Sjøkrigsskolen

Bacheloroppgave

Personellkontroll system for fartøy

av

Jørgen Hjellup Horne

Vebjørn Bryne Nygård

Levert som en del av kravet til graden:

BACHELOR I MILITÆRE STUDIER MED FORDYPNING I ELEKTRONIKK OG DATA

Innlevert: Desember 2019

Godkjent for offentlig publisering

Publiseringsavtale

En avtale om elektronisk publisering av bachelor/prosjektoppgave

Kadettene har opphavsrett til oppgaven, inkludert rettighetene til å publisere den.

Alle oppgaver som oppfyller kravene til publisering vil bli registrert og publisert i Bibsys Brage når kadetten(ene) har godkjent publisering.

Oppgaver som er graderte eller begrenset av en inngått avtale vil ikke bli publisert.

Vi gir herved Sjøkrigsskolen rett til å gjøre denne oppgaven tilgjengelig elektronisk, gratis og uten kostnader	[X] Ja	[ ] Nei
Finnes det en avtale om forsinket eller kun intern publisering? (Utfyllende opplysninger må fylles ut)	[ ] Ja	[X] Nei
Hvis ja: kan oppgaven publiseres elektronisk når embargoperioden utløper?	[ ] Ja	[ ] Nei

Plagiaterklæring

Vi erklærer herved at oppgaven er mitt eget arbeid og med bruk av riktig kildehenvisning.

Vi har ikke nyttet annen hjelp enn det som er beskrevet i oppgaven.

Vi er klar over at brudd på dette vil føre til avvisning av oppgaven.

Dato: 04 – 12- 2019

Jørgen Hjellup Horne _____
Kadett navn

_____
Kadett, signatur

Vebjørn Bryne Nygård _____
Kadett navn

_____
Kadett, signatur

Forord

Bacheloroppgaven er et krav for bachelor i militære studier med fordypning i elektronikk og data ved Sjøkrigsskolen. Oppgaven har gitt oss muligheten til å ta i bruk et stort spekter av fagfeltet vårt, og har gitt oss bedre kompetanse på datakommunikasjon, programmering, systemtenkning, logisk styring og prosjekt styring. Arbeidet med oppgaven startet i mai 2019 og ble avsluttet i desember 2019.

Både kadett Horne og kadett Nygård var veldig interessert i å utvikle sin kompetanse innenfor programmering og datakommunikasjon. Da ideen om personellkontroll på fartøy dukket opp ble vi raskt enige om hva vi ønsket å få ut av oppgaven. Under arbeidet med oppgaven har kadett Nygård hatt hovedansvaret for det tekniske og kadett Horne for det administrative.

Vi ønsker å rette en stor takk til medkadetter, lærere og eksterne for verdifulle diskusjoner og viktige innspill til oppgaven, vi ønsker å rette en ekstra stor takk til:

Alexander Sauter for teknisk støtte og veiledning.

Terje Fykse for meget god veiledning og nyttige diskusjoner under hele bacheloroppgaven.

Bergen, Sjøkrigsskolen, 04-12-2019

_____

Jørgen Hjellup Horne

_____

Vebjørn Bryne Nygård

Oppgaveformulering

Om bord på et fartøy kan det oppstå kaotiske situasjoner hvor ledelsen ikke har umiddelbar kontroll på besetningen sin. Har man bedre personellkontroll kan for eksempel et søk om bord enklere ledes da man kan prioritere seksjoner av fartøyet man vet det befinner seg mennesker. Dersom man har mer informasjon som kan understøtte beslutninger kan det i beste fall redde liv. Det er motivasjonen bak oppgaven.

Vi ønsker derfor å konstruere en modell av et personellkontrollsystem som kan hjelpe til å bedre personellkontroll på fartøyet.

Sammendrag

Om bord på et fartøy kan det oppstå kritiske situasjoner hvor usikkerhet råder. De første minuttene vil ofte være kaotiske og første prioritet vil være personellkontroll. Om bord på fartøy i dag blir personellkontroll kun basert på muntlig kommunikasjon og det er få, om noen, hjelpemidler til å finne manglende personell. For å kunne effektivisere og understøtte jobben med å skaffe personellkontroll er oppgaven å designe en modell av et personellkontrollsystem som viser en grafisk fremstilling av besetningens lokasjon om bord på fartøyet. Av hensyn til personvern vil det til vanlig, ikke være mulig å se hvem som er hvor, kun hvor mange i hver seksjon. Om det skulle oppstå en kritisk situasjon vil brukeren med autorisasjon kunne logge seg inn og hurtig kunne få oversikt over navn på personer i de forskjellige seksjonene.

Oppgaven har tatt for seg konstruksjon og design av en modell av et personellkontrollsystem, med et fokus på å fremstille en besetnings lokasjon om bord på et fartøy på en oversiktlig og ryddig måte. Oppgaven har hatt fire hovedmål:

1. Lage et brukergrensesnitt som gir en enkel og tydelig fremstilling av besetningens lokasjon.
2. Besetningens lokasjon om bord må bestemmes og endres ved avlesninger av Radio Frequency Identification (RFID)-brikke.
3. Brukergrensesnittet skal være enkelt å bruke, og ikke kreve mye opplæring.
4. Systemet skal ikke krenke, men opprettholde enkeltmenneskets personvern og rett til privatliv.

Hele systemet består av en hardwaredel, en datamaskin som er koblet til en mikrokontroller. Denne styrer igjen en RFID-antenne som kan lese av en RFID-brikke når besetningen passerer den. Antennen er for eksempel plassert ved et skott. Den andre delen er software som behandler dataen som antennen henter fra brikken og fremstiller brikkens-ID grafisk på nettsiden i riktig seksjon. Dette gjøres basert på et bilde av fartøyet.

Måloppnåelsen har vært høy og resultatet står i stil med, og gjør alt, som målene tilsier at oppgaven skal gjøre.

Den informasjonen en vaktsjef eller en fartøyssjef kan hente ut fra et slikt system bør kunne understøtte kritiske beslutninger i situasjoner med tidsnød og kan da være med på å gi et bedre beslutningsgrunnlag. Det anbefales derfor at Forsvaret vurderer å implementere et slikt system på sine fartøyer.

Innholdsfortegnelse

Figurer

Tabeller/Diagrammer

Nomenklatur / Forkortelser / Symboler

ER-diagram – Entity Relations Diagram

HF – High Frequency

HMI – Human Machine Interface

HTML – HyperText Markup Language

LAN – Local Access Network

LF – Low Frequency

NFC – Near Field Communication

PHP – Hypertext Preprocessor

RFID – Radio Frequency Identification

SQL – Structured Query Language

UDP – User Datagram Protocol

UHF – Ultra High Frequency

Ordforklaringer

Arduino – Mikrokontroller levert av en leverandør med samme navn

Java – Programmeringsspråk.

Linux – Operativsystem.

Mikrokontroller – Programmerbar prosessor som henter inn data fra sensorer.

Node – Knutepunkt.

Script – En kommandoliste som kjøres av et program.

1 Innledning

I innledningen vil leseren få en oversikt over motivasjonen bak oppgaven, hvilke rammer som eksisterer, hva målsetningen er, metoden som er brukt og hvordan oppgaven er bygget opp. Hensikten er at leseren skal kunne sitte med bakgrunnsinformasjon, som kan gjøre det lettere å forstå valg gjort underveis i oppgaven. Det er også viktig at leseren har et korrekt bilde av det som er forsøkt oppnådd når oppgaven blir lest.

1.1 Bakgrunn

En novemberkveld 2018 kolliderte Helge Ingstad med tankskipet Sola TS. Gjennom foredrag av og samtaler med folk som var om bord hersker det liten tvil om at det var stor usikkerhet rundt hva som hadde hendt, og hvor personellet om bord på fregatten var i de innledende minuttene. Takket være et godt trent mannskap ble alle liv berget. Det kom også frem at det tok betydelig tid før de fikk personellkontroll på hele besetningen (Rapport Helge Ingstad, 2019). På besøk hos Equinor sin oljeplattform Johan Sverdrup fattet vi interesse i deres system for personellkontroll. Antenner plassert rundt om på plattformen kunne seksjonere inn plattformen, og alle ansatte måtte gå med en passiv RFID-chip på kroppen når de beveget seg over le. I kontrollrommet ble personers plassering fremvist på en storskjerm, de hadde dermed alltid kontroll på hvor mange som befant seg i de forskjellige seksjonene. Den økte informasjonsmengden et slikt verktøy kan gi, kan betraktelig styrke beslutningsgrunnlaget i de første, og mest kritiske, minuttene av en krisesituasjon. I lys av Helge Ingstad ulykken var det relevant å konstruere en modell av et RFID-system som kan gi ledelseselementet om bord bedre kontroll over personellet på fartøyet.

Det ble ansett til å være mer hensiktsmessig å begrense oppgaven til kjerne-funksjonalitetene, det å loggføre og overvåke personellbevegelser. Oppgaven skulle vise funksjonsmåten for et minimalsystem som lett kan skaleres opp til å dekke et helt fartøy. Da prosessen med å finne problemstilling startet, var oppgavens mål å konstruere hele systemet fra bunnen av. Dette ville ha krevd kompetanse fra mer eller mindre alle fag elektronikk og data-linjen har hatt på skolen. Etter samtaler med kollegaer og veileder konkluderte kom det frem at det å bygge hele systemet fra bunnen var i overkant ambisiøst. Avgjørelsen falt derfor på å kjøpe hylleprodukter og fokusere på programmeringen. Det resulterte i problemstillingen til oppgaven: Konstruere en modell for et personellkontrollsystem om bord på fartøy.

1.2 Mål

Målet med oppgaven er å designe et system med enkelt og brukervennlig grensesnitt som gir sentrale roller om bord på et fartøy økt kontroll på eget personell. For å oppnå målet er følgende mål i prioritert rekkefølge definert:

1. **Brukergrensesnittet må kunne gi informasjon på en ryddig og enkel måte om besetningens posisjon på fartøyet.**
    - For at Human Machine Interface(HMI) skal være til hjelp er det viktig at den kan vise hvor om bord på fartøyet personell befinner seg. Med det menes det at nødvendig personell skal kunne se i hvilken seksjon av fartøyet hver enkelt person befinner seg i.
2. **Kunne hente ut data fra RFID Tags.**
    - For å kunne fremvise endring av lokasjon på en besetning er det kritisk å motta informasjon om deres plassering om bord. Da er systemet avhengig av å få inn data fra RFID-brikker.
3. **Brukergrensesnittet må være brukervennlig.**
    - Det skal være enkelt å forstå produktet og dets funksjoner. Det må legges inn funksjoner som kan løse hverdagslige oppgaver på lavest mulig nivå. Derfor må det være enkelt å legge inn nye RFID-brikker med tilhørende personer eller fjerne tidligere RFID-brikker som ikke lenger er i bruk, for å alltid ha oppdatert informasjon synlig på HMI.
4. **Tilfredsstille forsvarets personvernregler.**
    - Skal det bli aktuelt å implementere denne typen system i virkeligheten er det viktig at systemet tilfredsstiller gjeldende regelverk. Derfor er det mål å designe et system som i minst mulig grad krenker enkeltpersoners privatliv og personvern. Dette må skje uten at det bryter med intensjonen til oppgaven.

1.3 Begrensninger

Rent formelt er oppgaven avgrenset til 6 måneder og har 20.000kr i budsjett. Oppgaven har potensial til å inkludere veldig mange av fagfeltene som det undervises i på elektronikk og data utdanningen ved sjøkrigsskolen. For å ikke gjøre oppgaven for teknisk har det vært nødvendig å definere en del begrensninger. Hovedfokus har vært på å utvikle et brukergrensesnitt for å kunne behandle data fra RFID-brikker og fremstille personers plassering på en oversiktlig måte. De prioriteringene som har blitt gjort har ført at enkelte deler av systemet er hyllevare, andre deler er hentet fra åpenkilde biblioteker. Derimot er majoriteten programmert og designet laget spesifikt til denne oppgaven.

De spesifikke kravene som er til oppgaven, eller ble identifisert underveis i arbeidet er som følger:

- RFID brikke og antenne skal være et hylleprodukt.
- HMI skal være en grafisk fremstilling som skal være ryddig og kreve minimalt med opplæring for å forstå.
- Funksjonaliteten til programmet har vært i fokus og det har ikke blitt tatt hensyn til datasikkerhet. Det blir ikke gjort noe forsøk på å kryptere eller sikre dataene i noen annen grad enn den sikkerheten teknologien i seg selv gir.
- Det er ikke tatt hensyn til behandling av personopplysninger da det kun kommer til å bruke fiktive personer i dette test-systemet.
- Budsjettramme på 20.000 NOK.
- Tidsramme på cirka 6 måneder

1.4 Metode

Oppgaven har vært delt i flere faser. Den første fasen gikk ut på å skaffe seg kunnskap om RFID-systemer og hvordan teknologien blir anvendt i forskjellige miljøer i dag. Videre fulgte en designfase hvor det ble undersøkt mulige komponenter som kunne brukes, hva ønsket måloppnåelse var og hvordan de ulike målene skulle vektlegges. Denne fasen ble i stor grad gjort ved hjelp av kompetanse internt på skolen, i den hensikt å holde budsjett samt å ha et realistisk størrelsesmål for oppgaven. Deretter startet konstruksjonsfasen hvor alt av fysiske komponenter ble koblet sammen til et system. Videre måtte det kunne hentes avlest data fra systemet. Etter konstruksjonsfasen startet programmeringsfasen hvor data fra de fysiske komponentene skulle kunne presenteres for en bruker. Til

slutt inntok kadett Horne rollen som "djevelens advokat" for å teste programmets robusthet. Kontinuerlig gjennom utviklingen har det foregått omfattende testing av programmeringskode. Testingen er ikke beskrevet i stor grad i denne oppgaven, da den er gjort kontinuerlig underveis i konstruksjonen av produktet.

Komponentene som er benyttet er hyllevare, men de måtte kobles sammen og programmeres for å kunne kommunisere sammen. Her har det blitt brukt en ferdig åpen kildekode i Arduino, med små justeringer, for å kunne hente data fra RFID-antennen inn til mikrokontrolleren i et lesbart format.

Ved valg av både komponenter og programmeringsspråk har det blitt prioritert enheter og språk som har blitt benyttet av Sjøkrigsskolen gjennom utdanningsløpet. Dette ble gjort i den hensikt å spare tid, samt at læringen ble sentrert rundt utviklingen av selve oppgaven og ikke gjøre oppgaven vanskeligere enn det den trengte være.

1.5 Struktur

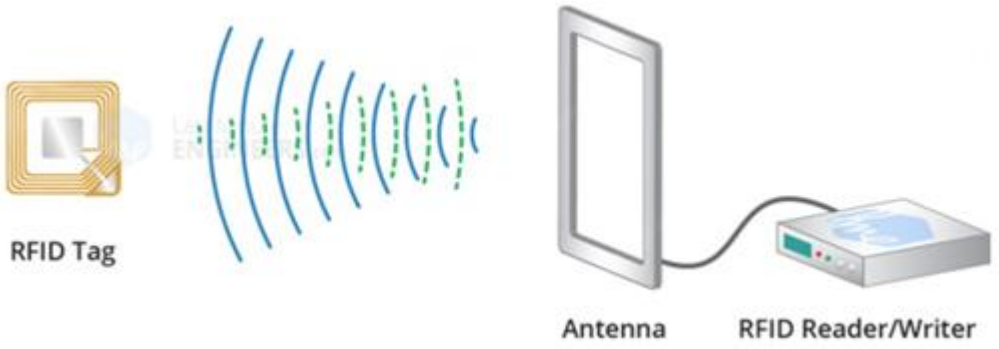
Oppgaven starter med å presentere relevant teori for å gi leseren en forståelse for hva RFID-teknologien er og brukes til. Deretter er det en kort forklaring av de forskjellige komponentene og programmeringsspråkene som har blitt brukt. Det er i liten grad lagt vekt på å gå inn på veldig tekniske detaljer eller spesifikk funksjon, men det er ment til å gi leseren av oppgaven nok forståelse av de forskjellige komponentene i oppgaven til å skjønne systemet. Videre er neste del av oppgaven en beskrivelse av produktet hvor det kommer frem mer i detalj på hvordan sluttproduktet ser ut. Her følger det med koblingsskjema og forklaringer av kode. Deretter gis en overfladisk drøfting over viktige emner som er relevant for en implementering av denne type systemer, men som for denne oppgaven ble sett bort ifra. Avslutningsvis kommer konklusjonen, bibliografi og vedlegg.

2 Teoretisk bakgrunn

Dette kapittelet vil gi leseren en overfladisk forståelse for teknologi som er anvendt i oppgaven, med det vil leseren ha et bedre grunnlag for å forstå systemet. Komponentene og programvare som er brukt er forklart hver for seg, men noen av komponentene/programmene er så nært relatert at det ville vært unaturlig og ikke beskrive sammenhengen mellom dem.

2.1 RFID teknologi

RFID er en teknologi som blir stadig mer utbredt i dagens samfunn. Det brukes i alt fra sporing av pakker, bøker på bibliotek eller varekontroll. Det finnes få grenser for denne teknologien. Et RFID system består av en brikke/chip, en antenne og en mottaker. (Figur 2.1) Chipen inneholder informasjon som brikke-id eller varenummer. Antennen til leseren sjekker kontinuerlig etter brikker. Når en brikke kommer innenfor rekkevidden til antennen etterspør antennen informasjon fra brikken, som den så sender som til antennen. En annen formulering vil være at brikken svarer på en forespørsel fra antennen til leseren. Det finnes to forskjellige typer brikker til RFID system. Det ene er en aktiv brikke, hvor RFID brikken inneholder et batteri. Dette sørger for at den er større, men også at rekkevidden blir lengre. Et eksempel på en aktiv RFID brikke er autopass-brikken som er i biler for bompassering. Det at den er aktiv gjør at den kan avleses på større avstander med større hastighet, men den har også kortere levetid. På den andre siden har en passiv brikke ikke noen form for energikilde selv. Den skaper sin egen energi gjennom induksjon når den er inne i magnetfeltet til antennen. Da får den nok energi til å kunne sende informasjonen sin. Resultatet er at en passiv brikke har kortere rekkevidde, men har derimot ubegrenset levetid.



Figur 2.1. Oversikt over et RFID system bestående av en RFID tag som leses av en antenne som er koblet til en mikroprosessor som leser/skriver data (RC522 RFID modul, 2018).

RFID teknologien opererer innenfor 3 frekvensbånd med Low Frequency (LF), High Frequency (HF) og Ultra High Frequency (UHF). LF RFID opererer mellom 30 til 300KHz. HF mellom 3 til 30 Mhz og UHF opererer mellom 300MHz til 3 GHz. UHF RFID systemer har den egenskapen at en passiv sensor kan bli avlest opptil 12 meter unna antennen, men den er mer utsatt for forstyrrelser fra for eksempel væsker eller metall. Når RFID brukes i HF-båndet er det som oftest snakk om Near field communication(NFC) teknologi som er i bruk, denne gjør dataen mye sikrere da leseavstanden er mye kortere, gjerne bare et par cm. Da brukes frekvensen 13.56MHz og dette er veldig vanlig å finne igjen i adgangskort eller bankkort (Lowry solutions, 2014).

2.2 Arduino Uno SMD

Arduino er en plattform hvor man kan både koble sammen elektroniske komponenter og sensorer. Den fysiske plattformen er en mikrokontroller som kan programmeres gjennom tilhørende programvare. Det er en åpen kilde med en bred brukermasse og godt dokumentert programvare. Arduino egner seg veldig bra for å hente data fra sensorer inn i et program eller script, for eksempel en RFID leser. Videre er Arduino kompatibel med de aller fleste operativsystem, inkludert Linux, noe som var viktig for oss. Programmeringsspråket til Arduino er i stor grad basert på C++ (Arduino(b), 2019).

Tabell 2.1: Tekniske spesifikasjoner Arduino Uno SMD (Arduino(a), 2019)

Mikrokontroller	ATmega328P
Operativ spenning	5V

Anbefalt inn spenning	5-12V
Max inn spenning	6-20V
Digitale I/O pins	14 stk (6 med PWM output)
Analoge input pins	6 stk
DC strøm per I/O pin	20mA
DC strøm for 3.3V pin	50mA
Minne	32 KB
SRAM	2KB
EEPROM	1KB
Klokke hastighet	16MHz
Led_builtin	13

2.3 Raspberry Pi 3 model B+

Raspberry Pi er en liten datamaskin som kan kjøre enkle programmer som ikke krever mye prosessorkraft. Den krever lite energi og baserer seg på operativsystemet Linux. Det er et veldig billig og mye brukt datamaskin, den er veldig vanlig innenfor hobby programmering eller mindre prosjekter. Dette skaper en stor brukermasse og derfor sørger for at deres åpne kildekode er svært godt utprøvd og robust.

Tabell 2.2: Tekniske spesifikasjoner Raspberry Pi 3 modell B+(Raspberry Pi, 2019)

Prosessor	Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4GHz
Minne	1GB LPDDR2 SDRAM
Tilkobling	o 2.4GHz, 5GHz IEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE

	○ Gigabit Ethernet over USB 2.0 ○ 4 x USB 2.0 porter
Andre tilkoblinger	40-pin GPIO header
Video og bilde	○ 1 x HDMI ○ MIPI DSI display port ○ MIPI CSI camera port ○ 4 pole stereo output and composite video port
Multimedia	H.264, MPEG-4 decode (1080p30); H.264 envode(1080p30); OpenGL ES 1.1,2.0graphics
SD kort support	Micro SD format for å laste operativsystem og datalagring
Strømforsyning	○ 5V/2.5A DC via micro USB ○ 5V DC via GPIO header ○ Strøm over Ethernet
Klima	Opererer fra 0 – 50 grader Celsius

2.4 Node-RED

Er et Java Script basert programmeringsverktøy som baserer seg på blokkprogrammering på en webplattform. I Node-RED kan man enkelt koble sammen komponenter som baserer seg på forskjellige dataprotokoller. Da Node-RED også baserer seg på åpen kildekode finner man enkelt frem til ferdig designede biblioteker/paletts som inneholder noder eller blokker som gjør mye av formateringen av data automatisk, og man trenger i stor grad bare å sende informasjonsstrømmen til riktig plass (Node-RED, 2019).

2.5 MySQL

MySQL er et program for å administrere databaser. I en database kan man enkelt organisere informasjon i forskjellige tabeller for å minimere bruken av lagringsplass. Gitt at databasen er utformet bra vil det sørge for at det ikke er noen dobbeltlagring av informasjon. Er det et sted en datamaskin virkelig overgår oss mennesker er det på datakraft eller

evnen til å gå gjennom informasjon, og ved bruk av databaser kan man virkelig få utnyttet denne styrken. Gjennom programmet MySQL kan man legge til, fjerne og endre informasjon som ligger i en database. Structured Query Language (SQL) er et veldig utbredt databasespråk i verden i dag.

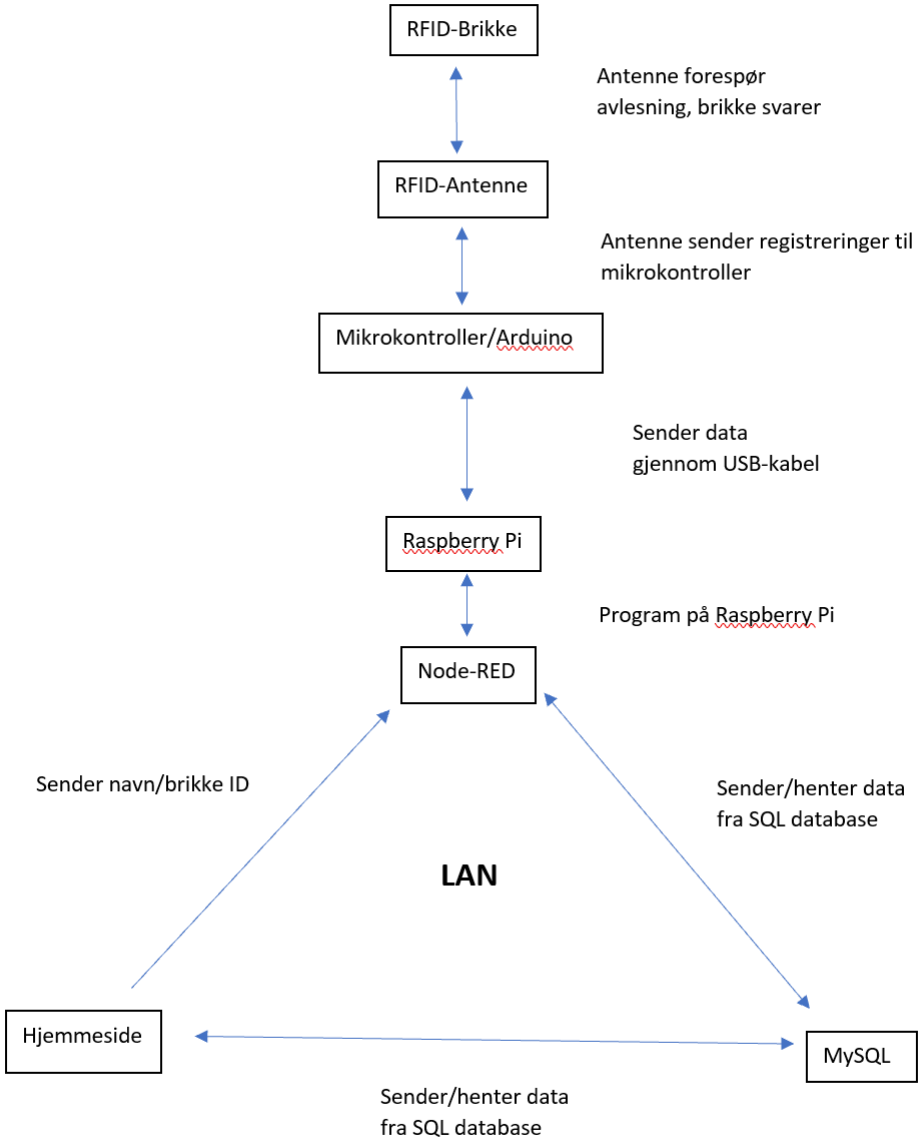
2.6 HTML

Programmeringsspråket HyperText Markup Language (HTML) er spesielt egnet til å designe websider. Gjennom å opprette en webserver henter du HTML dokumentene du har lagret lokalt for å få frem designet på nettsiden. En HTML-nettside baserer seg på mange HTML elementer eller blokker og språket brukes primært for å fremstille det visuelle på nettsiden.

2.7 PHP

Et annet programmeringsspråk som er veldig relevant i denne bacheloren er Hypertext Preprocessor (PHP). Med PHP kan man enkelt kjøre script eller lettere sagt gjennomføre handlinger i programmet. PHP og HTML er to språk som snakker veldig bra sammen. Og i en HTML-kode er det veldig enkelt å kalle på et PHP-script til å gjennomføre handlinger basert på forskjellige kriterier. En veldig forenklet, men grei måte å se det på er at HTML er designet, mens PHP står for dynamisk innhold på siden.

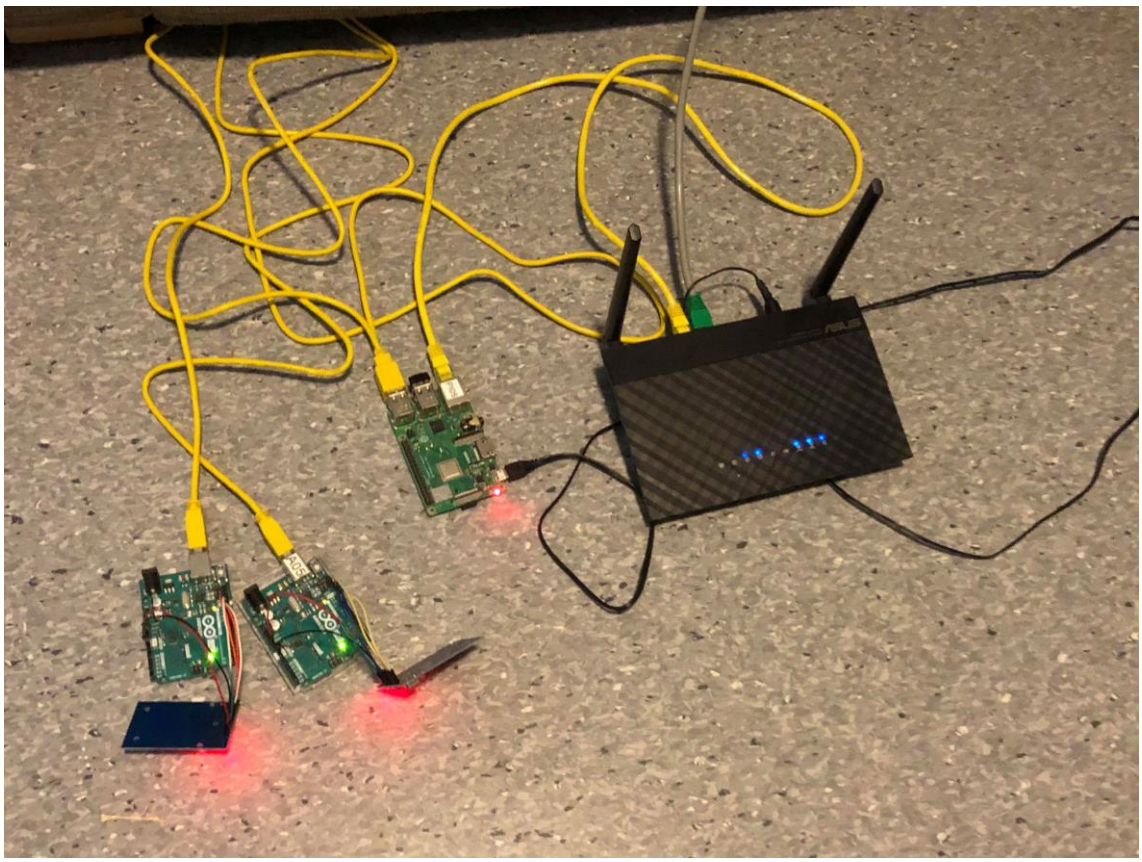
3 Produktbeskrivelse



Figur 3.1 Oversikt over systemets datastrøm

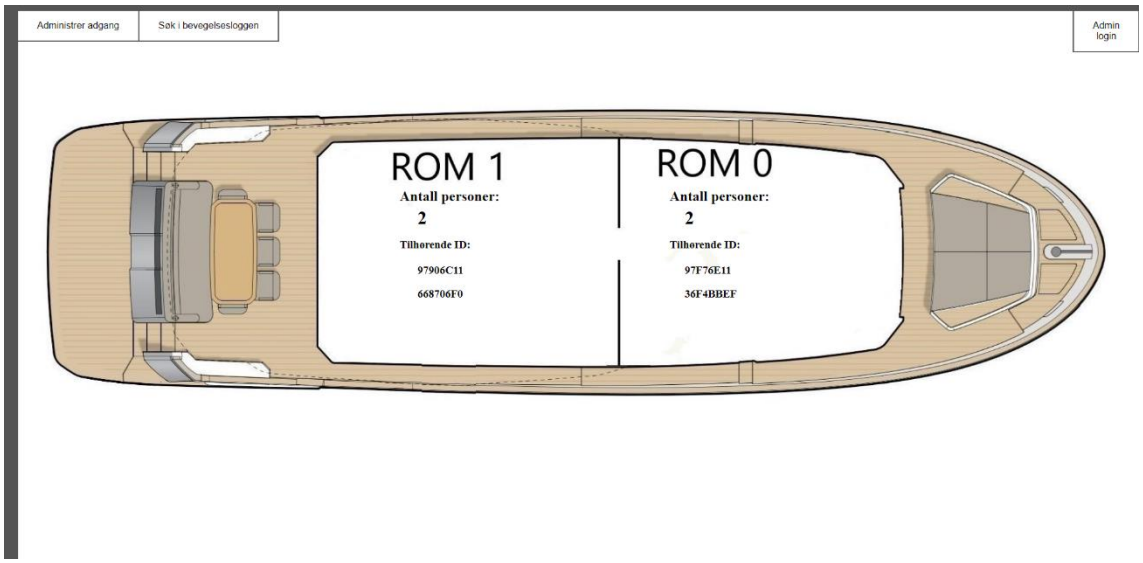
Systemet baserer seg på fire hovedkomponenter som samhandler på en således måte at det utgjør et fullverdig personellkontrollsystem med et par kjernefunksjoner. Disse funksjonene innebærer følgende: En åpen nettside som alle kan koble seg på, der man kan se hvilke brikker og tilhørende ID-nummer, samt det totale antallet, som befinner seg hvor til enhver tid. Dette presenteres grafisk på nettside, med en oversikt over alle rom i systemet (Figur 3.3). En kan også søke i en logg som vil vise hvem som har beveget seg hvor i det tidspunktet det har blitt søkt (Figur 3.28). Videre vil kort som ikke er registrert ikke bli lest av programvaren, derav er det og lagt inn funksjoner som lar alle brukere med

fysisk tilgang til kort som tilhører systemet å både legge til og fjerne kort, og tilhørende navn, fra systemet.



Figur 3.2 Hardware komponentene brukt i oppgaven. To antenner koblet i hver sin Arduino som igjen er koblet til en Raspberry Pi. Systemet er koblet til et lokalt LAN nettverk gjennom en ruter.

Den siste hovedfunksjonen er muligheten til å logge seg inn på siden, og få tilgang til hvilke personer, gitt ved navn og ikke bare ID-nummer, som befinner seg hvor. Denne funksjonen er altså ikke tilgjengelig for alle brukere. Funksjonen er tiltenkt brukt under krisesituasjoner der sikkerhetsbehovet til enkeltmennesket er viktigere enn retten til privatliv. Dette kan eksempelvis være ved en evakuering, da vil det være essensielt å forsikre seg om at alle er gjort rede for, og dersom dette ikke er tilfellet kan man bruke denne innloggingen til å finne ut hvem som er hvor, og ikke bare hvor det er mennesker (Figur 3.27).



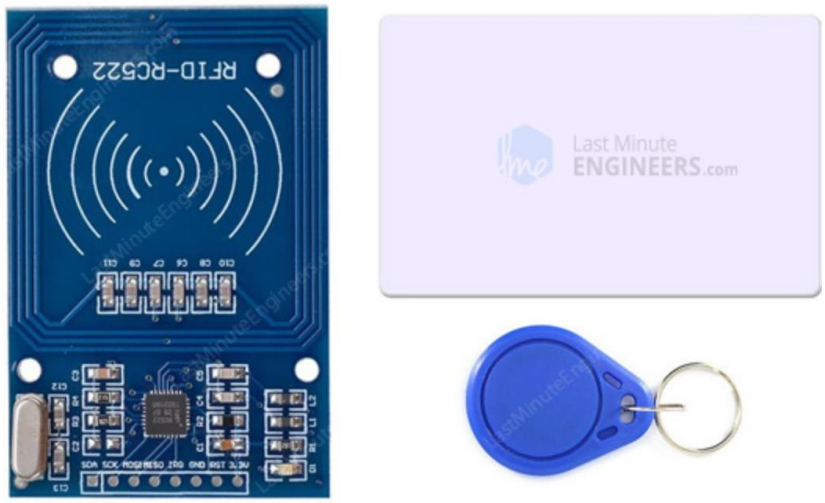
Figur 3.3 Skjermdump av hjemmeside

Ved intensjonelle forsøk på å misbruke systemet er dette mulig, om du innehar kompetansen til å gjøre det. Systemet i sin helhet er ment til å kjøre på et lokalt nettverk, Local Access Network (LAN), som igjen innebærer at et angrep på systemet må gjøres lokalt. Hele systemets HMI er basert på PHP-scriptede sider, og er derav sårbart for eksempelvis SQL-injections. Dette er selvsagt en risiko, derimot kan en argumentere for at sannsynligheten for at dette skjer er liten med tanke på at systemet kun er laget for å kjøre lokalt. Selv om konsekvensen for systemets operabilitet ved en slik hendelse kan være katastrofal.

Avgjørelsen om å ha så få funksjoner som mulig bak en innloggings-mur har vært en rød tråd gjennom hele oppgaven. Grunnen til dette er ganske enkel, og det er det faktum at systemet kjøres lokalt som ble tungen på vektskålen. Filosofien blir at alle som overvåkes, ved korrekt opplæring, har tilgang til flest mulig av systemets funksjoner. Dette har som hensikt å minske frykten for overvåkning dersom alle involverte vet hva systemet faktisk gjør. På samme måte som angrep på systemet må komme innenfra, vil også misbruk av systemet måtte komme innenfra. Dersom dette inntreffer kan en spørre seg om det er det faktum at systemet krever en høy grad av tillit hos brukerne for å fungere, eller det faktum at brukerne av systemet har misbrukt det, som er problemet. Det er lagt inn sikkerhetstiltak i programmeringskoden for å hindre utilsiktete feil, men med kunnskap, vilje og teknisk kompetanse er ikke systemet sikkert for intensjonelle angrep.

3.1 Hardware

En av første avgjørende valgene som måtte gjøres var å velge hva slags type hardware som skulle brukes i oppgaven. Da det er mange forskjellige varianter av RFID var det viktig at det ble tatt et reflektert og informert valg. Utstyret som var aktuelt varierte veldig i pris ut ifra hvilken teknologi den baserte seg på. Det ble vurdert ut ifra blant annet rekkevidde, bølgelengde og pris da komponenter ble undersøkt. Ettersom det var blitt besluttet at softwaren og fremstillingen av personers posisjon skulle være hovedfokuset, så ble det bestemt å ikke kjøpe dyre komponenter med mange flere teknologiske muligheter enn det som det var hadde behov for. Det ble besluttet å anskaffe RC522 RFID moduler som er kompatible med mikrokontrolleren Arduino. Valget av mikrokontroller falt derav på Arduino, som skolen kunne stille med. Mikrokontrolleren ble igjen koblet til Raspberry Pi som til oppgaven var mulig å låne av skolen.

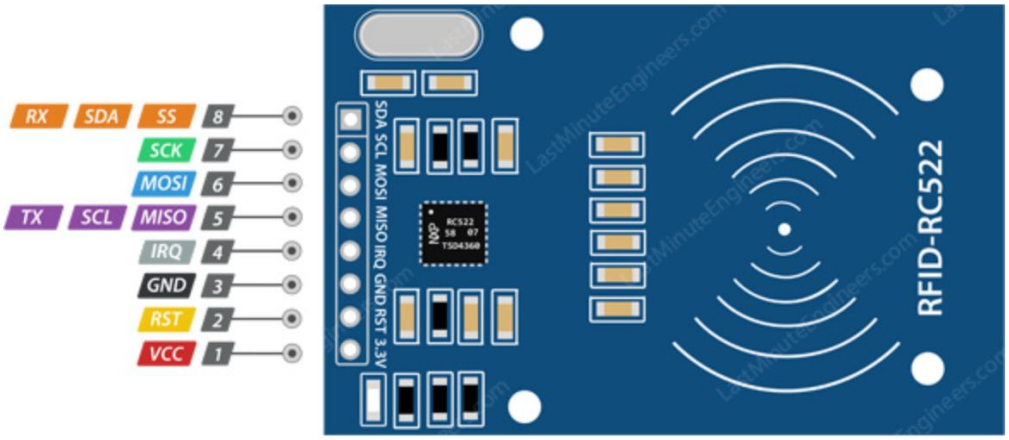


Figur 3.4 RC522 RFID modul sett

RC522 RFID modul består av en antenne med tilkoblet sender/mottaker og to forskjellige brikker til avlesning, et kort og en chip. Modulen opererer med en frekvens på 13.56 Mhz og kan lese alle RFID brikker med standarden ISO 14443A. Leseren kan kobles sammen med en mikrokontroller, se figur 3.3 og har en maksimal overføringshastighet på 10Mbps (RD522 RFID Modul, 2018).

Tabell 3.1: Tekniske spesifikasjoner for RC522 (RD522 RFID Modul, 2018)

Frequency Range	13.56 MHz ISM Band
Host Interface	SPI / I2C / UART
Operating Supply Voltage	2.5 V to 3.3 V
Max. Operating Current	13-26mA
Min. Current(Power down)	10µA
Logic Inputs	5V Tolerant
Read Range	5 cm

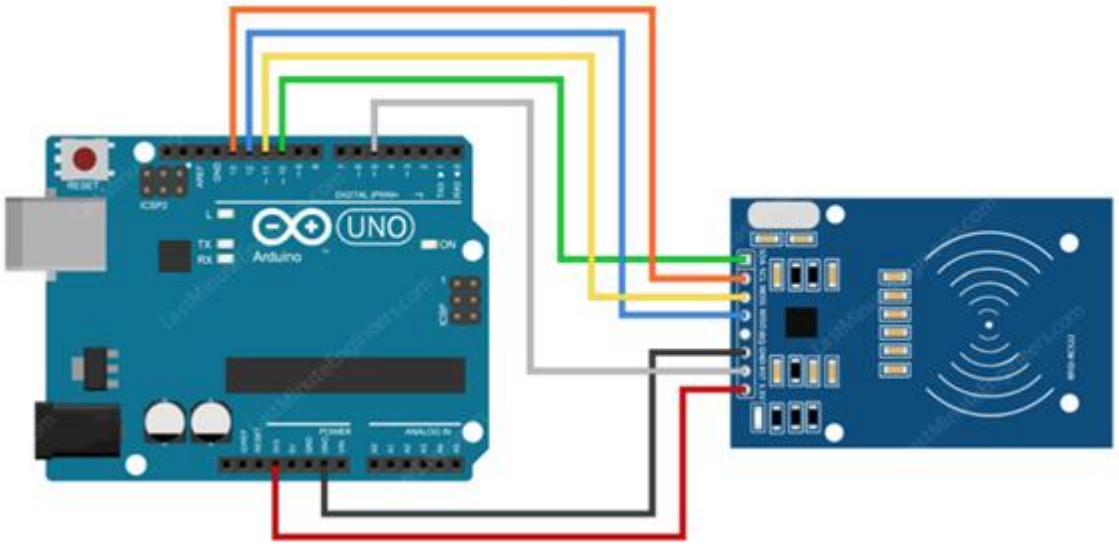


Figur 3.5 RC522 tilkoblingspunkter (RC522 RFID Modul, 2018)

Modulen kommuniserer med mikrokontroller gjennom 8 pins. Her er en kort forklaring av hvert tilkoblingspunkt:

Tabell 3.2: Tilkoblinger RFID leser (RC522 RFID Modul, 2018)

VCC	forsyner modulen med strøm, som man kan lese i Tabell 3.1 kan inn spenningen variere fra 2.5 til 3.3 volt. Høyere spenning kan ødelegge modulen.
RST	styrer om modulen skal koble til eller fra alle tilkoblingspunktene. Hvis inn signalet er lavt vil modulen koble fra alle tilkoblingspunkter eller, hvis signalet er høy vil da modulen koble til alle tilkoblings-punkter igjen.
GND	er jordingspunktet.
IRQ	sender signal til microkontrolleren når en RFID brikke blir lest av leseren.
MISO/SCL/TX	oppfører seg som en master-in-slave-ut for SPI grensesnitt.
MOSI	er master-ut-slave-in for SPI grensesnitt.
SCK	aksepterer klokkepulser fra SPI bus fra f.eks Arduino.
SS/SDA/RX	er signal inngang for SPI grensesnitt.



Figur 3.6: Koblingsskjema for RC522 RFID med Arduino Uno (RC522 RFID Modul, 2018)

Koblingen mellom RC522 RFID modul og Arduino er noe som ligger ute som åpen kildekode, og har i sin helhet blitt hentet fra nettet, den fysiske kablingen er vist på figur 3.6 (RC522 RFID modul, 2018). Programmeringskoden er gjort ved å modifisere en åpen kildekode som du kan lese mer om i avsnitt 3.2.1. Koblingen mellom Arduino og Raspberry Pi er gjort med USB kabel.

Ettersom valget falt på RFID brikken RC522 krevde denne bruk av en mikrokontroller, som i denne oppgaven er en Arduino UNO (Figur 3.6). Denne kan kobles direkte inn i en Windowsbasert datamaskin, og derav kunne bruken av Raspberry Pi(Figur 3.7) vært kuttet. Derimot talte to hovedfaktorer for å bruke en sekundær datamaskin i systemet. Den første er størrelse. En windowsbasert datamaskin i samme størrelsesorden ville havnet i en helt annen prisklasse. Den andre faktoren er mulighet for skalering. En datamaskin kan realistisk sett kun dekke nok sensorer for ett rom. Grunnen til dette er at samtlige av RFID-antennene krever en mikrokontroller hver. Om en da skulle basere seg på å koble til mer enn fire mikrokontrollere via USB-porter til en datamaskin på størrelse med en Raspberry kan fort føre til at systemet blir ustabilt.

Figur 3.7 Raspberry Pi 3 modell B+(Raspberry Pi, 2019)

Raspberry Pi kjører Node-RED som system for å sende data mellom RFID-antennen og SQL-databasen. Dette er gunstig med tanke på skalering, da flere ulike Raspberry Pi som kjører hver sin Node-RED kan alle skrive til samme database. Dermed er bruken av Raspberry med på å gjøre skalering av systemet enklere, og billigere enn om man skulle brukt en tilsvarende Windowsbasert datamaskin.

3.2 Programmering

Oppgaven baserer seg i all hovedsak på javascript i Node-RED, HTML og PHP til websiden, samt C++ til å programmere hvordan mikrokontrolleren, Arduino, skal funge-re. Videre har det blitt nyttet SQL til alt av databaser som oppgaven bruker.

3.2.1 Arduino kode

Oppgaven til alle Arduino UNO som er koblet til systemet er å registrere hvorvidt en brikke har blitt presentert for leseren eller ei. Dersom et kort presenteres har den og som

oppgave å skrive tilhørende ID til brikken gjennom USB til Raspberry Pi og Node-RED. Det eksisterte allerede et bibliotek til Arduino Integrated Development Environment (IDE) til den samme antennen som er brukt i oppgaven, som er produsert av brukeren miguelbalboa på GitHub (Arduino åpen kildekode, 2019). Samme bibliotek, av samme bruker, er og å finne på Arduino sine egne hjemmesider. På begge lokasjoner er det lagt ut som åpen kildekode.

Derimot gjorde ingen av scriptene i koden nøyaktig det som oppgaven vår krever. Resultatet ble at bibliotekene som er byggesteinene til scriptene forble uforandret, men selve scriptet er modifisert til å passe kravene som oppgaven stiller.

```
#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN         9
#define SS_PIN          10

MFRC522 mfrc522(SS_PIN, RST_PIN);

MFRC522::MIFARE_Key key;

void setup() {
    Serial.begin(9600);
    while (!Serial);
    SPI.begin();
    mfrc522.PCD_Init();
}

void loop() {
    if ( ! mfrc522.PICC_IsNewCardPresent())
        return;
0
    if ( ! mfrc522.PICC_ReadCardSerial())
        return;

    dump_byte_array(mfrc522.uid.uidByte, mfrc522.uid.size);
    Serial.println();




}

void dump_byte_array(byte *buffer, byte bufferSize) {
    for (byte i = 0; i < bufferSize; i++) {
        Serial.print(buffer[i] < 0x10 ? " 0" : " ");
        Serial.print(buffer[i], HEX);
    }
}
```

Figur 3.8 Skjermdump av Arduinokode

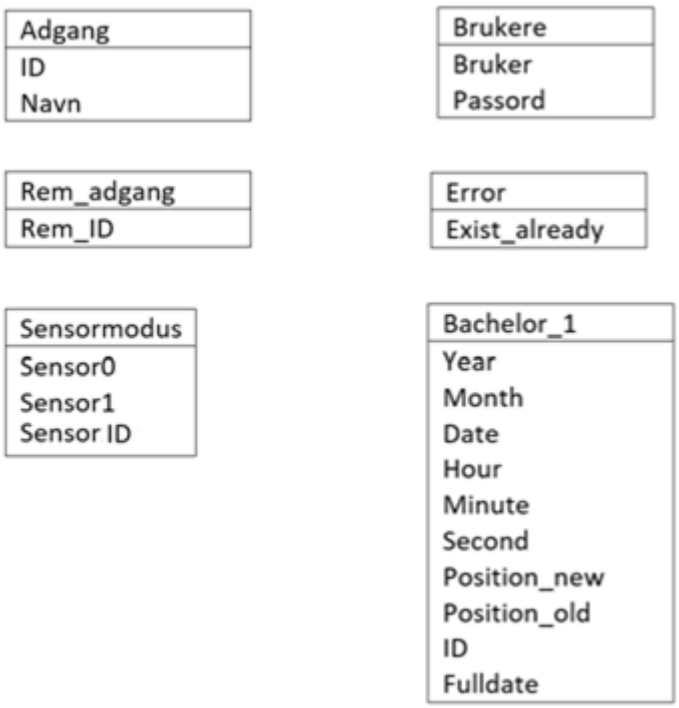
Skjermdumpen i Figur 3.8 viser selve koden som er lastet opp på alle Arduino som er koblet til RFID antennene. Dette er en modifisert kode, som i deler kan hentes fra tidligere nevnt bibliotek.

Det er i hovedsak tre løkker som henter, og fremstiller alle data som kommer fra Arduino. Void setup kjøres en gang for å starte kommunikasjonen både med antennen, og Raspberry Pi. Void loop er løkken som faktisk sjekker om et kort er tilstede, og andre del av løkken er den som leser kortet om det skulle være tilstede. Siste løkke, void dump_byte_array, er løkken som sender strengen med ID til Raspberry, og Node-RED

For utenom endringene gjort i skjermdumpen til venstre er koden tilhørende Arduino i sin helhet hentet fra en åpen kilde, og beskrives derfor ikke mer utfyllende.

3.2.2 MySQL

For å kunne lagre og behandle dataene ble det ansett at det som var den mest effektive løsningen ville være bruk av databaser. Sjøkrigsskolen underviser i MySQL, noe som gjorde at valget av leverandøren av SQL tjenester enkel. Databasen er blitt endret ganske mye i løpet av arbeidet med oppgaven, men Entity Relations (ER) diagrammet (Figur 3.9) for databasen Bachelor endte til slutt som dette.



Figur 3.9 ER-diagram for databasen Bachelor

Databasen består av seks tabeller, og de utfører fire forskjellige funksjoner. To av tabellene, Rem_adgang og Error, brukes av PHP scriptet til å fremvise en feilmelding på hjemmesiden. Hvis det er noen logiske motsetninger i det en bruker prøver å utføre vil Node-RED sende inn en verdi i en av disse tabellene som da trigger et PHP-script. Disse tabellene har ingen annen funksjon enn å bidra i feilmeldinger. Tabellen Brukere inneholder brukernavn og passord for administratorer. Det er kun systemansvarlige som har tilgang til å endre denne databasen, da det må gjøres direkte i tabellen. Tabellene Bachelor_1 og Adgang, den førstnevnte vil være selve loggen til alle bevegelsene på fartøyet og den andre en liste over brikke-id med tilhørende eier som er blitt registrert. Det er disse to tabellene som er de mest brukte i oppgaven. De brukes til alt fra å fremvise posisjon, sjekke om brikke-ID allerede er registrert eller om navnet allerede er knyttet til en brikke-

ID. Den siste tabellen Sensormodus justeres av brukeren manuelt gjennom hjemmesiden. Her kan brukeren bestemme at en sensor skal lese, slette eller registrere et nytt kort. Dette vil endre verdien i kolonnen til sensoren som er valgt. Denne verdien hentes så fra databasen for å avgjøre hvordan programmet responderer på en avlesning av RFID-brikke.

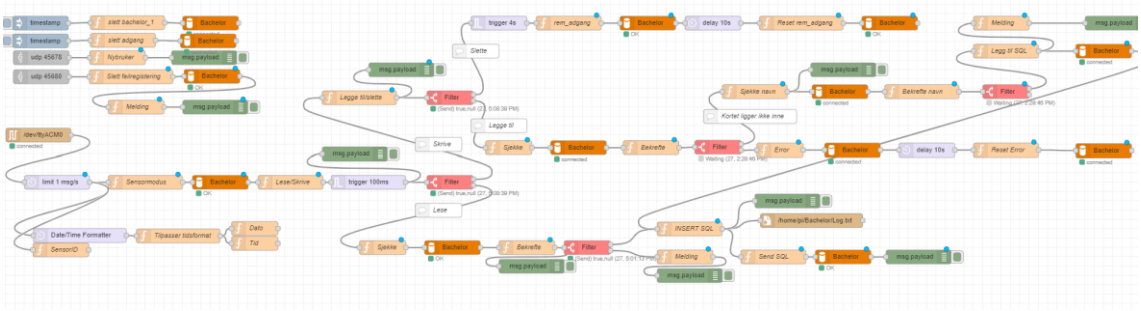
Year	Month	Date	Hour	Minute	Second	Position_new	Position_old	ID	fulldate
2019	11	28	18	3	22	1	1	97906C11	20191128180322
2019	11	28	18	3	43	0	0	97F76E11	20191128180343
2019	11	29	11	11	10	1	1	36F4BBEF	20191129111110
2019	11	29	11	11	11	0	0	36F4BBEF	20191129111111
2019	11	29	11	11	13	1	1	668706F0	20191129111113
2019	11	29	11	11	14	0	0	668706F0	20191129111114
2019	11	29	11	11	17	1	1	97906C11	20191129111117
2019	11	29	11	11	18	1	1	97F76E11	20191129111118
2019	11	29	11	11	20	1	1	97F76E11	20191129111120
2019	11	29	11	11	23	0	0	36F4BBEF	20191129111123
2019	11	29	11	11	25	0	0	668706F0	20191129111125
2019	11	29	11	11	26	0	0	668706F0	20191129111126
2019	11	29	11	11	27	0	0	668706F0	20191129111127
2019	11	29	11	11	29	1	1	97F76E11	20191129111129
2019	11	29	11	11	29	0	0	97F76E11	20191129111129

Figur 3.10 Skjermdump av tabellen Bachelor_1

For å alltid kunne presentere de nyeste avlesningene, og kun de nyeste avlesningene, til de registrerte brikkers ID har det vært en omfattende prosess og revidering av tabellene for å klare og få en nøyaktig nok spørring.

3.2.3 Node-RED kode

Oppgaven benytter Node-RED til å behandle, formatere og videresende data til en SQL-database som har blitt laget til oppgaven. Programkoden i Node-RED er ganske omfattende, og vil her koden som gjelder for en enkelt leser bli forklart. En tilsvarende kode eksisterer for hver leser i systemet. Det viste seg å være veldig vanskelig å få Node-RED til å kommunisere direkte med PHP, derfor kommuniserer Node-RED utelukkende med MySQL. Kort fortalt gjør Node-RED endringer i SQL-databasene som igjen forårsaker handlinger i PHP-script.

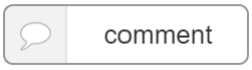


Figur 3.11: Node-RED kode for en RFID leser. De viktigste delene av koden blir forklart senere i avsnittet.

Node-RED programmet består av ulike blokker som er koblet sammen. De forskjellige blokkene utfører hver sine funksjoner. Her er en oversikt over de nodene som har blittbenyttet i programmet.

Tabell 3.3: Noder brukt i Node-RED

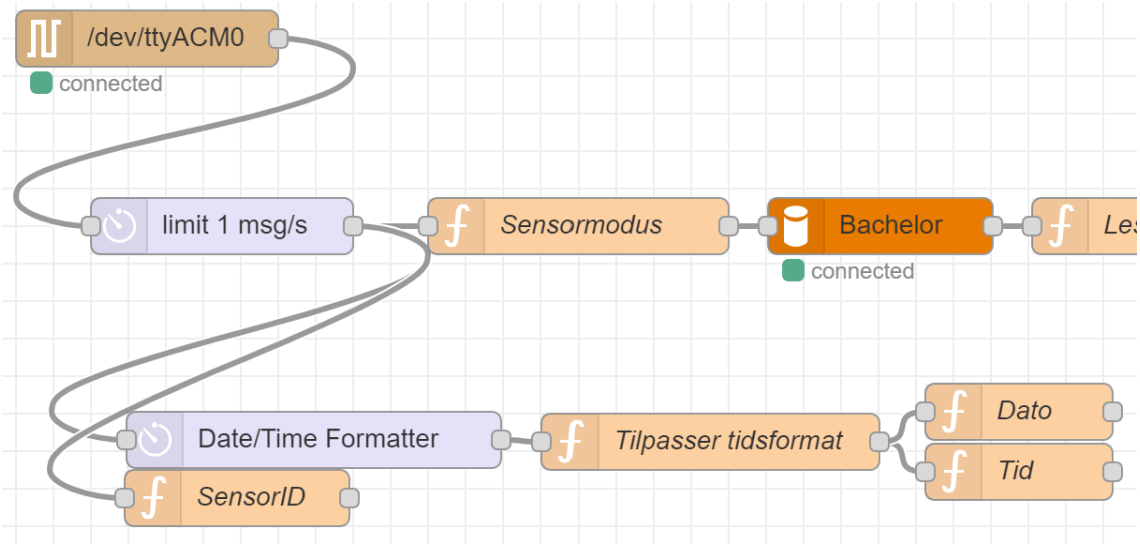
Type	Bilde	Beskrivelse	Palette	Versjon
Inject	inject	Sender et signal videre til andre noder.	Node-red	1.0.2
Function	function	En Javascript funksjonsnode.	Node-red	1.0.2
MySQL	mysql	Kobler Node-RED opp mot en SQL-database	Node-red-node-mysql	0.0.19
Udp in	udp in	Får datainput gjennom UDP	Node-red	1.0.2
Debug	debug	Viser verdien til signalet.	Node-red	1.0.2
Moment	moment	Konverterer dato/tid til et objekt eller en tekst.	Node-red-contrib-moment	3.0.3

Comment		Kommentar, ingen funksjon.	Node-red	1.0.2
Filterultimate		Sender utsignalet videre til en av to lokasjoner avhengig av innsignalet.	Node-red-contrib-boolean-logic-ultimate	1.0.11
Trigger		Omgjør signal til en puls.	Node-red	1.0.2
File in		Sender meldingen til en fil.	Node-red	1.0.2
Serial in		Leser data fra en usb port i Raspberry pi.	Node-red-node-serialport	0.9.1
Delay		Utsetter signalet/meldingen	Node-red	1.0.2

Når det leses av en RFID-brikke blir det sendt et lesersignal i Serial in noden, se Figur 3.12. Videre begrenses avlesningshastigheten til en leser til å være 1Hz før dataen blir behandlet i resten av Node-RED. Dette er en forsinkelse som har blitt lagt inn i den hensikt å begrense antall avlesninger av samme kort til loggen. En senere SQL spørring avhenger av å ha ulike registreringstider for hver av RFID-brikkene som blir registrert. Dette gjøres for å få vist dataen riktig på nettsiden.

Det første som skjer, se Figur 3.12, er at programmet gjennom en serie med funksjonsnoder oppretter globale variabler for tidspunktet avlesningen skjedde og i hvilken leser som sendt avlesningen til Node-RED. Her lages det globale variabler for år, måned, dag, time,

minutt, sekund og sensorID. Disse blir så brukt senere i Node-RED til å legge inn i SQL eller kjøre SQL spørringer.

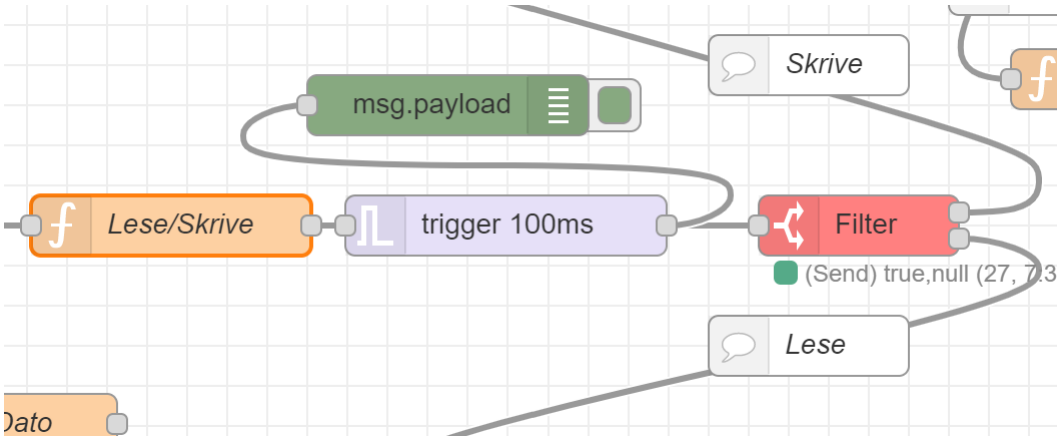


Figur 3.12 Node-RED mottak av leserdata

```
var myStr = msg.payload
msg.payload= myStr.replace(/ /g, '')
msg.payload = msg.payload.substr(0, 8)
global.set("ID0", msg.payload);
var newentry = {};
newentry.topic = ("SELECT Sensor0 FROM sensormodus WHERE ID='A'")
newentry.payload = '';
return newentry
```

Figur 3.13 Funksjonsnode Sensormodus

Videre viser Figur 3.13 et bilde av javakoden i funksjonsnoden Sensormodus, se Figur 3.12 her sjekkes det hvilken status RFID-leseren er satt i og det blir hentet fra MySQL gjennom en SQL-spørring som sendes til en SQL-node. Det er viktig for programmet å vite hvilken sensormodus leseren er i da dette har stor påvirkning for hvordan dataen skal behandles og rutes senere. I praksis bestemmer det om programmet skal lese av og fremvise posisjonen til RFID-brikken eller om det skal slette eller legges til en bruker. Funksjonsnoden sensormodus tar også gjennom sine fire første linjer med kode og henter nødvendig informasjon fra datastrengen som blir lest av RFID-brikken.

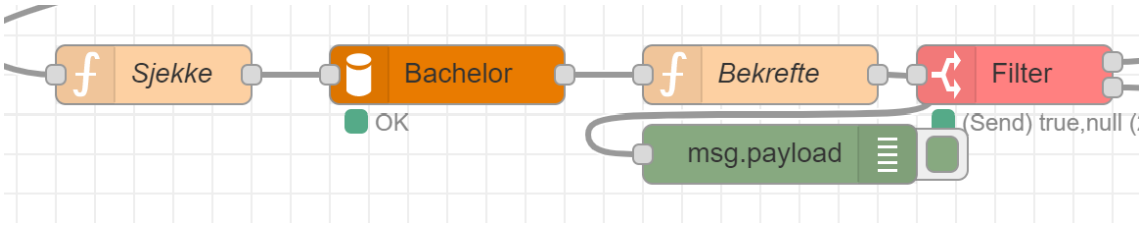


Figur 3.14 Skrive eller lese av kort

```
1  global.set("filtervalg", msg.payload[0].Sensor0)
2  var modus = msg.payload[0].Sensor0
3  if (modus == 0) //lesemodus
4  {
5      msg.payload = false
6  }
7  else //skrivemodus
8  {
9      msg.payload = true
10 }
11 global.set("velger0", msg.payload)
12 return msg
```

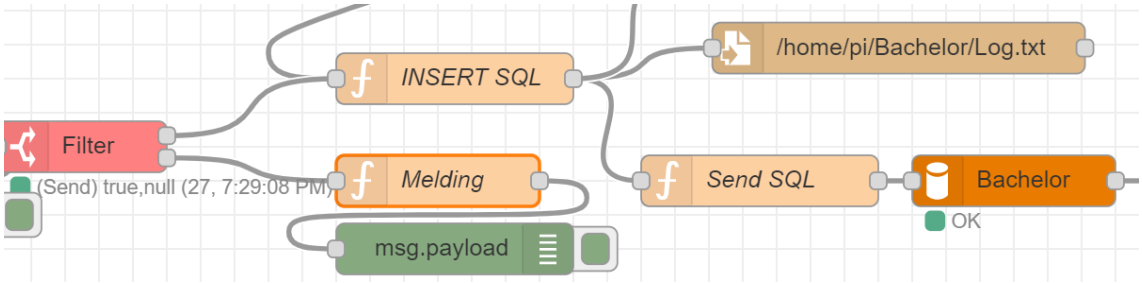
Figur 3.15 Lese/Skrive node

Etter at sensormodus er hentet i fra SQL-databasen møter kommer første ruting av datastrømmen i Node-RED koden. Her bestemmes det om datastrømmen skal rutes den ene eller den andre veien avhengig av verdien til sensormodus. Hvordan dette gjøres kan dere se av javakoden i Figur 3.13. Basert på verdien til variabelen «velger0», se Figur 3.15, vil filternoden rute datastrømmen til «skrive» gitt input lik TRUE, eller «lese» gitt input lik FALSE. Figur 3.14 viser en av triggernodene i oppgaven. Den er lagt inn som en ekstra sikkerhet for at ikke systemet skal få for mye data å behandle på en gang. Da prosessorkraften til en Raspberry Pi er begrenset er det lagt inn flere slike sikkerhetstiltak for å hindre at programmet krasjer.



Figur 3.16 Kontroll av kort som skal leses

Når en person som er registrert i systemet beveger seg rundt på fartøyet blir brikken lest når han passerer en leser. Da blir dataen rutet videre i lese-sløyfen i koden. Her blir brikke-ID sjekket opp mot databasen om brikke-ID finnes registrert i databasesystemet som en aktiv RFID-brikke. Dette blir igjen hentet gjennom en SQL-spørring i funksjonsnoden «Sjekke», se Figur 3.16. Videre blir resultatet fra funksjonsnoden «Sjekke» brukt til å avgjøre outputen fra funksjonsnoden «Bekrefte». Om brikke-ID ikke er registret i databasen returnerer «Bekrefte» false og datastrømmen blir rutet til en ny funksjonsnode som sender meldingen «Feil. Kortet ligger ikke i databasen. Kontakt systemansvarlig.» til nettsiden. Derimot om outputen er TRUE vil filteret, se Figur 3.16, sende videre i programmet.



Figur 3.17 Leser inn i SQL

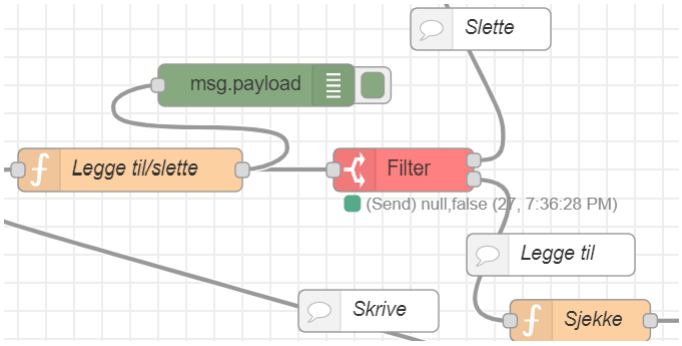
Tidligere i Node-RED har nå RFID brikken blitt kontrollert, validert og godkjenner nå alle kravene til at lesingen kan legges til i SQL-databasen og da dukke opp på den digitale fremvisningen på nettsiden. Når lesingen har passert alle filtrene som ligger inne kommer det et signal inn i funksjonsnoden «INSERT SQL», se Figur 3.17 og Figur 3.18.

```
global.set("Posgam0", global.get("Posny0"))
global.set("Posny0", global.get("SensorID0"))
var ID = global.get("ID0")
var SensorID0 = global.get("SensorID0")
var Hour = global.get("Hour0")
var Minute = global.get("Minute0")
var Second = global.get("Second0")
var Year = global.get("Year0")
var Month = global.get("Month0")
var date = global.get("Date0")
var Posny = global.get("Posny0")
var Posgam = global.get("Posgam0")
msg.payload = 'INSERT INTO bachelor_1 (Year, Month, Date, Hour, Minute, Second, Pos
global.set("Datastreng0", msg.payload)
return msg
```

Figur 3.18 Java kode i funksjonsnode "INSERT SQL"

Noden som heter «INSERT SQL» henter all data som tidligere har blitt lagt inn i globale variabler og disse brukes til å lage en output som er en SQL-INSERT spørring hvor all dataen som trengs blir lagt inn. Hele SQL-spørringen er tilpasset tabellen «bachelor_1» og siden har vært begrenset antall avlesninger en leser kan gjøre til en lesing pr sekund vil det ikke kunne oppstå to identiske rekker i SQL-tabellen. Den siste funksjonsnoden «Send SQL» som du ser i Figur 3.17 sender datastrengen inn i MySQL via en ny SQL-node.

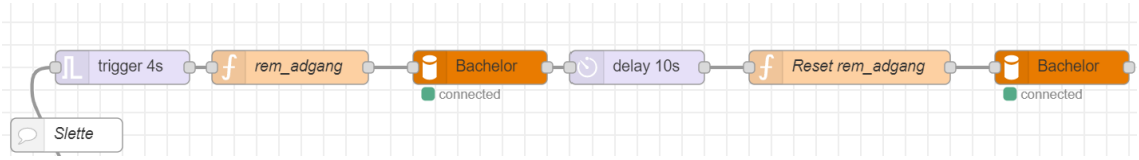
Dersom det skulle være behov for å legge til eller fjerne en bruker vil filtreringen sørge for at dataen blir behandlet på en annerledes måte. Dersom dataen i filtreringsnoden, som vist på figur 3.14, hadde fått en verdi med input TRUE så hadde datastrømmen blitt rutet til å skrive endringer i adgangen i databasen istedenfor å bare skrive til bevegelsesloggen.



Figur 3.19 Filtrering av Legge til/slette

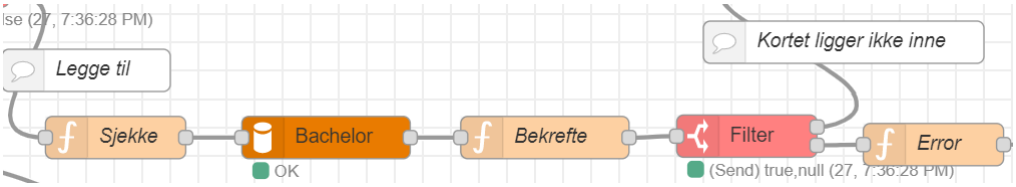
Funksjonsnoden «Legge til/slette» sjekker verdien til en variabel som ble definert tidligere i Node-RED i noden «Sensormodus», se Figur 3.12. Ut ifra verdien til denne variabelen vet programmet om brukeren har valgt å slette en bruker med tilhørende brikke-ID

eller om det ønskes å legge til en ny bruker på et nytt kort. Filteret ruter da datastrømmen til riktige funksjoner videre.



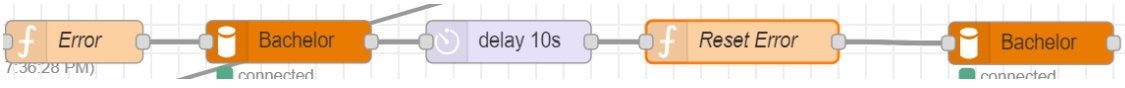
Figur 3.20 Rekken med noder som sletter et kort

Signalet går inn i en triggernode som hindrer nye signaler fra starte funksjonene for 4 sekunder. Dette er lagt inn for at nettsiden skal få oppdatert seg, så endringen synes visuelt på nettsiden før man får gjort andre endringer fra samme leser. Videre i funksjonsnoden «rem_adgang», se Figur 3.20, legges den registrerte brikke-ID inn i en hjelpetabell, som heter rem_adgang, i databasen. Etter en forsinkelse på ti sekunder gjør den neste funksjonsnoden «Reset rem_adgang» at tabellen rem_adgang blir tømt igjen. Selve sletningen av kort gjøres ikke i Node-RED, men gjennom PHP delen av oppgaven før tabellen rem_adgang blir tømt. Når et kort blir slettet blir alle linjer i alle tabellene i databasen som inneholder brikkens ID fjernet. PHP slettingen av kort er forklart bedre i kapitel 3.2.3 PHP.



Figur 3.21 Legge til brikke, bekrefte at ID ikke er i systemet

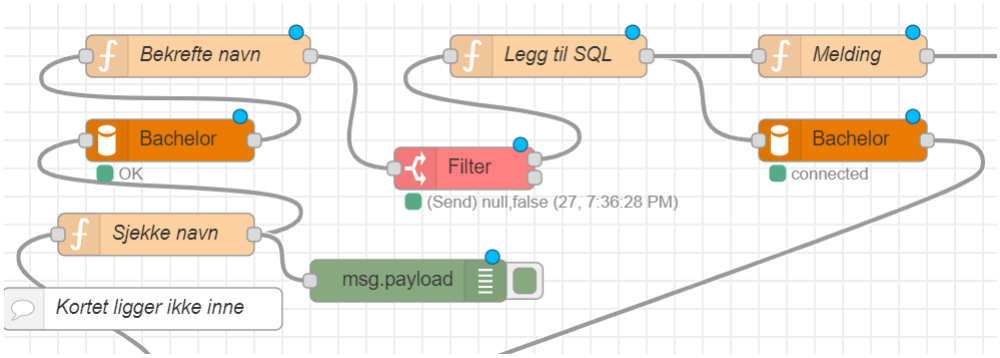
I funksjonsnoden «Error» legges det til en verdi i en hjelpetabell kalt «error» i databasen. Etter 10 sekunder gjør funksjonsnoden «Reset Error» at denne tabellen blir tom igjen. Verdien som blir lagt inn i tabellen Error brukes av PHP programmet til å sende ut en feilmelding som fremstilles visuelt på nettsiden.



Figur 3.22 Legge inn kort. Error det eksisterer allerede et kort med denne ID

I funksjonsnoden «Error» legges det til en verdi i en hjelpetabell kalt «error» i databasen. Etter 10 sekunder gjør funksjonsnoden «Reset Error» at denne tabellen blir tom igjen.

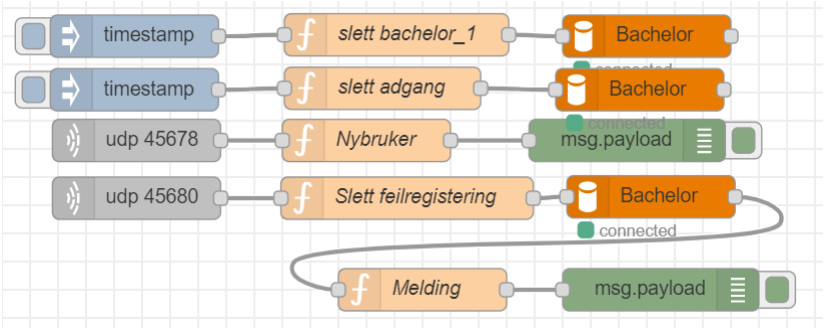
Verdien som blir lagt inn i tabellen Error brukes av PHP programmet til å sende ut en feilmelding som fremstilles visuelt på nettsiden.



Figur 3.23 Legger brikke ID til i SQL-database

Om brikkens ID ikke blir funnet i databasen blir signalet rutet videre i en annen del av Node-RED koden sett i Figur 3.23. Initialt i koden blir det kontrollert om navnet, på personen som skal legges til i systemet, allerede eksisterer i databasen. Om navnet allerede finnes vil filternoden hindre videre behandling av signalet og PHP programmet sende en feilmelding ut på nettsiden. Derimot om navnet ikke eksisterer vil funksjonsnoden «Legg til SQL», se Figur 3.23, legge inn brikke-ID og navn på brukeren inn i databasen. Signalet blir så rutet til funksjonsnoden INSERT SQL, se Figur 3.11, for da blir posisjonen der kortet ble scannet inn koblet sammen med det nyregistrerte kortet. Det resulterer i at fra det øyeblikk man registrerer en brikke på en person vil også plasseringen være synlig på den grafiske fremstillingen av de om bord.

I tillegg til å behandle avlesning av RFID-brikker har Node-RED en annen og viktig funksjon.



Figur 3.24 Mottak av data fra PHP-script

På figur 3.24 er det to manuelle, og to automatiske kjeder med noder. De to øverste kjedene er til for å kunne nullstille. Dersom begge timestampnodene aktiveres vill alt innholdet i henholdsvis «bachelor_1» og «adgang» fjernes, og systemet er da nullstilt. Dette er handlinger som kun kan gjøres fra Node-RED, og det er konstruert slik for å unngå at dette skjer ved et uhell.

Noden «UDP 45678» lytter på port 45678 for å få inn et signal fra nettsiden, som er samme port som deler av PHP-scriptene sender på. Hvis en bruker skal registrere ett nytt kort vil navnet på personen som skal motta kortet bli lagret i en variabel gjennom funksjonsnoden «Nybruker». Denne variabelen brukes i resten av Node-RED koden for å få knyttet navn opp mot brikke-ID ved etablering av en ny brikke eier. Det er viktig å merke seg at denne variabelen blir lagt inn i SQL-tabellen «adgang» som inneholder eiere av brikke-ID. Navnet som blir lagt inn har enda ikke fått tildelt noen brikke-ID da dette må leses inn på valgt leser. Noden «UDP 45680» lytter etter å få inn et signal fra et PHP-script. Denne noden vil få et inn signal gitt at det skjer en feil ved registrering av ny bruker. Funksjonen «Slett feilregistrering» tar og fjerner navn fra tabellen med brukere hvor brikke ID er lik null. Så om det skjer en feil, operasjonen blir avbrutt eller brikkens ID allerede er registrert, vil «UDP 45680» få et signal og fjerne alle overflødige navn i tabellen som omhandler brukere. Et resultat er at to eiere av en brikke-ID ikke kan ha samme registrerte navn i systemet.

3.2.3 PHP

Tidligere ble det nevnt at hele front end delene av oppgaven er skrevet i PHP, og systemet gjør det ved hjelp av 12 ulike filer. Disse utgjør igjen fem hovedfunksjoner. Det er en hjemmeside, administratorinnlogging, hjemmeside for administrator, å legge til kort, fjerne kort, samt å søke. Utenom dette er det en PHP-fil som kun inneholder back end funksjoner. Det å bruke PHP fremfor å utvikle en applikasjon til å gjøre samme ting handler om tilgjengelighet. Der PHP er mer sårbar for angrep er det av samme grunn og i mye større grad tilgjengelig. Der en applikasjon krever en forhåndsinstallasjon vil en løsning som er basert på PHP være tilgjengelig på alle plattformer som har en nettleser, uten behov for å gjøre noe på forhånd.

Det første brukeren av systemet vil møte er hjemmesiden til systemet. Her vises et nåtidsbilde av alle rom, hvor mange som er i hvilke rom, og hvilke ID-brikker som er i de ulike rommene. Her har brukeren videre direkte tilgang til tre funksjoner. Oppe til høyre er

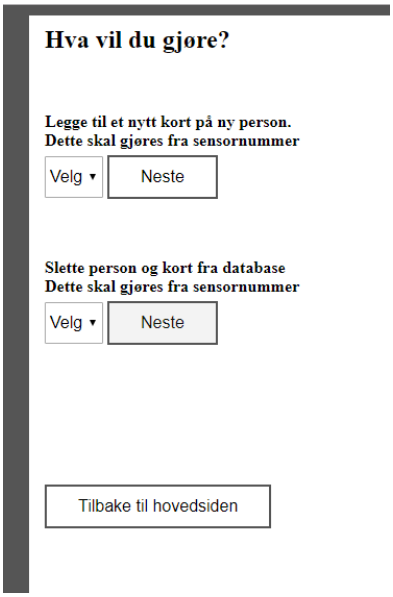
innloggingen til administratorer, som igjen fører til et lignende bilde der navn vises istedenfor ID. Videre kan bruker klikke seg til søkefunksjonen, samt administrere tilgangen til systemet.

```php
<div class="rom0">
    <?php

        try {
            $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
            $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

                $stmt = $conn->prepare('SELECT count(adgang.ID)
                                        FROM adgang INNER JOIN (SELECT bachelor_1.ID
                                                                FROM bachelor_1
                                                                INNER JOIN(SELECT ID, MAX(fulldate) as tid
                                                                        FROM bachelor_1 GROUP BY ID) as test
                                                                        WHERE test.tid = bachelor_1.fulldate AND test.ID = bachelor_1.ID AND position_new=0
                                                                        ORDER BY fulldate desc) as mellom
                                                                WHERE mellom.ID = adgang.ID');
                $stmt->execute();

            // set the resulting array to associative
            $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
            foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
                echo '<h1>'.$v.'</h1>';
            }
        }
        catch(PDOException $e) {
            echo "Error: " . $e->getMessage();
        }

        echo '<br>';

        try {
                $stmt = $conn->prepare('SELECT adgang.ID
                                        FROM adgang
                                        INNER JOIN (SELECT bachelor_1.ID
                                                    FROM bachelor_1
                                                    INNER JOIN(SELECT ID, MAX(fulldate) as tid
                                                    FROM bachelor_1 GROUP BY ID) as test
                                                    WHERE test.tid = bachelor_1.fulldate AND test.ID = bachelor_1.ID AND position_new=0
                                                    ORDER BY fulldate desc) as mellom
                                        WHERE mellom.ID = adgang.ID');
                $stmt->execute();

            // set the resulting array to associative
            $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
            foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
                echo '<h3>'.$v.'</h3>';
            }
        }
        catch(PDOException $e) {
            echo "Error: " . $e->getMessage();
        }
        $conn = null;
    ?>
</div>
```

Figur 3.25 Skjermdump av PHP kode

Programkoden i Figur 3.25 viser et utsnitt som dekker det viktigste av det som vises på hjemmesiden. Utsnittet viser mer nøyaktig hva som vises i rom 0. Teksten som vises på hjemmesiden er resultatet av to SQL-spørringer som PHP gjør til en database. De to lange strengene, den oransje teksten, er de faktiske spørringene som blir sendt til databasen. Svaret på den ene spørringen er antallet personer i det gitte rommet. Den andre spørringen spør ikke etter antallet registrerte enheter i rommet, men etter den faktiske ID-en til brikkene, og presenterer disse. For hvert nytt rom som skal legges til så må en lignende mengde kode legges til scriptet. Dette gjør at PHP delen av systemet er forholdvis enkel å skalere opp til å behandle mer enn to rom. Koden virker for to rom, og det å skalere den opp er i utgangspunktet en smal sak. Dette gjelder også for resten av PHP-koden i systemet.

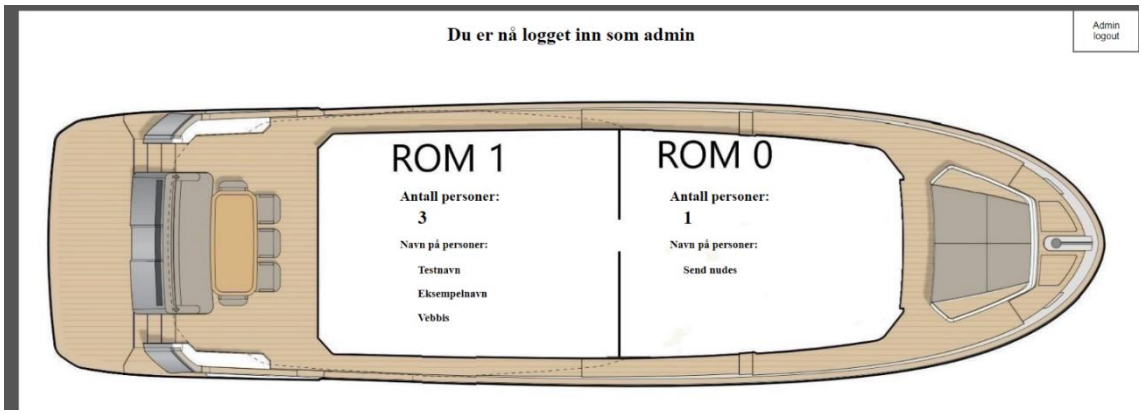
Om brukeren velger å administrere adgang, så blir brukeren møtt av en enkel nettside som gir muligheten til enten å legge til eller slette en bruker fra databasen (Figur 3.26). For å



Figur 3.26 Skjermdump av kortadministrasjon

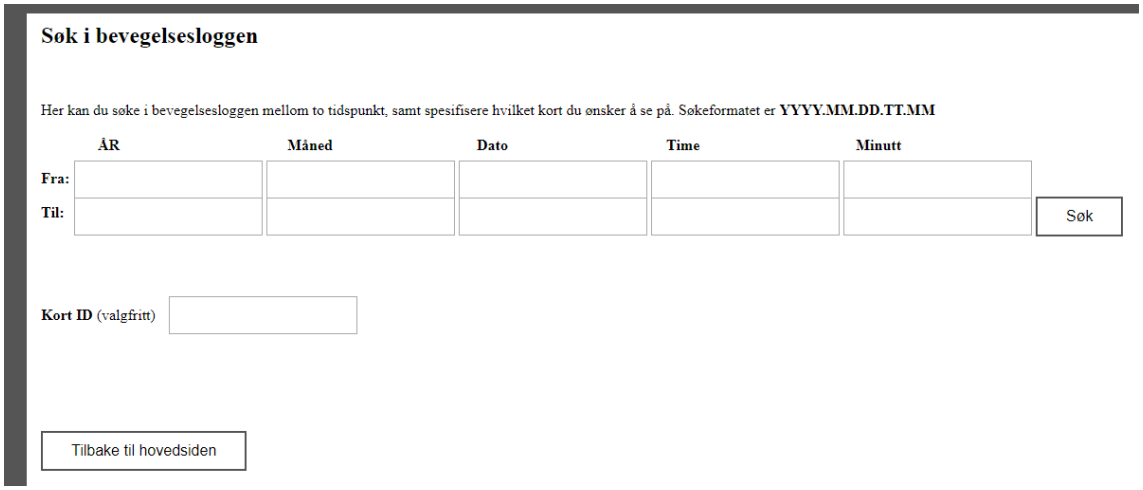
øke brukervennligheten til systemet er det mulig å gjøre all form for endring ved alle sensorer. Det er bare å velge hvilken sensor endringen skal gjøres fra, og da vil systemet endre modus til den valgte sensoren fra bare å lese passeringer til å skrive til databasen, enten i form av å legge til eller å slette. Dersom brukeren ønsker å legge til en bruker må sensor velges, og så trykke neste. Da settes sensoren i modus for å legge til databasen. Når navn er skrevet inn, og kort har blitt presentert til den gitte sensoren, så blir personene automatisk lagt inn i databasen. Da blir initialposisjon satt til samme som det rommet registreringen skjedde fra, og all bevegelse etter dette vil registreres av sensorene. Den logiske prosessen for å slette en person fra systemet er nesten den samme som for å legge til. Den eneste forskjellen er at brukeren ikke trenge å skrive inn navn på innehaveren av kortet for å slette det. Dersom brukeren ønsker å slette kortet må den bare velge slett under administrasjon og presentere kortet. Da vil det komme opp hvilket kort som har blitt slettet fra databasen, og hvem kortet tilhørte.

I oppgavebesvarelsen har det ikke blitt lagt inn en funksjon for å søke opp innehavere av kort i den hensikt å bevare retten til privatliv, og ikke falle utenfor personvernloven. Derimot eksisterer det en administratorside som har tilgang til navn. Den er nesten helt lik hjemmesiden, bare at ID er byttet ut med navn, som vist i figur 3.27. Her mangler alle andre funksjoner, og det står klart og tydelig at du er logget inn som administrator. Denne siden er kun tiltenkt bruk under prekære situasjoner, og er låst bak en innloggingsside. Det er kun systemansvarlig som har mulighet til å legge til nye brukere til administratorsiden, da disse må lages i SQL-databasen direkte.



Figur 3.27 Hjemmesiden i admin modus

Det er, som tidligere nevnt, ikke mulig å søke opp hvem som eier kort, eller å finne ut av hvor navngitte personer er uten videre. Det som derimot er mulig er å søke opp bevegelser i tidsrom.



Figur 3.28 Søkefunksjonen på hjemmesiden

Skjermdumpen i Figur 3.28 vil gi alle bevegelser imellom de gitte tidspunktene. Der vil du kunne se hvilken ID som har beveget seg mellom hvilke rom, til hvilket tidspunkt. Dersom brukeren velger å søke generelt uten å spesifisere ID mellom to tidspunkt vil den gi svaret med alle bevegelser på alle registrerte ID-er. Skulle brukeren søke på et gyldig tidspunkt, men med et ugyldig identifikasjonsnummer vil brukeren bli returnert til søke-siden sammen med en feilmelding som forteller at ID-nummeret ikke eksisterer. Måten dette gjøres på er at PHP-scriptet først sjekker hvorvidt brukeren har skrevet noe som helst i Kort ID- feltet. Dersom dette ikke er tilfellet kjører den en spørring rett til databa-sen som ignorerer feltet Kort ID. Har brukeren derimot skrevet noe i feltet sjekker scriptet først om det som er skrevet i feltet stemmer overens med noen av ID-brikkene som er registrert i databasen. Dersom dette gir et treff så skrives samme spørring som tidligere, bare at ID-nummeret blir lagt til som en ekstra betingelse. Skulle det derimot ikke bli noen treff vil brukeren bli presentert med en feilmelding som sier at kortet ikke ligger i databasen.

Denne typen logikk, og feilmeldingsgenerering, er gjennomgående i alle PHP-scriptene i systemet. Dette er lagt inn for å forhindre at brukeren ved uaktsomhet skal klare å sette systemet ut av spill. Mesteparten av denne typen sjekk går gjennom et back-end-script som har som eneste oppgave å omdirigere mellom sidene basert på hvilke verdier bruke-ren skriver inn. Det som kanskje er dette scriptets viktigste oppgave, er å tilbakestille alle

sensorer til lesemodus dersom noe skulle gå galt under enten registrering eller sletting. Dersom brukeren på noe som helst tidspunkt enten trykker på hjemknappen eller skriver inn noe ugyldig, som resulterer i en feilmelding, nullstilles alle variabler som ble satt fra brukeren startet til feilmeldingen ble generert.

4 Drøfting

Gjennom denne oppgaven er det blitt konstruert et system der en RFID-leser sender data gjennom en mikrokontroller til en Raspberry Pi. Ved hjelp av flere lesere er et tenkt fartøy, som er delt inn i seksjoner, via en datamaskin tilkoblet et LAN får man opp en hjemmeside med oversikt over lokasjonen til besetningen via deres RFID-brikker. Det hele blir fremstilt over et bilde av fartøyet for å gjøre det enkelt for en bruker å tolke informasjonen. Når programmet er i bruk skjermes privatlivet til de om bord ved at det kun vises brikke-ID på skjermene. Samtidig kommer det tydelig frem for en bruker av programmet hvor mange personer som oppholder seg i de forskjellige seksjonene. Dersom det oppstår situasjoner som krever det, kan autorisert personell med utvidet tilgang til programmet endre fremvisningen og få frem navn på alle registrerte personer sammen med deres tilhørende lokasjon.

Alle avlesninger av registrerte RFID-brikker blir også registrert i en tekstfil lokalt på Raspberry Pi om LAN-et skulle slutte å funke. En bruker av hjemmesiden har kun nødvendige valgalternativer, noe som gjør at det er lett å få oversikt. I tillegg er det konkrete og tydelige forklaringer på hva man som bruker skal gjøre for å oppnå ønsket sluttresultat.

Systemet er konstruert med noe av de billigste RFID antennene og brikkene du finner på markedet i dag. Det begrenset avlesningsrekkevidden til noen centimeter. Ved å investere mer i hardware kan systemet utvikles til å lese av RFID brikker bare ved at personen passerer et skott, en dør eller går i en trang gang. Det ville gjort systemet mer praktisk anvendelig.

I starten av idemyldringsfasen, før det var bestemt hvilke komponenter som skulle brukes, ble det jobbet mye med å designe plassering av antennene for å best mulig dekke en seksjon, uten at leserne interfererte med hverandre. Bedre hardware ville også muligens ha forandret den logiske oppbygningen programkoden nå bruker for å plassere en brikke i en seksjon. Det hadde vært en interessant utvidelse av oppgaven om budsjettet hadde vært større.

En annen ting som kan, og bør, legges til i programmet er en eller annen form for kryptering for å sikre innholdet. Det er ikke tatt hensyn til i oppgaven, men er absolutt noe som må bli gjort før man eventuelt tar det i bruk.

Videre i dette avsnittet skal vi utrede litt om utfordringer en eventuell implementering av teknologien vil ha for Forsvaret. Vi kommer til å ta for oss tre aspekter, praktiske utfordringer, rettigheter og sikkerhet. Drøftingen blir skrevet ut fra vårt kunnskapsnivå som snart ferdige bachelorstudenter innenfor militær ledelse med fordypning i elektronikk og data. Det finnes utfordringer eller løsninger som ikke blir drøftet i denne oppgaven.

4.1 Praktiske utfordringer

Teknologien er i rask utvikling, og det samme gjelder for RFID teknologien. Denne oppgaven har tatt utgangspunkt i, og brukt, et av de rimeligste alternativene som finnes på markedet for denne type identifisering. Om teknologien skulle bli implementert i Forsvaret ville det vært et stort behov for å oppgradere hardware-delen av systemet. For å kunne gjøre systemet så sikkert som mulig mot menneskelige feil måtte det vært konstruert med en leser som kunne registrere RFID-brikkene på lengre avstand enn det som er gjort i denne oppgaven. Om bord på et marinefartøy har alle i besetningen flere roller å fylle, som dykker, røykdykker, stempler(prøve å stanse vanninntrenging) og flere. Dette kan være en utfordring med en RFID brikke som man må ha med seg. Det er blitt tenkt mye på hvordan dette kunne vært løst i praksis for å skape minst mulig bry for mannskapet, samtidig som det er minst mulig rom for personlige feil. Om man skulle gått med et kort rundt halsen ville dette raskt vært i veien ved omkledning til for eksempel røykdykkerdrakt. Ved en eventuell kollisjon på natten er det lett å se for seg at noen glemmer å ta med seg kortet sitt. Av praktiske årsaker kan en hensiktsmessig løsning være å ta i bruk armbånd istedenfor kort.

Figur 4.1 RFID armband (RFID armband, 2019)

Armbåndet vil være mindre i veien i den daglige tjenesten og det vil være en høyere terskel for å ta av, men også her finnes det rom for menneskelige feil. En teknologi som har begynt å spre seg i verden og som flere store bedrifter har begynt å se muligheten i er en mikrobrikke implantert i hånden. Brikken er ikke større enn et riskorn og er en RFID-brikke. Det er nok litt futuristisk, men det vil redusere faren for at noen tar av seg, mister eller glemmer RFID-brikken. Noe som i dag ville vært en anselig feilkilde i systemet.

En annen interessant teknologi som kunne løst problemet, men som er i helt startfasen, er RFreeID. I stedet for at mennesker går rundt med passive brikker blir det satt opp et radiofrekvensfelt i et skott, eller lignende. Dette gjøres ved at det blir satt en sender på den ene siden av døren, og flere brikker langs dørkarmen på den andre siden. Når folk passerer igjennom blir forandringen i mottatt signal fra alle brikkene kjørt gjennom en algoritme som kan avgjøre ganglaget og høyden til personen som gikk gjennom. Det er derimot avhengig av at man har en stor database av passeringer fra de ansatte om bord for at algoritmen skal klare å identifisere hvem som passerte gjennom (Rfree ID, 2019). Den største tenkte utfordringen med dette systemet er igjen alle de ulike rollene besetningen har. Det er naturlig å anta at signaturen i uniform er annerledes om personen gikk i fullt røykdykkerutstyr.

4.2 Rettigheter

Den europeiske menneskerettighetskommisjonens (EMK) artikkel 8 sier *Enhver har rett til respekt for sitt privatliv og familieliv, sitt hjem og sin korrespondanse* (Datatilsynet(a), 2019).

Grunnloven paragraf 102 sier *Enhver har rett til respekt for sitt privatliv og familieliv, sitt hjem, sitt hjem og sin kommunikasjon. Husransakelse må ikke finne sted, unntatt i kriminelle tilfeller. Statens myndigheter skal sikre et vern om den personlige integritet* (Datatilsynet(a), 2019).

Det å være om bord på et fartøy er en helt spesiell situasjon. Man kan være ute på tokt i måneder i strekk, og fartøyet vil fungere som et hjem. Hvorvidt der er mulig å legitimere det å følge med på folks bevegelser om bord kan bli stoppet av den enkeltes rett til privatliv. På en annen side er denne oppgaven tenkt opp mot militære fartøy, og det kan sammenlignes med militære installasjoner på land. På militære baser i Norge driver Forsvaret med adgangskontroll og registrerer hvor på basene folk har registrert seg. Det finnes besetningsmedlemmer på fregatt som tilnærmet bor på fartøyet året rundt. Så om noe liknende system skal bli tatt i bruk av marinen, må det grundige undersøkelser til for å kartlegge om dette i det hele tatt er lovlig. Ved å ha større seksjoner om bord og akseptere at nøyaktigheten til plasseringen blir lavere reduseres inngripen. Om man velger å installere systemet med lesere for hvert eneste rom vil det være en større inngripen i privatlivet enn om fartøyet ble delt i færre seksjoner. Innenfor en militær organisasjon burde det vurderes om den enkeltes rett på privatliv burde vike for å skape best mulig grunnlag for personsikkerhet ved en eventuell ulykke, og redusere sjansen for en katastrofe med tap av menneskeliv.

Forsvaret har i 2019 bedrevet en omfattende kampanje for å øke de ansattes kunnskap om personvern og behandling av personopplysninger så problemstillingen rundt personells rettigheter er høyst relevant og viktig for Forsvaret i dag. Det var en omfattende prosess for Equinor å få godkjent sin personellkontrollordning om bord på oljeplattformen Johan Sverdrup. De fikk tilslutt godkjent søknaden sin da de, i likhet med denne oppgaven, hadde anonymisert de ansatte under daglig drift, men sikret at de ved stort behov kunne avsløre identiteten på de ansatte som befant seg rundt om på plattformen.

4.3 Sikkerhet

Forsvaret er under daglige internasjonale cyberangrep. Motivasjonen og hva de ønsker å få ut av disse angrepene har ikke vi mye kunnskap eller kompetanse om. Det kan derfor ikke utelukke at bevegelsesmønstre om bord på fartøy kan være av stor interesse for utenforstående. Derfor vil det være av høy viktighet at systemet er robust og vanskelig for uønskede å få tilgang til. Det er ikke gjort noen tiltak for å hindre utenforstående uønskede adgang til systemet i denne oppgaven, noe som er viktig å presisere. Av tekniske årsaker er det noe sikkerhet som vil være naturlig integrert i systemet. Om systemet opereres på et LAN, som i denne oppgaven, vil rekkevidden være en naturlig begrensning. Man måtte så å si være om bord på fartøyet for i det hele tatt ha muligheten til å få forbindelse med systemet. Under Defcon(hacker konferanse) i 2008 i USA ble det blant annet laget en antenne, av relativt billige komponenter, som klarte å lese av en passiv RFID-brikke på ca. 23 meters avstand(69 feet) (The last Hope, 2008). Teknologien utvikles raskt, og det bør forventes at enda lavere signalstyrke kan bli avlest i dag.

Selv om et fartøy er av militær karakter er det ikke uvanlig at det er besøk eller omvisninger om bord. Dette er helt klart en sikkerhetstrussel mot systemet. Det er dermed garantert at andre enn besetningen vil være innenfor forbindelsesrekkevidde til LAN-et. Det er derfor viktig at det legges inn sikkerhetsparametere som passord for Wi-Fi og eventuelt ytterligere krypteringstiltak.

En annen sikkerhetstrussel er fysisk tilkobling. Det er lite trolig, men når det er andre enn norske militært ansatte om bord kan det være en mulighet for at noen forsøker å koble seg fysisk til systemet, eller prøver å avlese data i noen av kablene om bord. Det er et moment som det er viktig å være bevisst på. Derimot er det vanskelig å gjøre aktive tiltak mot denne typen trusler, annet enn å være årvåken når man ferdes om bord på fartøyet.

Det blir, som nevnt i kapittelet 4.2, samlet inn data om besetningen som ferdes om bord, og disse dataene vil Forsvaret være lov forpliktet til å behandle på en sikker og god måte. Det kan være av interesse å samle på dataen for å skape statistikk eller kontrollere at sikkerhetsrunder om bord gjennomføres i henhold til instruks. Da er det viktig at også dette aspektet av sikkerheten er tatt vare på innenfor loven om personopplysninger sin ramme.

5 Konklusjon med anbefaling

Oppgaven har tatt for seg konstruksjon og design av et personellkontrollsystem, med et fokus på å fremstille en besetnings lokasjon om bord på et fartøy på en oversiktlig og ryddig måte.

Sluttproduktet tilfredsstiller alle målene satt i oppgaven. Nettsiden er oversiktlig og gir tilgang til de viktigste funksjonene, har tydelige forklaringer og gir en veldig ryddig grafisk fremstilling av lokasjonen til besetningen. Systemet leser av RFID-brikker slik som ønsket. Samtidig kommer det tydelig frem for en bruker av programmet hvor mange personer som oppholder seg i de forskjellige seksjonene. Antallet funksjoner i nettsiden er begrenset for å holde kravet til opplæring så lavt som mulig. Videre er fremvisningen anonymisert frem til det er nødvendig å finne lokasjonen til besetningen. For å knytte bevegelser opp mot personopplysninger kreves det spesiell tilgang som det skal være tydelige instrukser på når kan anvendes. Denne oppgaven bekrefter at vi har hatt en høy grad av måloppnåelse. Av målsetningene vi satt er de tre første utført i meget høy grad, mens det fjerde målet om å tilfredsstille krav om personvern er vanskelig å måle. På en annen side er personvernet om bord tilfredsstilt på lik linje med personvernet på enhver militær base i Norge.

Dataen som bli samlet inn kan være av interesse for Forsvaret for å kunne kartlegge og forbedre sine rutiner om bord. Ved hjelp av informasjonen kan man etterprøve opplæring på for eksempel vakt runder. Dette kan resultere i en mer sikker drift av fartøy og redusere risikoen for uhell om bord.

Vi anbefaler at Forsvaret gjennomfører en risikovurdering i den hensikt å kartlegge fordeler og ulemper ved et slikt system. Vi mener et slikt system kan redusere konsekvensene ved en ulykke og bedre rutiner om bord, noe vi alltid må tilstrebe i forsvaret.

Bibliografi

Arduino(a)

Arduino Uno Rev3 SMD. Åpnet 26. november 2019.

https://store.arduino.cc/arduino-uno-rev3-smd

RC522 RFID modul

Last Minute Engineers. «In-Depth: What Is RFID? How It Works? Interface RC522 with Arduino», 30. juli 2018. Åpnet 24.november 2019.

https://lastminuteengineers.com/how-rfid-works-rc522-arduino-tutorial/

Arduino åpen kildekode

«MFRC522-1.4.4». Åpnet 28. november 2019.

https://www.arduinolibraries.info/libraries/mfrc522

Datatilsynet(a)

Datatilsynet. «Personvern». Åpnet 29. november 2019.

https://www.datatilsynet.no/rettigheter-og-plikter/hva-er-personvern/

Datatilsynet(b)

Datatilsynet. «Unntak». Åpnet 29. november 2019.

https://www.datatilsynet.no/rettigheter-og-plikter/virksomhetenes-plikter/gi-informasjon/informasjon-og-apenhet/unntak-fra-plikten-til-a-gi-informasjon/

RFID armbånd

«rfid bracelet – Google Søk». Åpnet 29. november 2019. https://www.google.com/search?q=rfid+bracelet&source=lnms&tbm=isch&sa=X&ved=2ahUKEwis84S88Y_mAhXltYsKHbhBA_UQ_AUoAXoECAwQAw&biw=1500&bih=858#imgrc=4n3yLPNram2IVM

Lowry solutions

Solutions, Lowry. «What Are the Different Types of RFID Technology?» *Lowry Solutions* (blog), 12. desember 2014.

https://lowrysolutions.com/blog/what-are-the-different-types-of-rfid-technology/

Sparkfun

«What is an Arduino». Åpnet 20. november 2019.

https://learn.sparkfun.com/tutorials/what-is-an-arduino/all

Seedsstudio

«UART vs I2C vs SPI – Communication Protocols and Uses». Åpnet 21. november 2019.

https://www.seeedstudio.com/blog/2019/09/25/uart-vs-i2c-vs-spi-communication-protocols-and-uses/

Rfree-ID

Zhang, Qian, Run Zhao, Dong Li, og Dong Wang. «Unobtrusive and Robust Human Identification Using COTS RFID». *Computer Networks*, 16. juli 2019. https://doi.org/10.1016/j.comnet.2019.05.018

The last hope 2008 (Hacker konferanse)

Foredrag av Tiffany Strauchs Rad. Åpnet 17. november 2019.

https://www.youtube.com/watch?v=JUhrwL3vo5w&t=786s

Rapport Helge Ingstad

Statens havarikomisjon. «DELRAPPORT 1 OM KOLLISJONEN MELLOM FREGATTEN KNM HELGE INGSTAD OG TANKBÅTEN SOLA TS UTENFOR STURETERMINALEN I HJELTEFJORDEN, HORDALAND, 8. NOVEMBER 2018». November 2019.

Arduino(b)

«Hva er arduino og Genuino?». Åpnet 02. desember 2019.

https://www.kjell.com/no/kunnskap/hvordan-virker-det/arduino/introduksjon/hva-er-arduino-og-genuino

Raspberry Pi

«Raspberry Pi 3 Model B+». Åpnet 22. november 2019.

https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

Node-RED

"Node-RED". Åpnet 02. desember 2019.

https://nodered.org/

Vedlegg

A. Admin_logon.php

```
<html>
<head>
  <title>Søk</title>
</head>
<style>
    .button {
        border: none;
        color: white;
        padding: 10px 30px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 16px;
        margin: 5px 0px;
        -webkit-transition-duration: 0.4s; /* Safari */
        transition-duration: 0.4s;
        cursor: pointer;
    }

    .button1 {
        background-color: white;
        color: black;
        border: 2px solid #555555;
    }

    .button1:hover {
        background-color: #555555;
        color: white;
    }

    #top, #left {
        background: #555555;
        position: fixed;
    }

    #left {
        top: 0;
        bottom: 0;
        width: 25px;
    }

    #left {
        left: 0;
    }

    #top {
        left: 0;
        right: 0;
        height: 10px;
    }

    #top {
        top: 0;
    }

    .flytte_tekst {
        position: absolute;
        top: 0px;
        left: 40px;
```

```
    }
        .year {
        position: absolute;
        top: 140px;
        left: 100px;
    }
        .month {
        position: absolute;
        top: 140px;
        left: 300px;
    }
        .date {
        position: absolute;
        top: 140px;
        left: 500px;
    }
        .hour {
        position: absolute;
        top: 140px;
        left: 700px;
    }
        .minute {
        position: absolute;
        top: 140px;
        left: 900px;
    }
        .boks {
        position: absolute;
        top: 145px;
        left: 100px;
    }
        .boks2 {
        position: absolute;
        top: 115px;
        left: 100px;
    }
        .boks3 {
        position: absolute;
        top: 50px;
        left: 50px;
        }
    </style>
    <body>
        <div id="left"></div>
        <div id="top"></div>
        <div class = "boks3">
                <?php
            if ($_GET["error"] > 0)
            {
                echo "<h4>Det har skjedd en feil. ";
                if ($_GET["error"] == 9)
                {
                    echo " Feilen er ukjent, prøv igjen";
                }
                else if ($_GET["error"] == 1)
                {
                    echo "Feil brukernavn eller passord. ";

                }
                echo "<br><br><br></h4>";
            }
            ?>
        </div>

        <div class= "flytte_tekst">
            <h2>Logg inn under</h2>
            <br><br><br>
```

```
            <b>Brukernavn</b><br><br>
            <b>Passord</b>
            <div class= boks2 style= "width:1400px">
            <form action="redirect.php" method="get">
                <input style="font-size:16px; height:40px;" type="text" name="bruker">
            </div>


            <div class= boks style= "width:1400px">
                <input style="font-size:16px; height:40px;" type="password" name="passord">
                <button class="button button1" type="submit" name="red_log" value = 400>Logg inn</button>
            </div>
            </form>
             <br><br><br><br><br><br><br>
            <form action="redirect.php" method="get">
            <button class="button button1" type="submit" name= "hjemknapp" value= 100>Tilbake til hovedsiden</button>
            </form>
        </div>
    </body>
    </html>
```

B. Index.php

```
<html>
<head>
  <title>Hjemmeside</title>
    <meta http-equiv="refresh" content="3">
    <meta name="viewport" content="width=device-width, initial-scale=1">
<style>
    .button {
        border: none;
        color: white;
        padding: 16px 32px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 16px;
        margin: 0px 0px;
        -webkit-transition-duration: 0.4s; /* Safari */
        transition-duration: 0.4s;
        cursor: pointer;
    }

    .button1 {
        background-color: white;
        color: black;
        border: 2px solid #555555;
    }

    .button1:hover {
        background-color: #555555;
        color: white;
    }

    #top, #left {
        background: #555555;
        position: fixed;
    }

```

```
37.     #left {
38.         top: 0;
39.         bottom: 0;
40.         width: 25px;
41.     }
42.
43.     #left {
44.         left: 0;
45.     }
46.
47.     #top {
48.         left: 0;
49.         right: 0;
50.         height: 10px;
51.     }
52.
53.     #top {
54.         top: 0;
55.     }
56.
57.     .rom0 {
58.         position: absolute;
59.         top: 320px;
60.         right: 680px;
61.     }
62.     .rom0_info {
63.         position: absolute;
64.         top: 290px;
65.         right: 625px;
66.     }
67.
68.
69.     .rom1 {
70.         position: absolute;
71.         top: 320px;
72.         left: 700px;
73.     }
74.     .rom1_info {
75.         position: absolute;
76.         top: 290px;
77.         left: 670px;
78.     }
79. </style>
80. </head>
81. <body>
82.     <?php
83.         $servername = "192.168.1.3";
84.         $username = "admin";
85.         $password = "Test123";
86.         $dbname = "Bachelor";
87.
88.         class TableRows extends RecursiveIteratorIterator {
89.             function __construct($it) {
90.                 parent::__construct($it, self::LEAVES_ONLY);
91.             }
92.
93.             function current() {
94.                 return "<td style='width:150px;bor-
    der:1px solid black;'>" . parent::current(). "</td>";
95.             }
96.
97.             function beginChildren() {
98.                 echo "<tr>";
99.             }
100.
101.                function endChildren() {
102.                    echo "</tr>" . "\n";
```

```
                }
            }
        ?>
        <div id="left"></div>
        <div id="top"></div>
        <form action="nytt_kort.php" method="get">
            <button class="button button1" type="submit" style="mar-
    gin: 0px 15px">Administrer adgang</button>
        </form>

        <form action="search.php" method="get">
            <button class="button button1" type="submit" style="margin: -
    70px 220px">Søk i bevegelsesloggen</button>
        </form>

        <form action="admin_logon.php" method="get">
            <button class="button button1" type="submit" style="margin: -
    86px 1800px">Admin login</button>
        </form>

        <img src="img/boat_web_done.jpg" alt="Snow" style="width:95%; mar-
    gin: 20px 50px">
        <div class="rom1_info">
            <h2> Antall personer: </h2>
            <br>
            <h3> Tilhørende ID: </h3>
        </div>
        <div class="rom1">
            <?php

                try {
                    $conn = new PDO("mysql:host=$server-
    name;dbname=$dbname", $username, $password);
                    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEP-
    TION);

                        $stmt = $conn->prepare('SELECT count(adgang.ID)
                                                FROM adgang
                                                INNER JOIN (SELECT bache-
    lor_1.ID
                                                            FROM bachelor_1
                                                            INNER JOIN(SE-
    LECT ID, MAX(fulldate) as tid
                                                            FROM bache-
    lor_1 GROUP BY ID) as test
                                                            WHERE test.tid = ba
    chelor_1.fulldate AND test.ID = bachelor_1.ID AND position_new=1
                                                            OR-
    DER BY fulldate desc) as mellom
                                                WHERE mellom.ID = ad-
    gang.ID');
                        $stmt->execute();

                    // set the resulting array to associative
                    $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
                    foreach(new TableRows(new RecursiveArrayIterator($stmt-
    >fetchAll())) as $k=>$v) {
                        echo '<h1>'.$v.'</h1>';
                    }
                }
                catch(PDOException $e) {
                    echo "Error: " . $e->getMessage();
                }
                echo '<br>';
                try {

                        $stmt = $conn->prepare('SELECT adgang.ID
```

```
157.                                                 FROM adgang
158.                                                 INNER JOIN (SELECT bache-
    lor_1.ID
159.                                                             FROM bachelor_1
160.                                                             INNER JOIN(SE-
    LECT ID, MAX(fulldate) as tid
161.                                                             FROM bache-
    lor_1 GROUP BY ID) as test WHERE test.tid = bache-
    lor_1.fulldate AND test.ID = bachelor_1.ID AND position_new=1
162.                                                             OR-
    DER BY fulldate desc) as mellom
163.                                                 WHERE mellom.ID = ad-
    gang.ID');
164.                         $stmt->execute();
165.
166.                     // set the resulting array to associative
167.                     $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
168.                     foreach(new TableRows(new RecursiveArrayIterator($stmt-
    >fetchAll())) as $k=>$v) {
169.                         echo '<h3>'.$v.'</h3>';
170.
171.                     }
172.                 }
173.                 catch(PDOException $e) {
174.                     echo "Error: " . $e->getMessage();
175.                 }
176.                 $conn = null;
177.             ?>
178.         </div>
179.         <div class="rom0_info">
180.             <h2> Antall personer: </h2>
181.             <br>
182.             <h3> Tilhørende ID: </h3>
183.         </div>
184.         <div class="rom0">
185.             <?php
186.
187.                 try {
188.                     $conn = new PDO("mysql:host=$server-
    name;dbname=$dbname", $username, $password);
189.                     $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEP-
    TION);
190.
191.                         $stmt = $conn->prepare('SELECT count(adgang.ID)
192.                                                 FROM adgang INNER JOIN (SE-
    LECT bachelor_1.ID
193.                                                                         FROM ba
    chelor_1
194.                                                                         IN-
    NER JOIN(SELECT ID, MAX(fulldate) as tid
195.
        FROM bachelor_1 GROUP BY ID) as test
196.
        WHERE test.tid = bachelor_1.fulldate AND test.ID = bachelor_1.ID AND posi-
    tion_new=0
197.
        ORDER BY fulldate desc) as mellom
198.                                                                         WHERE m
    ellom.ID = adgang.ID');
199.                         $stmt->execute();
200.
201.                     // set the resulting array to associative
202.                     $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
203.                     foreach(new TableRows(new RecursiveArrayIterator($stmt-
    >fetchAll())) as $k=>$v) {
204.                         echo '<h1>'.$v.'</h1>';
205.                     }
```

```
                }
                catch(PDOException $e) {
                    echo "Error: " . $e->getMessage();
                }

                echo '<br>';

                try {
                        $stmt = $conn->prepare('SELECT adgang.ID
                                                FROM adgang
                                                INNER JOIN (SELECT bache-
lor_1.ID
                                                            FROM bachelor_1
                                                            INNER JOIN(SE-
LECT ID, MAX(fulldate) as tid
                                                            FROM bache-
lor_1 GROUP BY ID) as test
                                                            WHERE test.tid = ba
chelor_1.fulldate AND test.ID = bachelor_1.ID AND position_new=0
                                                            OR-
DER BY fulldate desc) as mellom
                                                WHERE mellom.ID = ad-
gang.ID');
                        $stmt->execute();

                    // set the resulting array to associative
                    $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
                    foreach(new TableRows(new RecursiveArrayIterator($stmt-
>fetchAll())) as $k=>$v) {
                        echo '<h3>'.$v.'</h3>';
                    }
                }
                catch(PDOException $e) {
                    echo "Error: " . $e->getMessage();
                }
                $conn = null;
            ?>
        </div>
    </body>
    </html>
```

C. Index_admin.php

```
<html>
<head>
  <title>Hjemmeside</title>
    <meta http-equiv="refresh" content="3">
    <meta name="viewport" content="width=device-width, initial-scale=1">
<style>
    .button {
        border: none;
        color: white;
        padding: 16px 32px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 16px;
        margin: 0px 0px;
        -webkit-transition-duration: 0.4s; /* Safari */
        transition-duration: 0.4s;
        cursor: pointer;
    }

    .button1 {
```

```
        background-color: white;
        color: black;
        border: 2px solid #555555;
    }

    .button1:hover {
        background-color: #555555;
        color: white;
    }

    #top, #left {
        background: #555555;
        position: fixed;
    }

    #left {
        top: 0;
        bottom: 0;
        width: 25px;
    }

    #left {
        left: 0;
    }

    #top {
        left: 0;
        right: 0;
        height: 10px;
    }

    #top {
        top: 0;
    }

    .rom0 {
        position: absolute;
        top: 320px;
        right: 680px;
    }
    .rom0_info {
        position: absolute;
        top: 290px;
        right: 625px;
    }


    .rom1 {
        position: absolute;
        top: 320px;
        left: 700px;
    }
    .rom1_info {
        position: absolute;
        top: 290px;
        left: 670px;
    }
        .info {
        position: absolute;
        top: 10px;
        left: 750px;
    }


</style>
</head>
<body>
```

```
89.     <?php
90.         $servername = "192.168.1.3";
91.         $username = "admin";
92.         $password = "Test123";
93.         $dbname = "Bachelor";
94.
95.         class TableRows extends RecursiveIteratorIterator {
96.             function __construct($it) {
97.                 parent::__construct($it, self::LEAVES_ONLY);
98.             }
99.
100.                function current() {
101.                    return "<td style='width:150px;bor-
    der:1px solid black;'>" . parent::current(). "</td>";
102.                }
103.
104.                function beginChildren() {
105.                    echo "<tr>";
106.                }
107.
108.                function endChildren() {
109.                    echo "</tr>" . "\n";
110.                }
111.            }
112.        ?>
113.        <div id="left"></div>
114.        <div id="top"></div>
115.        <div class="info"><h1> Du er nå logget inn som admin </h1> </div>
116.
117.        <form action="index.php" method="get">
118.            <button class="button button1" type="submit" style="mar-
    gin: 0px 1800px">Admin logout</button>
119.        </form>
120.
121.        <img src="img/boat_web_done.jpg" alt="Snow" style="width:95%; mar-
    gin: 20px 50px">
122.        <div class="rom1_info">
123.            <h2> Antall personer: </h2>
124.            <br>
125.            <h3> Navn på personer: </h3>
126.        </div>
127.        <div class="rom1">
128.            <?php
129.
130.                try {
131.                    $conn = new PDO("mysql:host=$server-
    name;dbname=$dbname", $username, $password);
132.                    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEP-
    TION);
133.
134.                        $stmt = $conn->prepare('SE-
    LECT count(adgang.navn) FROM adgang INNER JOIN (SELECT bache-
    lor_1.ID FROM bachelor_1 INNER JOIN(SELECT ID, MAX(fulldate) as tid FROM bach-
    elor_1 GROUP BY ID) as test WHERE test.tid = bache-
    lor_1.fulldate AND test.ID = bachelor_1.ID AND position_new=1 OR-
    DER BY fulldate desc) as mellom where mellom.ID = adgang.ID');
135.                        $stmt->execute();
136.
137.                    // set the resulting array to associative
138.                    $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
139.                    foreach(new TableRows(new RecursiveArrayIterator($stmt-
    >fetchAll())) as $k=>$v) {
140.                        echo '<h1>'.$v.'</h1>';
141.                    }
142.                }
143.                catch(PDOException $e) {
144.                    echo "Error: " . $e->getMessage();
```

```
145.                }
146.                echo '<br>';
147.                try {
148.                        $stmt = $conn->prepare('SE-
    LECT adgang.navn FROM adgang INNER JOIN (SELECT bachelor_1.ID FROM bache-
    lor_1 INNER JOIN(SELECT ID, MAX(fulldate) as tid FROM bache-
    lor_1 GROUP BY ID) as test WHERE test.tid = bache-
    lor_1.fulldate AND test.ID = bachelor_1.ID AND position_new=1 OR-
    DER BY fulldate desc) as mellom where mellom.ID = adgang.ID');
149.                        $stmt->execute();
150.
151.                    // set the resulting array to associative
152.                    $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
153.                    foreach(new TableRows(new RecursiveArrayIterator($stmt-
    >fetchAll())) as $k=>$v) {
154.                        echo '<h3>'.$v.'</h3>';
155.
156.                    }
157.                }
158.                catch(PDOException $e) {
159.                    echo "Error: " . $e->getMessage();
160.                }
161.                $conn = null;
162.            ?>
163.        </div>
164.        <div class="rom0_info">
165.            <h2> Antall personer: </h2>
166.            <br>
167.            <h3> Navn på personer: </h3>
168.        </div>
169.        <div class="rom0">
170.            <?php
171.
172.                try {
173.                    $conn = new PDO("mysql:host=$server-
    name;dbname=$dbname", $username, $password);
174.                    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEP-
    TION);
175.
176.                        $stmt = $conn->prepare('SE-
    LECT count(adgang.navn) FROM adgang INNER JOIN (SELECT bache-
    lor_1.ID FROM bachelor_1 INNER JOIN(SELECT ID, MAX(fulldate) as tid FROM bach-
    elor_1 GROUP BY ID) as test WHERE test.tid = bache-
    lor_1.fulldate AND test.ID = bachelor_1.ID AND position_new=0 OR-
    DER BY fulldate desc) as mellom where mellom.ID = adgang.ID');
177.                        $stmt->execute();
178.
179.                    // set the resulting array to associative
180.                    $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
181.                    foreach(new TableRows(new RecursiveArrayIterator($stmt-
    >fetchAll())) as $k=>$v) {
182.                        echo '<h1>'.$v.'</h1>';
183.                    }
184.                }
185.                catch(PDOException $e) {
186.                    echo "Error: " . $e->getMessage();
187.                }
188.
189.                echo '<br>';
190.
191.                try {
192.                        $stmt = $conn->prepare('SE-
    LECT adgang.navn FROM adgang INNER JOIN (SELECT bachelor_1.ID FROM bache-
    lor_1 INNER JOIN(SELECT ID, MAX(fulldate) as tid FROM bache-
    lor_1 GROUP BY ID) as test WHERE test.tid = bache-
    lor_1.fulldate AND test.ID = bachelor_1.ID AND position_new=0 OR-
    DER BY fulldate desc) as mellom where mellom.ID = adgang.ID');
```

```
                        $stmt->execute();

                    // set the resulting array to associative
                    $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
                    foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
                        echo '<h3>'.$v.'</h3>';
                    }
                }
                catch(PDOException $e) {
                    echo "Error: " . $e->getMessage();
                }
                $conn = null;
            ?>

        </div>
    </body>
    </html>
```

D. Nytt_kort.php

```
<html>
<head>
  <title>Legg til kort</title>
</head>
<style>
    .button {
        border: none;
        color: white;
        padding: 10px 30px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 16px;
        margin: 5px 0px;
        -webkit-transition-duration: 0.4s; /* Safari */
        transition-duration: 0.4s;
        cursor: pointer;
    }

    .button1 {
        background-color: white;
        color: black;
        border: 2px solid #555555;
    }

    .button1:hover {
        background-color: #555555;
        color: white;
    }

    #top, #left {
        background: #555555;
        position: fixed;
    }

    #left {
        top: 0;
        bottom: 0;
        width: 25px;
    }

    #left {
        left: 0;
```

```
44.     }
45.
46.     #top {
47.         left: 0;
48.         right: 0;
49.         height: 10px;
50.     }
51.
52.     #top {
53.         top: 0;
54.     }
55.
56.     .flytte_tekst {
57.         position: absolute;
58.         top: 0px;
59.         left: 40px;
60.     }
61. </style>
62. <body>
63.     <div id="left"></div>
64.     <div id="top"></div>
65.     <div class= "flytte_tekst">
66.         <?php
67.         if ($_GET["errorcode"] > 0)
68.         {
69.             echo "<h2>Det har skjedd en feil. ";
70.             if ($_GET["errorcode"] == 9)
71.             {
72.                 echo " Feilen er ukjent, prøv igjen";
73.             }
74.             else if ($_GET["errorcode"] == 1)
75.             {
76.                 echo "Personen du har forsøkt å legge inn eksisterer alle-
    rede. ";
77.
78.             }
79.             else if ($_GET["errorcode"] == 2)
80.             {
81.                 echo "Du glemte å velge sensornummer. Sensornummer må vel-
    ges for å legge til person i databasen.";
82.
83.             }
84.             else if ($_GET["errorcode"] == 3)
85.             {
86.                 echo "Ingen navn ble skre-
    vet inn. Husk å skrive inn navn på personen som skal legges til";
87.             }
88.             else if ($_GET["errorcode"] == 4)
89.             {
90.                 echo "Kortet tilhører allerede en person. Bruk et an-
    net kort.";
91.             }
92.             else if ($_GET["errorcode"] == 5)
93.             {
94.                 echo "Du prøvde å slette et kort som ikke ligger i databa-
    sen.";
95.             }
96.             echo "<br><br><br></h2>";
97.         }
98.         ?>
99.
100.            <form action="redirect.php" method="get">
101.            <h2>Hva vil du gjøre?</h2>
102.            <br><br>
103.                <b>Legge til et nytt kort på ny person.
104.            <br>
105.                Dette skal gjøres fra sensornummer
```

```
106.
107.           <br>
108.
109.           <select style="font-size:16px; height:40px;" type="text" name="sen-
    sornummer">
110.             <option value="">Velg</option>
111.             <option value="0">0</option>
112.             <option value="1">1</option>
113.           </select>
114.
115.           <button class="button button1" type="submit" name="sen-
    sorverdi" value= "1">Neste</button>
116.           <input type="hidden" name ="red_1_2" value= 200 />
117.           </form>
118.            <br> <br> <br>
119.
120.
121.           <form action="slette_kort.php" method="get">
122.               Slette person og kort fra database
123.           <br>
124.               Dette skal gjøres fra sensornummer</b>
125.            <br>
126.           <select style="font-size:16px; height:40px;" type="text" name="sen-
    sornummer">
127.             <option value="">Velg</option>
128.             <option value="0">0</option>
129.             <option value="1">1</option>
130.           </select>
131.           <button class="button button1" type="submit" name="sen-
    sorverdi" value= "2">Neste</button>
132.           </form>
133.           <br><br><br><br><br><br>
134.           <form action="redirect.php" method="get">
135.           <button class="button button1" type="sub-
    mit" name= "hjemknapp" value= 100>Tilbake til hovedsiden</button>
136.           </form>
137.       </div>
138.   </body>
139.   </html>
```

E. Nytt_kort2.php

```
1.  <html>
2.  <head>
3.    <title>Legg til kort</title>
4.  </head>
5.  <style>
6.      .button {
7.          border: none;
8.          color: white;
9.          padding: 10px 30px;
10.         text-align: center;
11.         text-decoration: none;
12.         display: inline-block;
13.         font-size: 16px;
14.         margin: 5px 0px;
15.         -webkit-transition-duration: 0.4s; /* Safari */
16.         transition-duration: 0.4s;
17.         cursor: pointer;
18.     }
19.
20.     .button1 {
21.         background-color: white;
22.         color: black;
```

```
        border: 2px solid #555555;
    }

    .button1:hover {
        background-color: #555555;
        color: white;
    }

    #top, #left {
        background: #555555;
        position: fixed;
    }

    #left {
        top: 0;
        bottom: 0;
        width: 25px;
    }

    #left {
        left: 0;
    }

    #top {
        left: 0;
        right: 0;
        height: 10px;
    }

    #top {
        top: 0;
    }

    .flytte_tekst {
        position: absolute;
        top: 0px;
        left: 40px;
    }
</style>
<body>
    <div id="left"></div>
    <div id="top"></div>
    <div class= "flytte_tekst">
        <h2>Hvem skal bruke kortet?</h2>
        <br>
        <form action="redirect.php" method="get">
        <b><font size="6">Navn:</font></b> <input style="font-size:16px; height:40px;" type="text" name="navn123">
        <button class="button button1" type="submit" name="red_2_3" value= 300>Videre</button>
        </form>
        <br><br>
        <form action="redirect.php" method="get">
        <button class="button button1" type="submit" name= "hjemknapp" value= 100>Hjem</button>
        </form>
    </div>
<?php

$servername = "192.168.1.3";
$username = "admin";
$password = "Test123";
$dbname = "Bachelor";


try {
```

```
86.     $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $pass-
    word);
87.     $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
88.
89.     $stmt = $conn->prepare("UPDATE sensormodus SET sensor" . $_GET["sensor-
    nummer"]. "=". $_GET["sensorverdi"].  " WHERE ID='A'");
90.     $stmt->execute();
91.
92.     // set the resulting array to associative
93.     $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
94.     foreach(new TableRows(new RecursiveArrayIterator($stmt->fetch-
    All())) as $k=>$v) {
95.         echo $v;
96.         }
97. }
98. catch(PDOException $e) {
99.     echo "Error: " . $e->getMessage();
100.
101.
102.
103.
104.    }
105.    $conn = null;
106.    ?>
107.    </body>
108.    </html>
```

F. Nytt_kort3.php

```
1.  <html>
2.  <head>
3.    <title>Legg til kort</title>
4.  </head>
5.  <style>
6.      .button {
7.          border: none;
8.          color: white;
9.          padding: 10px 30px;
10.         text-align: center;
11.         text-decoration: none;
12.         display: inline-block;
13.         font-size: 16px;
14.         margin: 5px 0px;
15.         -webkit-transition-duration: 0.4s; /* Safari */
16.         transition-duration: 0.4s;
17.         cursor: pointer;
18.     }
19.
20.     .button1 {
21.         background-color: white;
22.         color: black;
23.         border: 2px solid #555555;
24.     }
25.
26.     .button1:hover {
27.         background-color: #555555;
28.         color: white;
29.     }
30.
31.     #top, #left {
32.         background: #555555;
33.         position: fixed;
34.     }
35.
```

```
36.     #left {
37.         top: 0;
38.         bottom: 0;
39.         width: 25px;
40.     }
41.
42.     #left {
43.         left: 0;
44.     }
45.
46.     #top {
47.         left: 0;
48.         right: 0;
49.         height: 10px;
50.     }
51.
52.     #top {
53.         top: 0;
54.     }
55.
56.     .flytte_tekst {
57.         position: absolute;
58.         top: 0px;
59.         left: 40px;
60.     }
61. </style>
62. <body>
63.     <div id="left"></div>
64.     <div id="top"></div>
65.     <meta http-equiv="refresh" content="3" >
66.     <div class= "flytte_tekst">
67.             <h2>Skan nå kor-
    tet som <?php echo $_GET["navn123"]?> skal ha </h2>
68.         </form>
69.         <br><br>
70.         <form action="redirect.php" method="get">
71.         <button class="button button1" type="sub-
    mit" name= "hjemknapp" value= 100>Tilbake til hovedsiden</button>
72.         </form>
73.     </div>
74.
75. <?php
76.
77.     class TableRows extends RecursiveIteratorIterator {
78.         function current() {
79.             return  parent::current();
80.         }
81.     }
82.
83.     $servername = "192.168.1.3";
84.     $username = "admin";
85.     $password = "Test123";
86.     $dbname = "Bachelor";
87.
88.
89.     $sock = socket_create(AF_INET, SOCK_DGRAM, SOL_UDP);
90.     $msg = $_GET["navn123"];
91.     $len = strlen($msg);
92.     socket_sendto($sock, $msg, $len, 0, '192.168.1.2', 45678);
93.     socket_close($sock);
94.
95.
96.
97.
98.     try {
99.         $conn = new PDO("mysql:host=$server-
    name;dbname=$dbname", $username, $password);
```

```
100.
101.           $stmt = $conn->prepare('SE-
    LECT COUNT(navn) FROM adgang WHERE navn="'.$_GET["navn123"].'"');
102.           $stmt->execute();
103.
104.           // set the resulting array to associative
105.           $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
106.           foreach(new TableRows(new RecursiveArrayIterator($stmt->fetch-
    All())) as $k=>$a);
107.       }
108.       catch(PDOException $e) {
109.           echo "Error: " . $e->getMessage();
110.       }
111.
112.       try {
113.           $stmt = $conn->prepare('SELECT COUNT(exist_already) FROM error');
114.           $stmt->execute();
115.
116.           // set the resulting array to associative
117.           $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
118.           foreach(new TableRows(new RecursiveArrayIterator($stmt->fetch-
    All())) as $k=>$s);
119.       }
120.       catch(PDOException $e) {
121.           echo "Error: " . $e->getMessage();
122.       }
123.
124.       if ($s > 0)
125.       {
126.           header("Location: nytt_kort.php?errorcode=4");
127.       }
128.       else if ($a < 1)
129.       {
130.           try {
131.               $stmt = $conn->prepare("INSERT INTO adgang (ID, NAVN) VAL-
    UES ( NULL, '" . $_GET["navn123"]. "')");
132.               $stmt->execute();
133.
134.               // set the resulting array to associative
135.
136.               foreach(new TableRows(new RecursiveArrayIterator($stmt->fetch-
    All())) as $k=>$v) {}
137.           }
138.           catch(PDOException $e) {
139.               echo "Error: " . $e->getMessage();
140.           }
141.       }
142.           else if ($a == 1)
143.       {
144.               try
145.               {
146.                   $stmt = $conn->prepare('SE-
    LECT ID FROM adgang WHERE navn="'.$_GET["navn123"].'"');
147.                   $stmt->execute();
148.
149.                   // set the resulting array to associative
150.
151.                   foreach(new TableRows(new RecursiveArrayIterator($stmt-
    >fetchAll())) as $k=>$h) {}
152.                   echo $h;
153.                   if ($h != NULL)
154.                   {
155.                   header("Loca-
    tion: nytt_kort4.php?navn123=".$_GET["navn123"]."&ID=".$h."");
156.                   }
157.               }
158.               catch(PDOException $e)
```

```
159.                {
160.                    echo "Error: " . $e->getMessage();
161.                }
162.
163.        }
164.        else
165.        {
166.            echo "<script> location.href='nytt_kort.php?error-
    code=9'; </script>";
167.        }
168.
169.
170.
171.
172.    $conn = null;
173.    ?>
174.    </body>
175.    </html>
```

G. Nytt_kort4.php

```
1.  <html>
2.  <head>
3.    <title>Legg til kort</title>
4.  </head>
5.  <style>
6.      #top, #left {
7.          background: #555555;
8.          position: fixed;
9.      }
10.
11.     #left {
12.         top: 0;
13.         bottom: 0;
14.         width: 25px;
15.     }
16.
17.     #left {
18.         left: 0;
19.     }
20.
21.     #top {
22.         left: 0;
23.         right: 0;
24.         height: 10px;
25.     }
26.
27.     #top {
28.         top: 0;
29.     }
30.
31.     .flytte_tekst {
32.         position: absolute;
33.         top: 0px;
34.         left: 40px;
35.     }
36. </style>
37. <body>
38.     <meta http-equiv="refresh" content="5;url=/redi-
    rect.php?hjemknapp=100" />
39.     <div id="left"></div>
40.     <div id="top"></div>
41.     <div class= "flytte_tekst">
```

```
        <h2>Nå har <?php echo $_GET["navn123"]?> blitt lagt til i databasen med følgende kort ID: <?php echo $_GET["ID"] ?> </h2>
        <br><br>
        <b>Du vil nå bli videreført til hjemmesiden innen 5 sekunder.</b>
    </div>
</body>
</html>
```

H. Redirect.php

```
<html>
<head>
  <title>Redirect</title>
</head>
<body>

<?php
    $servername = "192.168.1.3";
    $username = "admin";
    $password = "Test123";
    $dbname = "Bachelor";
    if($_GET["hjemknapp"] == 100)
    {
        echo "<script> location.href='index.php'; </script>";

        try {
            $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
            $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

            $stmt = $conn->prepare("UPDATE sensormodus SET sensor0= '0' WHERE ID='A'");
            $stmt->execute();
            $stmt = $conn->prepare("UPDATE sensormodus SET sensor1= '0' WHERE ID='A'");
            $stmt->execute();

            // set the resulting array to associative
            $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
            foreach(new TableRows(new RecursiveArrayIterator($stmt->fetchAll())) as $k=>$v) {
                echo $v;
                }
        }
        catch(PDOException $e) {
            echo "Error: " . $e->getMessage();

        }

        $conn = null;
    }
    else if($_GET["red_1_2"] == 200)
    {

        if ($_GET["sensornummer"] == "")
        {
            echo "<script> location.href='nytt_kort.php?errorcode=2'; </script>";
        }
        else if ($_GET["sensornummer"] >= 0)
        {
            echo "<script> location.href='nytt_kort2.php?sensornummer=".$_GET["sensornummer"]."&sensorverdi=".$_GET["sensorverdi"]."'; </script>";
```

```
48.         }
49.             else
50.         {
51.             echo "<script> location.href='nytt_kort.php?error-
    code=9'; </script>";
52.         }
53.     }
54.     else if($_GET["red_2_3"] == 300)
55.     {
56.
57.         class TableRows extends RecursiveIteratorIterator {
58.             function current() {
59.                 return  parent::current();
60.             }
61.         }
62.
63.         try {
64.             $conn = new PDO("mysql:host=$server-
    name;dbname=$dbname", $username, $password);
65.             $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
66.
67.                 $stmt = $conn->prepare('SE-
    LECT COUNT(navn) FROM adgang WHERE navn="'.$_GET["navn123"].'"');
68.                 $stmt->execute();
69.
70.             // set the resulting array to associative
71.             $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
72.             foreach(new TableRows(new RecursiveArrayIterator($stmt->fetch-
    All())) as $k=>$v);
73.         }
74.
75.         catch(PDOException $e) {
76.             echo "Error: " . $e->getMessage();
77.         }
78.         $conn = null;
79.         echo "</table>";
80.         if ($v > 0)
81.         {
82.             echo "<script> location.href='nytt_kort.php?error-
    code=1'; </script>";
83.             $sock = socket_create(AF_INET, SOCK_DGRAM, SOL_UDP);
84.             $msg = "delete";
85.             $len = strlen($msg);
86.             socket_sendto($sock, $msg, $len, 0, '192.168.1.2', 45680);
87.             socket_close($sock);
88.         }
89.         else if ($v < 1)
90.         {
91.             echo "<script> loca-
    tion.href='nytt_kort3.php?navn123=".$_GET["navn123"]."'; </script>";
92.         }
93.             else
94.         {
95.             echo "<script> location.href='nytt_kort.php?error-
    code=9'; </script>";
96.         }
97.     }
98.     else if ($_GET["red_log"] == 400)
99.     {
100.            try {
101.                class TableRows extends RecursiveIteratorIterator {
102.                    function current() {
103.                        return  parent::current();
104.                    }
105.            }
106.                    $conn = new PDO("mysql:host=$server-
    name;dbname=$dbname", $username, $password);
```

```
                    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEP-
TION);

                    $stmt = $conn->prepare("SE-
LECT COUNT(bruker) FROM brukere WHERE passord='".$_GET["pas-
sord"]."' AND bruker='".$_GET["bruker"]."'");
                    $stmt->execute();

                    // set the resulting array to associative
                    $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
                    foreach(new TableRows(new RecursiveArrayIterator($stmt-
>fetchAll())) as $k=>$z) {}
                    echo $z;
            }
            catch(PDOException $e) {
                    echo "Error: " . $e->getMessage();
            }
            if($z > 0)
            {
                header("Location: index_admin.php");
            }
            else if ($z == 0)
            {
                header("Location: admin_logon.php?error=1");
            }
            else
            {
                header("Location: admin_logon.php?error=9");
            }
            $conn = null;
        }
    ?>
    </body>
    </html>
```

I. Search.php

```
<html>
<head>
  <title>Søk</title>
</head>
<style>
    .button {
        border: none;
        color: white;
        padding: 10px 30px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 16px;
        margin: 5px 0px;
        -webkit-transition-duration: 0.4s; /* Safari */
        transition-duration: 0.4s;
        cursor: pointer;
    }

    .button1 {
        background-color: white;
        color: black;
        border: 2px solid #555555;
    }

    .button1:hover {
        background-color: #555555;
```

```
28.         color: white;
29.     }
30.
31.     #top, #left {
32.         background: #555555;
33.         position: fixed;
34.     }
35.
36.     #left {
37.         top: 0;
38.         bottom: 0;
39.         width: 25px;
40.     }
41.
42.     #left {
43.         left: 0;
44.     }
45.
46.     #top {
47.         left: 0;
48.         right: 0;
49.         height: 10px;
50.     }
51.
52.     #top {
53.         top: 0;
54.     }
55.
56.     .flytte_tekst {
57.         position: absolute;
58.         top: 0px;
59.         left: 40px;
60.     }
61.         .year {
62.         position: absolute;
63.         top: 140px;
64.         left: 100px;
65.     }
66.         .month {
67.         position: absolute;
68.         top: 140px;
69.         left: 300px;
70.     }
71.         .date {
72.         position: absolute;
73.         top: 140px;
74.         left: 500px;
75.     }
76.         .hour {
77.         position: absolute;
78.         top: 140px;
79.         left: 700px;
80.     }
81.         .minute {
82.         position: absolute;
83.         top: 140px;
84.         left: 900px;
85.     }
86.         .boks {
87.         position: absolute;
88.         top: 198px;
89.         left: 35px;
90.     }
91.         .boks2 {
92.         position: absolute;
93.         top: 165px;
94.         left: 35px;
```

```
95.     }
96.         .card {
97.         position: absolute;
98.         top: 110px;
99.         left: 100px;
100.        }
101.            .error {
102.            position: absolute;
103.            top: 235px;
104.            left: 100px;
105.        }
106.    </style>
107.    <body>
108.
109.        <div id="left"></div>
110.        <div id="top"></div>
111.        <div class =year><b>ÅR</b></div>
112.        <div class =month><b>Måned</b></div>
113.        <div class =date><b>Dato</b></div>
114.        <div class =hour><b>Time</b></div>
115.        <div class =minute><b>Minutt</b></div>
116.
117.        <div class= "flytte_tekst">
118.
119.            <h2>Søk i bevegelsesloggen</h2>
120.            <br><br>
121.                Her kan du søke i bevegelsesloggen mellom to tids-
    punkt, samt spesifisere hvilket kort du ønsker å se på. Søkeforma-
    tet er <b>YYYY.MM.DD.TT.MM</b> <br>
122.            <br><br><br>
123.            <b>Fra:</b><br><br>
124.            <b>Til:</b><br><br><br><br><br><br>
125.            <b>Kort ID</b> (valgfritt)
126.            <div class= boks2 style= "width:1400px">
127.            <form action="search2.php" method="get">
128.                <input style="font-
    size:16px; height:40px;" type="text" name="Year_f">
129.                <input style="font-
    size:16px; height:40px;" type="text" name="Month_f">
130.                <input style="font-
    size:16px; height:40px;" type="text" name="Date_f">
131.                <input style="font-
    size:16px; height:40px;" type="text" name="Hour_f">
132.                <input style="font-
    size:16px; height:40px;" type="text" name="Minute_f">
133.            </div>
134.
135.
136.
137.            <div class= boks style= "width:1400px">
138.                <input style="font-
    size:16px; height:40px;" type="text" name="Year_t">
139.                <input style="font-
    size:16px; height:40px;" type="text" name="Month_t">
140.                <input style="font-
    size:16px; height:40px;" type="text" name="Date_t">
141.                <input style="font-
    size:16px; height:40px;" type="text" name="Hour_t">
142.                <input style="font-
    size:16px; height:40px;" type="text" name="Minute_t">
143.                <button class="button button1" type="submit">Søk</button>
144.                <input class = "card" style="font-size:16px; height:40px; posi-
    tion: 100px;" type="text" name="Card_ID">
145.
146.            </div>
147.            </form>
148.             <br><br><br><br><br><br><br>
```

```
149.            <form action="redirect.php" method="get">
150.            <button class="button button1" type="sub-
    mit" name= "hjemknapp" value= 100>Tilbake til hovedsiden</button>
151.            </form>
152.        </div>
153.        <div class = "error">
154.            <?php
155.                if ($_GET["errorcode"] == 1)
156.                {
157.                    echo "<h2>Det har skjedd en feil. ID-en du søkte på eksis-
    terer ikke i databasen. </h2><br>";
158.                }
159.                ?>
160.        </div>
161.    </body>
162.    </html>
```

J. Search2.php

```
1.  <html>
2.  <head>
3.    <title>Søk</title>
4.  </head>
5.  <style>
6.      .button {
7.          border: none;
8.          color: white;
9.          padding: 10px 30px;
10.         text-align: center;
11.         text-decoration: none;
12.         display: inline-block;
13.         font-size: 16px;
14.         margin: 5px 0px;
15.         -webkit-transition-duration: 0.4s; /* Safari */
16.         transition-duration: 0.4s;
17.         cursor: pointer;
18.     }
19.
20.     .button1 {
21.         background-color: white;
22.         color: black;
23.         border: 2px solid #555555;
24.     }
25.
26.     .button1:hover {
27.         background-color: #555555;
28.         color: white;
29.     }
30.
31.     #top, #left {
32.         background: #555555;
33.         position: fixed;
34.     }
35.
36.     #left {
37.         top: 0;
38.         bottom: 0;
39.         width: 25px;
40.     }
41.
42.     #left {
43.         left: 0;
44.     }
45.
```

```
    #top {
        left: 0;
        right: 0;
        height: 10px;
    }

    #top {
        top: 0;
    }

    .flytte_tekst {
        position: absolute;
        top: 10px;
        left: 40px;
    }
        .year1 {
        position: absolute;
        top: 140px;
        left: 80px;
    }
        .month1 {
        position: absolute;
        top: 140px;
        left: 280px;
    }
        .date1 {
        position: absolute;
        top: 140px;
        left: 480px;
    }
        .boks {
        position: absolute;
        top: 173px;
        left: 35px;
    }
        .boks2 {
        position: absolute;
        top: 140px;
        left: 35px;
    }
        .p {
        border-bottom: 3px solid grey;
        background-color: lightgrey;
    }
</style>
<body>
    <div id="left"></div>
    <div id="top"></div>
    <div class= "flytte_tekst">
    <h2>Søkeresultat</h2>
    Mel-
lom <?php echo $_GET["Year_f"] ?>.<?php echo $_GET["Month_f"] ?>.<?php echo $_
GET["Date_f"] ?> klokken <?php echo $_GET["Hour_f"] ?>:<?php echo $_GET["Mi-
nute_f"] ?> og
        <?php echo $_GET["Year_t"] ?>.<?php echo $_GET["Month_t"] ?>.<?php ech
o $_GET["Date_t"] ?> klokken <?php echo $_GET["Hour_t"] ?>:<?php echo $_GET["M
inute_t"] ?>
    bevegde følgende ID-brikker seg mellom de gitte rommene <br><br>
    <table>
       <tr><th>ID</th><th>Gikk til rom</th><th>Gikk fra rom</th><th>Dato</th><
/tr>
           <?php

               class TableRows extends RecursiveIteratorIterator {
                   function __construct($it) {
                       parent::__construct($it, self::LEAVES_ONLY);
                   }
```

```
107.
108.                     function current() {
109.                         return "<td class='p' style='width:150px;bor-
    der:1px solid black;'>" . parent::current(). "</td>";
110.                     }
111.
112.                     function beginChildren() {
113.                         echo "<tr>";
114.                     }
115.
116.                     function endChildren() {
117.                         echo "</tr>" . "\n";
118.                     }
119.                 }
120.
121.                 class TableRows2 extends RecursiveIteratorIterator {
122.                     function current() {
123.                         return  parent::current();
124.                     }
125.                 }
126.
127.                 $servername = "192.168.1.3";
128.                 $username = "admin";
129.                 $password = "Test123";
130.                 $dbname = "Bachelor";
131.                 if ($_GET["Card_ID"] == "")
132.                 {
133.                     try {
134.                         $conn = new PDO("mysql:host=$server-
    name;dbname=$dbname", $username, $password);
135.                         $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EX-
    CEPTION);
136.                         $stmt = $conn->prepare('SELECT ID,Position_new, Posi-
    tion_old, fulldate FROM bachelor_1 WHERE fulldate BETWEEN '
137.                                                 .$_GET["Year_f"].$_GET["Month_f
    "].$_GET["Date_f"].$_GET["Hour_f"].$_GET["Mi-
    nute_f"]."00 AND ".$_GET["Year_t"].$_GET["Month_t"].$_GET["Date_t"].$_GET["Hou
    r_t"].$_GET["Minute_t"]."59");
138.                         $stmt->execute();
139.
140.                         // set the resulting array to associative
141.                         $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
142.                         foreach(new TableRows(new RecursiveArrayIterator($stmt-
    >fetchAll())) as $k=>$v) {
143.                             echo $v;
144.                             }
145.                     }
146.                     catch(PDOException $e) {
147.                             echo "Error: " . $e->getMessage();
148.                         }
149.                         $conn = null;
150.                     echo "</table>";
151.
152.                 }
153.                 else
154.                 {
155.
156.                     try {
157.                         $conn = new PDO("mysql:host=$server-
    name;dbname=$dbname", $username, $password);
158.                         $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EX-
    CEPTION);
159.
160.                             $stmt = $conn->prepare('SE-
    LECT COUNT(ID) FROM adgang WHERE ID="'.$_GET["Card_ID"].'"');
161.                             $stmt->execute();
162.
```

```
                        // set the resulting array to associative
                        $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
                        foreach(new TableRows2(new RecursiveArrayItera-
    tor($stmt->fetchAll())) as $k=>$s);
                    }

                    catch(PDOException $e) {
                        echo "Error: " . $e->getMessage();
                    }

                    if($s > 0)
                    {
                        try {
                            $conn = new PDO("mysql:host=$server-
    name;dbname=$dbname", $username, $password);
                            $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERR-
    MODE_EXCEPTION);

                                $stmt = $conn->prepare('SELECT ID,Posi-
    tion_new, Position_old, fulldate FROM bachelor_1 WHERE fulldate BETWEEN '
                                                        .$_GET["Year_f"].$_GET["Mon
    th_f"].$_GET["Date_f"].$_GET["Hour_f"].$_GET["Mi-
    nute_f"]."00 AND ".$_GET["Year_t"].$_GET["Month_t"].$_GET["Date_t"].$_GET["Hou
    r_t"]
                                                        .$_GET["Mi-
    nute_t"]."59 AND ID='".$_GET["Card_ID"]."'");
                                $stmt->execute();

                                // set the resulting array to associative
                            $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
                            foreach(new TableRows(new RecursiveArrayItera-
    tor($stmt->fetchAll())) as $k=>$v) {
                                echo $v;
                            }
                        }
                        catch(PDOException $e) {
                            echo "Error: " . $e->getMessage();
                        }
                        $conn = null;
                        echo "</table>";
                    }
                    else
                    {
                        header("Location: search.php?errorcode=1");
                    }

                }

            ?>
                    <br><br><br><br><br><br><br>
            <form action="redirect.php" method="get">
            <button class="button button1" type="sub-
    mit" name= "hjemknapp" value= 100>Tilbake til hovedsiden</button>
            </form>
        </div>

    </body>
    </html>
```

K. Slette_kort.php

```
1.  <html>
2.  <head>
3.    <title>Slette kort</title>
```

```
</head>
<style>
    .button {
        border: none;
        color: white;
        padding: 10px 30px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 16px;
        margin: 5px 0px;
        -webkit-transition-duration: 0.4s; /* Safari */
        transition-duration: 0.4s;
        cursor: pointer;
    }

    .button1 {
        background-color: white;
        color: black;
        border: 2px solid #555555;
    }

    .button1:hover {
        background-color: #555555;
        color: white;
    }

    #top, #left {
        background: #555555;
        position: fixed;
    }

    #left {
        top: 0;
        bottom: 0;
        width: 25px;
    }

    #left {
        left: 0;
    }

    #top {
        left: 0;
        right: 0;
        height: 10px;
    }

    #top {
        top: 0;
    }

    .flytte_tekst {
        position: absolute;
        top: 0px;
        left: 40px;
    }
</style>
<body>
    <div id="left"></div>
    <div id="top"></div>
    <meta http-equiv="refresh" content="3" >
    <div class= "flytte_tekst">
        <h2>Scan kortet for å slette det</h2>
        <br>
        <form action="/redirect.php?hjemknapp=100" method="post">
            <button class="button button1" type="submit">Hjem</button>
```

```
71.         </form>
72.     </div>
73.
74. <?php
75.     $servername = "192.168.1.3";
76.     $username = "admin";
77.     $password = "Test123";
78.     $dbname = "Bachelor";
79.
80.     class TableRows extends RecursiveIteratorIterator {
81.         function current() {
82.             return  parent::current();
83.         }
84.     }
85.
86.     try {
87.         $conn = new PDO("mysql:host=$server-
    name;dbname=$dbname", $username, $password);
88.
89.         $stmt = $conn->prepare("UPDATE sensormodus SET sensor" . $_GET["sen-
    sornummer"]. "=". $_GET["sensorverdi"].  " WHERE ID='A'");
90.         $stmt->execute();
91.
92.         // set the resulting array to associative
93.         $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
94.         foreach(new TableRows(new RecursiveArrayIterator($stmt->fetch-
    All())) as $k=>$v) {
95.         echo $v;
96.         }
97.     }
98.     catch(PDOException $e) {
99.         echo "Error: " . $e->getMessage();
100.        }
101.
102.       try {
103.           $stmt = $conn->prepare("SELECT COUNT(rem_ID) FROM rem_adgang");
104.           $stmt->execute();
105.
106.           // set the resulting array to associative
107.           $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
108.           foreach(new TableRows(new RecursiveArrayIterator($stmt->fetch-
    All())) as $k=>$s) {}
109.       }
110.       catch(PDOException $e) {
111.           echo "Error: " . $e->getMessage();
112.       }
113.       if ($s == 1)
114.       {
115.               try {
116.           $stmt = $conn->prepare("SELECT rem_ID FROM rem_adgang");
117.           $stmt->execute();
118.
119.           // set the resulting array to associative
120.           $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
121.           foreach(new TableRows(new RecursiveArrayIterator($stmt->fetch-
    All())) as $k=>$f) {}
122.           }
123.           catch(PDOException $e) {
124.               echo "Error: " . $e->getMessage();
125.           }
126.           try {
127.
128.               $stmt = $conn->prepare("SE-
    LECT COUNT(ID) FROM adgang WHERE ID='".$f."'");
129.               $stmt->execute();
130.
131.               // set the resulting array to associative
```

```
                $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
                foreach(new TableRows(new RecursiveArrayIterator($stmt->fetch-
    All())) as $k=>$u) {}
            }
            catch(PDOException $e) {
                echo "Error: " . $e->getMessage();
            }
            if ($u > 0)
            {
                header("Location: slette_kort2.php?ID=".$f."");
            }
            else
            {
            header("Location: nytt_kort.php?errorcode=5");
            }
        }
        else if ($s > 1)
        {
            header("Location: nytt_kort.php?errorcode=9");
        }
    $conn = null;
    ?>
    </body>
    </html>
```

L. Slette_kort2.php

```
<html>
<head>
  <title>Slette kort</title>
</head>
<style>
    #top, #left {
        background: #555555;
        position: fixed;
    }

    #left {
        top: 0;
        bottom: 0;
        width: 25px;
    }

    #left {
        left: 0;
    }

    #top {
        left: 0;
        right: 0;
        height: 10px;
    }

    #top {
        top: 0;
    }

    .flytte_tekst {
        position: absolute;
        top: 0px;
        left: 40px;
    }
</style>
<body>
```

```php
<?php

    $servername = "192.168.1.3";
    $username = "admin";
    $password = "Test123";
    $dbname = "Bachelor";

    class TableRows extends RecursiveIteratorIterator {
        function current() {
            return  parent::current();
        }
    }

    try{
        $conn = new PDO("mysql:host=$server-
name;dbname=$dbname", $username, $password);

        $stmt = $conn->prepare('SE-
LECT NAVN FROM adgang WHERE ID="'.$_GET["ID"].'"');
        $stmt->execute();

        // set the resulting array to associative
        $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
        foreach(new TableRows(new RecursiveArrayIterator($stmt->fetch-
All())) as $k=>$a);
    }
    catch(PDOException $e) {
        echo "Error: " . $e->getMessage();
    }


    try {
        $stmt = $conn->prepare("DE-
LETE FROM adgang WHERE ID='".$_GET["ID"]."'");
        $stmt->execute();

        // set the resulting array to associative
        $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
        foreach(new TableRows(new RecursiveArrayIterator($stmt->fetch-
All())) as $k=>$r) {}
    }
    catch(PDOException $e) {
        echo "Error: " . $e->getMessage();
    }

    try {
        $stmt = $conn->prepare("DELETE FROM bache-
lor_1 WHERE ID='".$_GET["ID"]."'");
        $stmt->execute();

        // set the resulting array to associative
        $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
        foreach(new TableRows(new RecursiveArrayIterator($stmt->fetch-
All())) as $k=>$r) {}
    }
    catch(PDOException $e) {
        echo "Error: " . $e->getMessage();
    }
$conn = null;
?>
    <meta http-equiv="refresh" content="5;url=/redi-
rect.php?hjemknapp=100" />
    <div id="left"></div>
    <div id="top"></div>
    <div class= "flytte_tekst">
        <h2>Nå har <?php echo $a ?> og det tilhørende kor-
tet med ID   <?php echo $_GET["ID"]?> blitt slettet fra databasen </h2>
```

```
96.         <br><br>
97.         <b>Du vil nå bli videreført til hjemmesiden innen 5 sekunder.</b>
98.     </div>
99. </body>
100.   </html>
```