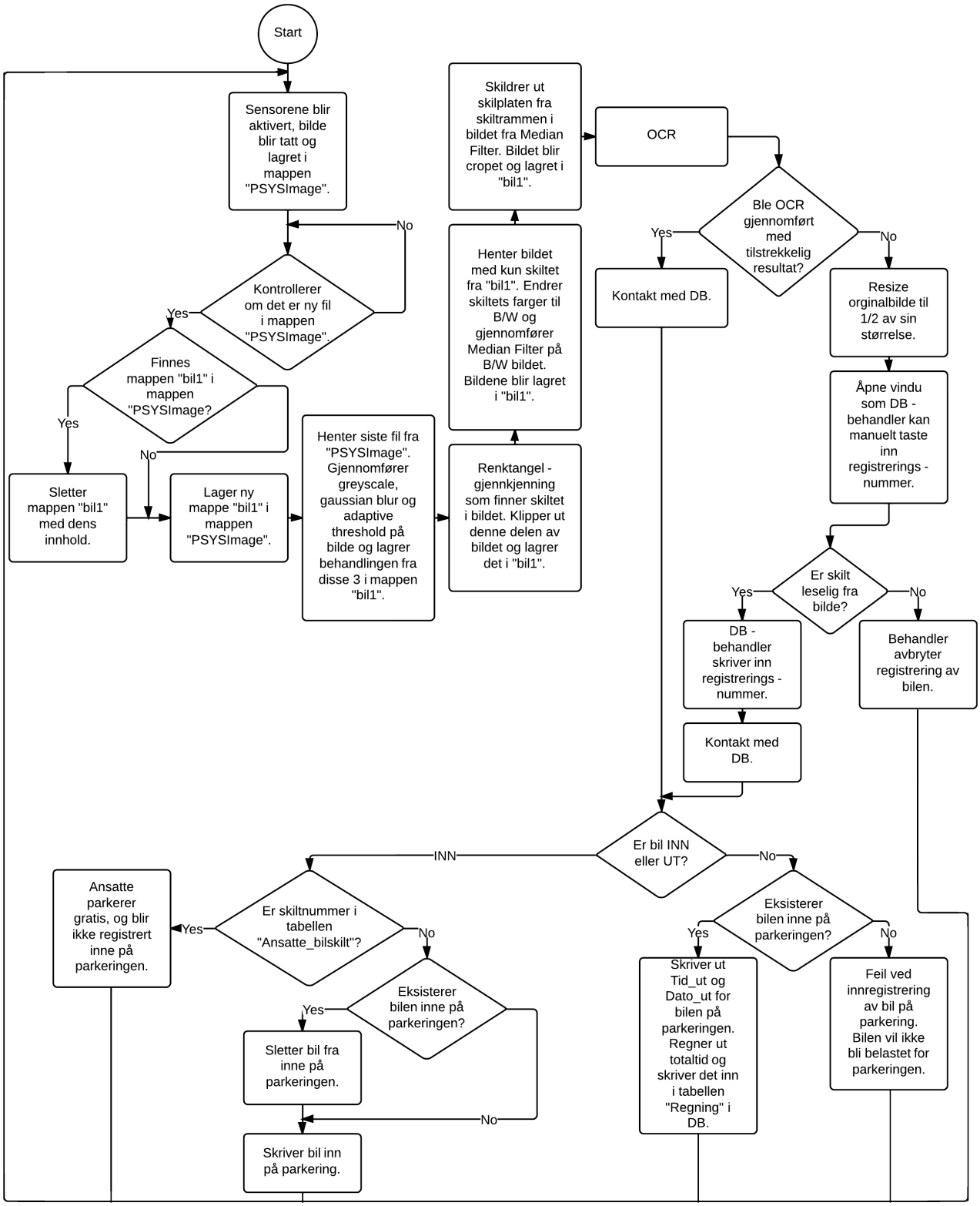


Distributed by:

VIVOTEK
www.vivotek.com

VIVOTEK INC.
6F, No.192, Lien-Cheng Rd., Chung-Ho, New Taipei City, 235, Taiwan, R.O.C.
| **T:** +886-2-82455282 | **F:** +886-2-82455532 | **E:** sales@vivotek.com

VIVOTEK USA, INC.
2050 Ringwood Avenue, San Jose, CA 95131
| **T:** 408-773-8686 | **F:** 408-773-8298 | **E:** salesusa@vivotek.com


Start
Sensorene blir aktivert, bilde blir tatt og lagret i mappen "PSYSImage".
Kontrollerer om det er ny fil i mappen "PSYSImage".
No
Yes
Finnes mappen "bil1" i mappen "PSYSImage?
Yes
No
Sletter mappen "bil1" med dens innhold.
Lager ny mappe "bil1" i mappen "PSYSImage".
Henter siste fil fra "PSYSImage". Gjennomfører greyscale, gaussian blur og adaptive threshold på bilde og lagrer behandlingen fra disse 3 i mappen "bil1".
Renktangel - gjennkjenning som finner skiltet i bildet. Klipper ut denne delen av bildet og lagrer det i "bil1".
Henter bildet med kun skiltet fra "bil1". Endrer skiltets farger til B/W og gjennomfører Median Filter på B/W bildet. Bildene blir lagret i "bil1".
Skildrer ut skilplaten fra skiltrammen i bildet fra Median Filter. Bildet blir cropet og lagret i "bil1".
OCR
Ble OCR gjennomført med tilstrekkelig resultat?
Yes
No
Kontakt med DB.
Resize orginalbilde til 1/2 av sin størrelse.
Åpne vindu som DB - behandler kan manuelt taste inn registrerings - nummer.
Er skilt leselig fra bilde?
Yes
No
DB - behandler skriver inn registrerings - nummer.
Behandler avbryter registrering av bilen.
Kontakt med DB.
Er bil INN eller UT?
INN
No
Er skiltnummer i tabellen "Ansatte_bilskilt"?
Yes
No
Ansatte parkerer gratis, og blir ikke registrert inne på parkeringen.
Eksisterer bilen inne på parkeringen?
Yes
No
Sletter bil fra inne på parkeringen.
Skriver bil inn på parkering.
Eksisterer bilen inne på parkeringen?
Yes
No
Skriver ut Tid_ut og Dato_ut for bilen på parkeringen. Regner ut totaltid og skriver det inn i tabellen "Regning" i DB.
Feil ved innregistrering av bil på parkering. Bilen vil ikke bli belastet for parkeringen.

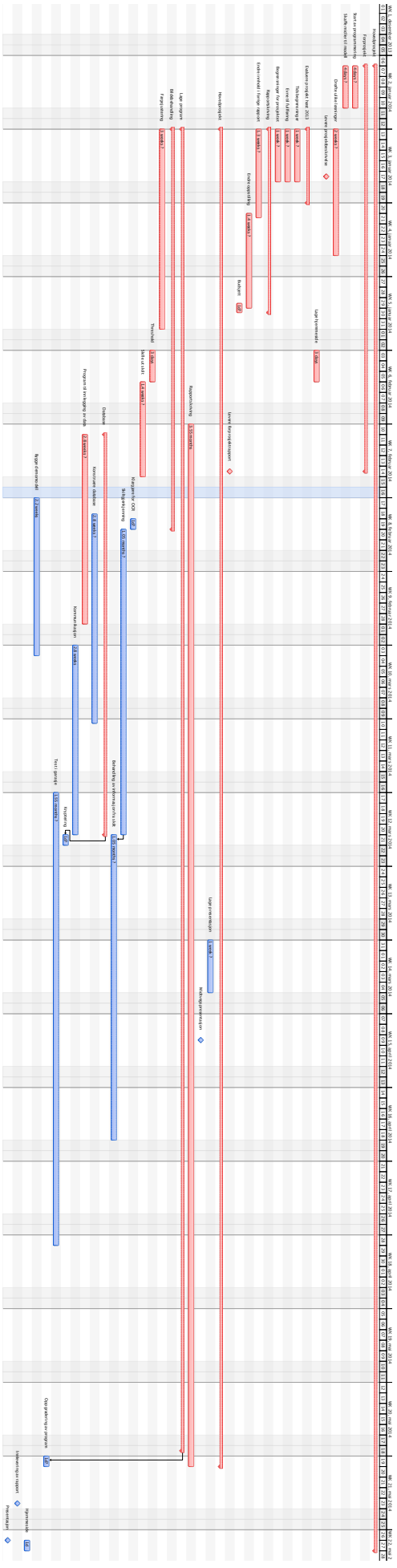
Instruksjoner

1. Trykk «Ny kunde»
2. Skriv info
3. Alle felt må fylles ut.
4. E-postadresse må være gyldig (bestå av @ og .no, .com osv)
5. Registreringsnummer må bestå av to bokstaver og fem tall.
6. Når du er ferdig, trykk «Registrer».
7. Når dialogboks dukker opp og sier du er registrert, trykk «OK».

Material	**Antall**	**Enhet**
Transporthjul 80mm	4	Stk
Hylleplate hvit 800x200x18mm	6	Stk
Hylleplate hvit 1500x400x18mm	4	Stk
Vinkelbeslag 50x50x35mm	20	Stk
Vinkelbeslag 90x90x40mm	14	Stk
Lekter 48x48mm	13,5	m
IR-lys	1	Stk
IP-kamera	1	Stk
Koblingsbokser	2	Stk
Ultralydsensorer	2	Stk
Cat5e kabel	30	m
Innkapsling/ koblingsboks	1	Stk
Arduino Mega	1	Stk
Relékort for Arduino (6 reléer)	1	Stk
Rekkeklemmer	1	Stk
Dobbeltsidig teip	2	m
LED-lyskilder	2	Stk
TET-nipler	2	Stk
D-Link router	1	Stk

Tekst markert med gult er materiell gruppen har fått sponset eller lånt.

Vedlegg 10



Vedlegg 6 Risikovurdering

Risiko	Sannsynlighet	Konsekvens	Risikoverdi
Sykefravær	1	2	2
Tidsnød	2	5	10
Økonomi	1	5	5
Volleyball	2	4	8
Manglende kunnskap om database programmering	2	5	10
Manglende kunnskap om Open CV programmering	2	5	10
Kommunikasjonssvikt	1	4	4
Internkonflikt	1	5	5
Feil på utstyr	2	5	10
Kollisjon med valgfag	2	3	6

Sykefravær

Sykefravær er en risiko som ikke kan forutses. Gruppemedlemmene har ulike oppgaver, men kan overta de forskjellige oppgavene dersom det skal bli et behov. Anser derfor risikoen som liten så lenge ikke flere gruppemedlemmer får et vesentlig fraværstall.

Tidsnød

Prosjektet har flere tidskrevende faktorer så det kreves god planlegging.

Økonomi

Vi har budsjettert prosjektet der det ikke er noen variable faktorer som kan øke mye i pris. Gruppen ser på denne delen som lite risikofylt.

Volleyball

To av medlemmene i gruppen er toppidrettsutøvere og har flere opphold både innlands og utlands som kan øydelegge for noe av arbeidet. Mest sannsynlig vil det ikke være for mye bry siden det meste kan arbeides med over alt i verden.

Manglende kunnskap om database og OpenCV/JavaCV programmering

I prosjektet skal det lages en demomodell som ønskes så lik som mulig opp mot en virkelig modell. Det kreves derfor en hel del kunnskap på dette området. Heldigvis har gruppen en god rettleder som kan bistå med mye kunnskap. Dersom man ikke får det til vil det ha stor konsekvens for resultatet.

Kommunikasjonssvikt

Gruppen skal ha faste arbeidstider på skolen som vil føre til jevnlig kommunikasjon. Samtidig skal vi ha ukentlige statusmøter som vil klargjøre arbeidsfordeling. Sosiale medier i form av Facebook blir brukt til å dele informasjon. Samt Dropbox er en formidlingssentral for prosjektet.

HØGSKULEN i SOGN OG FJORDANE

Vedlegg 13

Fullmaktsskjema / Avtale

Om innsyn i og elektronisk publisering av bacheloroppgåve eller masteroppgåve.

Namn på forfattar(ar): Lars Magnus Kaasa, Øystein Berge, Lars Fredrik Tvinde, Magnus Pihlstrøm

(Mrk: Skriv tydeleg (STORE BOKSTAVAR!.) Kvar student fyller ut eige skjema)

Tittel på oppgåva: P-sys, Parkeringssystem med automatisk skiltgjenkjenning

Utgjevingsår: 2014 Kandidatnummer: LMK(9), ØB(16), LFT(15), MP(11)

Namn på studium/Emnekode: H02-300 Hovedprosjekt 2014 Vår

Med dette gir ovannemnde student Høgskulen i Sogn og Fjordane (HiSF) retten til å gjere oppgåva tilgjengeleg for ålmenta gjennom å leggje oppgåva i ein database med open access (”Brage”) og/eller publisere trykte eksemplar av oppgåva. Brukarane av Brage har retten til fritt å kopiere eller vidareformidle materialet til ikkje-kommersiell bruk.

Studenten beheld opphavsretten til oppgåva si og kan samtidig gjere oppgåva tilgjengeleg på andre måtar.

Studenten garanterer at han er opphavsperson til innlevert materiale og har fullstendig råderett over dette, evt. at han har innhenta samtykke frå dei opphavspersonane han har nytta i oppgåva. Studenten garanterer at han ikkje har kjennskap til eller mistanke om at oppgåva inneheld materiale som kan stride mot gjeldande norsk rett eller innehalde lenkjer eller andre koplingar til slikt materiale.

Dersom HiSF skulle bli gjort erstatningsansvarleg overfor tredjepart fordi studenten ikkje oppfyller garantiar etter denne avtala, er studenten plikta til å halde HiSF fullt ut skadesløs.

Stad Dato

Førde 23.05.14

Signatur

Magnus Pihlstrøm Øystein Berge

Lars Fredrik Tvinde Lars M. Kaasa

03.10.12