Høgskulen på Vestlandet

BACHELOROPPGÅVE

Smart Valley

HO2-300

Avdeling for ingeniørfag

19.05.2017

Tal ord 13988

Mathanraj Manivannan, Paul Thomassen og Hatem Awadallah.

Intern rettleder: Ole Gaute Hovstad

Ekstern Rettleder: Nils Westerheim

Eg stadfestar at arbeidet er sjølvstendig utarbeida, og at referansar/kjeldetilvisingar til alle kjelder som er brukt i arbeidet er oppgitt, *jf. Forskrift om studium og eksamen ved Høgskulen på Vestlandet, § **10**.*

Høgskulen på Vestlandet

STUDENTRAPPORT

Campus Førde, Svanehaugsvegen 1, 6812 FØRDE www.hvl.no

TITTEL Bacheloroppgave HO2-300	**RAPPORTNR.** 01	**DATO** 19.05.17
PROSJEKTTITTEL Smart Valley	**TILGJENGE** Åpen	**TAL SIDER**
FORFATTARAR Mathanraj Manivannan, Hatem Awadallah og Paul Thomassen	**ANSVARLEGE RETTLEIARAR RETTLEIARAR/syt** Intern rettleder: Nils Westerheim Ekstern Rettleder: Ole Gaute Hovstad	
OPPDRAGSGJEVAR **SFE**		

SAMANDRAG

Hovedmålet vårt er å lage et program for et digitalt system som har en framtidsrettet løsning. Videre får en illustrert de automatiserte målernes funksjoner, hvordan man får brukt styringssystemet til å spare energiforbruket til kunden og hvordan nettselskapene oppnår en jevnere energibelastelse i strømnettet.

SUMMARY

Our main target is to make a program for a digital system which has a solution for the future. Secondly will we be able to illustrate the automatic gauges's functions, how we are able to use the control system to save the customer's energy consumption and how the network companies reach a smoother energy level in the current system.

EMNEORD

Smartteknologi, smart valley, ams, programmering, hovedprosjekt, smart grid.

1.0 Forord.

Rapporten vår er skrevet og jobbet med i sammenheng med hovedprosjektet vårt våren 2017, ved ingeniørstudiet ved Høgskulen på Vestlandet. Dette er jobbet av tre studentrt som har utført prosjektet sammen. Dette prosjektet er gitt av SFE. Oppgaven vår handler om at vi lager en automatisert programmeringsmodell til SFE og den omhandler også om automatiserte strømmålere(AMS).

Programmeringsbiten er automasjonsbiten, og AMS er elkraftbiten. Siden vi er studenter fra begge linjene, ble oppgaven bygget opp slik. Rapporten vil derfor gjennomgå programmering og AMS + teorien bak disse.

Utover arbeidet har vi fått hjelp fra våre rettledere og mange andre personer. Vi vil takke SFE for at de ga oss dette prosjektet som avslutningsprosjekt ved HISF. Dessuten vil vi og takke våre rettledere, Ole Gaute Hovstad (SFE),Nils Westerheim (HISF), og andre faglærere som har hjulpet oss under arbeidet. Vi vil og takke andre selskap som vi har tatt kontakt med i løpet av perioden.

Førde

Mathanraj Manivannan Hatem Awadallah Paul Thomassen

2.0 Sammendrag.

Denne rapporten er skrevet i sammenheng med avsluttende hovedprosjekt på studiet innen ingeniørfag ved Høgskulen i Sogn og Fjordane våren 2017. Oppgaven vår ble gitt av SFE. I denne oppgaven skal vi lage en programmert styringsmodell for en skole. Videre har vi og studert på AMS, som omfatter elkraftbiten. Siden vi er fra to studielinjer i prosjektet, har vi valgt å fokusere på programmeringsbiten for automasjonsretningen og AMS og smartteknologi for elkraftretningen.

Hovedmålet for dette prosjektet omhandler å effektivisere strømbruken ved skolene, og derfor kan vi koble disse to retningene sammen. Ved å lage et effektivt program i JAVA og å studere nærmere innen AMS, har vi da prøvd å løse problemstillingen vår og å nå dette målet. Prosjektet vil da omhandle alt frå teorisamling, programmering, drøfting og konkludering.

I prosjektet vil vi da få et resultat av den ferdige progammerte styringa vår og AMS studering. Rapporten inneholder nødvendig dokumentasjon rundt alle deler av modellen.

I rapporten har vi presentert hvordan vi har jobbet, både frå teorisamling, utførelse og til slutt drøfting. Grunngivelse, drøfting og valg knyttet til den teoretiske biten er gjennomført. Disse knytes til oppgaven vi fikk av SFE og er drøftet mellom SFE og oss i prosjektgruppa. Vi har og lagt til rette å ta hensyn til vilkårene satt av HISF.

3.0 Innholdsliste

4.0 Innledning.

Smart valley er et prosjekt av SFE som har gått i flere år. Overdrivet bruk av strøm, har ført til at energibransjen har blitt tvinget til å bygge dyre utbygginger av nettet. SFE prøver å finne nye metoder til å begrense strømbruken i hjemmene. [1]

Det lille tettstadet Hyen ble da prøvekanin, og her vil SFE prøve sitt prosjekt. I disse boligene ble det da installert automatiske strømmålere for å prøve ut prosjektet. Som en kan se ut av prosjektnavnet «Smart Valley», så går det ut på å bruke energi og strøm på en smart og effektiv måte, slik at dette er bærekraftig. Dette er med tanke på miljø, ressurser og økonomi.

Utgangspunktet for dette prosjektet kommer av hvor kostbart dette er for samfunnet. Om alle i småbygder kommer hjem rundt ettermiddagen, setter på komfyr, elbilen på lading, bruk av pc og gjør alt annet mulig, vil dette føre til at en stadig må bygge flere kraftlinjer til småbygdene. Det at stadig elektriske verktøy blir mye sterkere, får og store konsekvenser. Ved å finne smarte løsninger, som med automatiske straummålere, vil det gi oss ny kunnskap om hva som skjer rundt oss.

Målet med dette prosjektet, er å utnytte fleksibiliteten til kundene. I framtiden kan man se for seg at ny teknologi vil kommunisere med hverandre, og man kan således se for seg hvordan denne teknologien kan automatisk styre strømforbruket.

Utfordringen i dag er at alle skal bruke alt samtidig. Ved å ikke ta hensyn til hverandre, kan dette føre til dyrbare konsekvenser. Det er disse problemstillingene man prøver å løse med Smart Valley prosjektet.

Rapporten er skrevet med hensyn til dette prosjektet. Her blir det lagt vekt på at rapporten skal være opplysende. Hovedfokuset vårt knytes til den faglege biten både i automatiseringsteknikk og elkraft. Vi begynner med en informativ del av oppgaven, så teoretisk del, deretter drøftende/løysingsdel og til slutt en konklusjon av hele oppgaven.

5 Målsettinger

5.1 Hovedmål

Hovedmålet vårt er å lage et program for et digitalt system som har en framtidsrettet løsning. Videre får en illustrert de automatiserte målerne sine funksjoner og hvordan man får brukt styringssystemet til å spare energiforbruket til kunden og nettselskapene oppnår en jevnere energibelastelse i strømnettet.

5.2 Delmål

- Funksjonsbeskrivelse.
- Lage et temperaturreguleringsprogram.
- Funksjonen til AMS i forhold til avregning hos nettselskapene i framtiden.
- Hvordan er energipris og nettleiget bygget opp ? (i dag og i framtida.)
- Økonomi- og miljøvinning.
- Dokumentasjon og sluttrapport.
- Pressemelding.
- Presentasjon.
- Rapport.
- Nettside

6.0 Metode

Vi vil lage en virtuell modell med regulering av temperatur i et skolebygg.
Det er ei ønske frå byggeiger at han kan ha større kontroll over driftskostnadene. Dette gjøres ved bruk av det nye automatiserte systemet som knyttes til nettselskapets forbruksdataer.

Systemet skal kunne styres med bruk av timeplan der man kan koble inn og ut de ulike klasserommene inkludert svømmebasseng. Dette programmet tenker vi å lage i JAVA, der brukeren selv kan regulere temperaturen etter sine egne øsnker. Dette vil da dekke automasjonsbiten av prosjektet.

I elkraftbiten av prosjektet, vil vi bruke smart-teknologi(saman med AMS) til å utvikle en tariffmodell som er mer rettferdig for den kunden som vil være aktiv med å oppnå spart forbruk og å være miljøbevisst. [2]
Vi vil belyse hvilke utfordringer nettselskapene har med å få forbrukere aktive i forhold til utviklingen av smartgridsatsingen[3]
Målet med smartgridsatsingen er at samfunnet ønsker å ha ein sikker strømforsyning. Siden forbruket øker, blir det behov for smarte og fleksible nett.
Dette bygger på at dagens system hos kraftbransjen er manuelt styrt. Det er behov for mer avanserte styresystem for å imøtekomme og samkjøre støm som blir levert fra småkraft. Dette gjelder også fornybar energi som blir produsert fra sol, vind og vann.
En av utfordringene til kraftbransjen, er å tilpasse seg innføringen av smart-teknologien for å oppnå optimal nettnytte. [4]

7.0 Teori

7.1 Litt historie om java

Java er et programmeringsspråk utviklet av James Gosling. I starten var det ikke laget med internett i mentet. Istedenfor forestilte James Gosling og Sun Microsystem små samanhengende enheter som skulle kommunisere med hverandre. Slik brukte java programmeringsspråket mer tid på de mer avanserte oppgavene av nettverksprogrammering enn andre utfordrere [5]. Nettverksprogrammering er alltid en utfordring, men javaprogrammering tok dette til nye høyder ved å forenkle oppgavene innenfor programmering. Første tilveksten av java skjedde i januar 1995, og var originalt kalt «oak» men måtte bytte navn på grunn av brudd på varerettigheter. Den mer kjente JavaBeans. kom på banen i Java 1.1 i februar 1997. Gjennom årene har java fått andre kallenavn, som JDK 1.2 (også kalla JDK 1.2.) I dag har vi JDK 1.9 (også kjent som Java 9).

Java ble opprinnelig laget for å ha en følelse av CC+ språket, men er enklere å bruke, og tvinger til å bruke objektorigentert programmering (OOP). OOP er programmringspråkmodel organisert rundt objekter enn event og data istedenfor logikk. [6] Java kan og brukes til å lage program som kjører i en enkel datamaskin, eller kjøres gjennom servere og klienter i et nettverk.

7.2 Element og prinsipp i java.

Det er vanskelig å skrive en enkel årsak til hvorfor java har blitt så allestedsnærværende. Her er noen årsaker.

- Program, som er laget i java, tilbyr portabilitiet, evnen til å bli brukt i andre operativsystem enn der programmet ble laget. Koden er kompilert til det java kaller «bytekode», som kan bli kjørt i enhver nettverk i en server eller klient som har en Java Virtual Machine (JVM). JVM, en gjennomføring av Java Virtual Machine Specification, tolker kompilert Java binære koder (som kalles bytekode) for en dataprosessor, slik at den utfører javaprogrammets struksar.[7] Dette gjør ikke andre program som CC++, COBOl, hvilke kompilerer det til en binær fil som er mye vanskeligere.

- Java er objektorientert. En objekt tar fordel av å være en del av en klasse av objekter og arve koder felles i klassen. Evnen til å utvikle språk, lage fra bunnen av ved objektorientering, gjorde java til et spennende plattform i å programmere.
- Utviklere kan lære java fort. Syntaksen er ganske lik CC++ og java er også ganske lett å lære. Hvis man har bakgrunn fra C, så er dette nyttig.

7.3 Java GUI.

Å skrive sine egne grafiske klasser er som å oppdage hjulet på nytt. Disse grafiske klassene er veldig komplekse og involverer mange avanserte designmønster. Vi har to ulike sett med Java Applications programmering Interface (API) for grafisk programmering [8]. Det er AWT (Abstract Windowing Toolkit) og Swing.

1) AWT API var først presentert i JDK 1.0. Det meste av AWT komponenter har blitt foreldet og erstattet av Swing komponenter.
2) Swing API, en mye mer omfattende sett med grafiske biblioteker som forbedrer AWT ganske klart og ble introdusert i JDK 1.2.

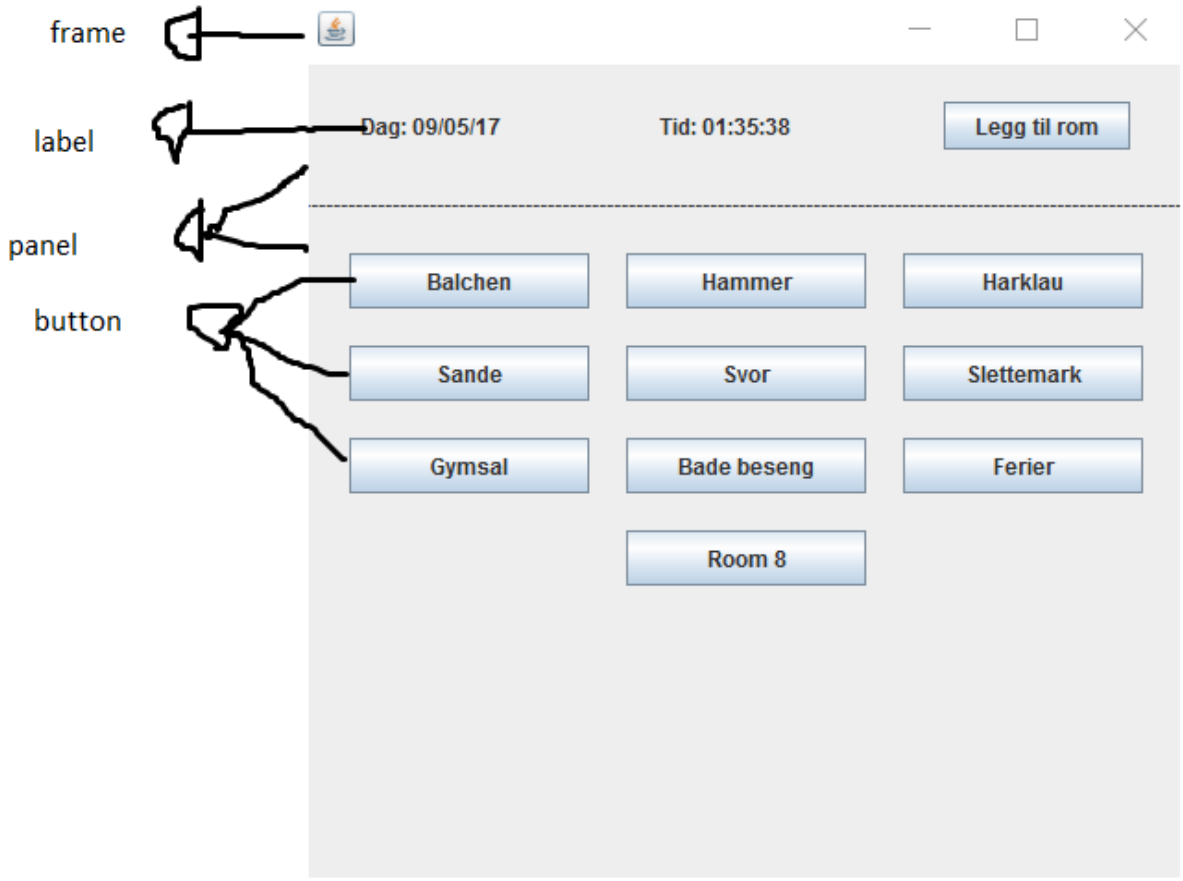
Utenom disse to, er det og laget andre grafiske API som virker med java som Eclipse's Standard Widget Toolkit(SWT og Google Web Toolkit(brukt i android telefonar).

7.3.1 AWT

I AWT brukes normalt det kun to pakker. Allikevel eksisterer det enorme 12 pakker, fordelt på 370 klasser! De to ulike pakkene i AWT er Java.awt og java.awt.event.

1) Java.awt package inneholder kjernen i AWTs grafiske klasser.
- Her har vi GUI komponent klasser, som «Button», «TextField» og «Label».
- Vi finner GUI Container klases som inneholder «frame» og «panel».
- Layout managers, som flowlayout.
- Custom graphics klasser som «color» og «font».
2) Java.awt.event package, støtter klasser som actionevent og mousevent. Den har og Event listener interfaces som actionlistener, mouselistener.

Det er to typar av gui elementar. Desse er components som «button», «label» og «textfield» og containers som frame and panel. Kontaineren held komponentane i spesifikke layout.



Figur 1 Containers og class.

I figuren over kan man se tre kontainere: en frame, og to panels. I framen har man oftest en tittel, logo og et alternativ til å minimere, maksimere og lukke. Det er også et valgfri meny og innholdsdisplay. Panel er det rektangulere området som brukes til å gruppere kontainere i en layout som man ønsker. I eksempelet over har vi, 11 Buttons(trykkes for å utføre noe) og 2 labels(som gir informasjon). Komponentene skal holdes i kontainere, og man må identifisere dette for å holde på komponentene. Hver kontainer har også en form for metode som en kaller for add(Component c).

7.3.1.1 AWT Container Classes

Primære kontainere.

I en hver gui program, har en toppkontainer. Her brukes som regel Frame, Dialog og Applet. I Frame har vi informasjon som viser til tittelen på programmet, et ikon, minimering/makismiering og lukk knapper, en meny som er valgfritt og innholdsdisplay.

En AWT dialog brukes til å samhandle med ulike brukere. Her har vi tittelikon, tittel, lukkeknapp og innholdsdisplay.

En AWT Applet (i pakke java.applet) er den primære kontaineren for applet, som er javaprogrammet som kjører i innsiden av en nettleser.

Sekundære kontainere.

Sekunderære kontainere plasseres på innsiden av top-level kontainer eller andre sekundere kontainere. AWT har desse sekundære kontainere:

Panel: Dette er en rektangulær boks under høyere nivå kontainer, brukt til layout(en set av relaterte gui komponenter som f.eks grid eller flow).

Scrollpane: Dette gir automatiske horisontale og vertikale bevegelser for komponenter.

AWT komponent klasse.

Tre steg er nødvendige for å lage og plassere en GUI komponent.

1) Man må deklarere komponenten med et navn (identifikasjon).
2) Konstruere komponent.
3) Identifisere kontaineren(som Frame eller Panel), som er designa til å holde komponenten.

I AWT har vi mange ferdiglaga Gui komponenter som kan brukes om og om igjen i pakka java.awt. Her brukes gjerne mange hyppig, og de som vi bruker er veldig ofte «button». «textfield», «labels og «checkboxgroup».

7.3.1.2 AWT GUI komponent: java.awt.label

En java.awt.label gir en beskrivende tekststreng. Man må legge merke til at System.out.print() skriver ut til systemets konsoll og ikke til grafikkskjermen. Man kan da bruke Label til å «lable»(merkelappe) en annen komponent (f.eks textfield) eller gi tekstbeskrivelse.

Konstruere en komponent og addere en kompoent til en kontainer.

For å få dette til, er det nødvending å gjøre dette i tre steg. Først må en deklarere komponenten med en namn (identifikasjon). Deretter må man konstruere komponenten ved å påkalle en passande konstruktør via en ny operatør. Til slutt må man identifisere kontaineren (som Frame eller Panel) designa for å holde komponenten. Kontaineren ligg således til komponenten via en metode aContainer.add(aComponent).

7.3.1.3 AWT GUI Komponent: java.awt.button.

Dette er en GUI komponent som gjennomfører en bestemt progammert handling ved klikking.

7.3.1.4 AWT GUI Component: java.awt.Textfield

Dette er en enkel linje med tekstboks, der brukere kan skrive inn tekster. Ved å trykke enter på tekstfield objekter, skjer en form for actionevent (en programmert handling).

7.3.1.5 AWT Event-Handling(hendelsehandtering).

Som mange andre programmer bruker java det såkalla «event» drivende programmeringsmodellen for å behandle «event-handling.» I slik programmering blir en del av hendelsehåndterande koder gjennomgått, når ein hendelse blir trigget av en eller annen form for reaksjon av bruker. Dette kan skje ved at en trykker på datamusen, eller f.eks ved å trykke enter.

AWT-event-handling klasser blir holdt i en pakke kalt java.awt.event. I dette er det et samspill mellom tre objekter i Event-Handlinga. Disse objektene er ei source(kjelde)-, listener-(lytte) og ein eventobjekt.

Sourceobjektet som «button» og «textfield» samhandler med brukareren. Idet denne utløses, skapes det et eventobjekt. Dette eventet vil da sende beskjed til alle registrerte listenerobjekter, og en passande eventbehandlarmetode av listeners(lyttarar) blir kalt tilbake for å få svar. Enklare forklart kan vi si at det å løse ut source, fører det til en event til alle dens listerners, som påberopa en passende event av listeners.

7.3.1.6 Layouts.

En kontainer bruker «layout-manager» for å plassere komponenter i java. AWT bringer følgende layouter i java: FlowLayout, GridLayout, Borderlayout, Boxlayout osv. Disse er da lagret i pakken Java.awt. Swing adderte flere pakker i javax.swing.

Flowlayout.

I Flowlayout er komponentene plassert fra venstre til høyre og dette er da inni kontainere. Arrangementet starter først i rekkefølgen de ble lagt til, og idet den ene raden er fylt, så begynner man på neste. Dette betyr at bredden til displayvindauget er den som har størst betydning for layoutens utseende.

GridLayout.

I Gridlayout er komponentene plassert i et nett (matrise) av rader og kolonner inni kontaineren. Komponentene er da lagt til fra venstre til høyre, topp til bunn metode, i rekkefølgen de er addert.

Borderlayout.

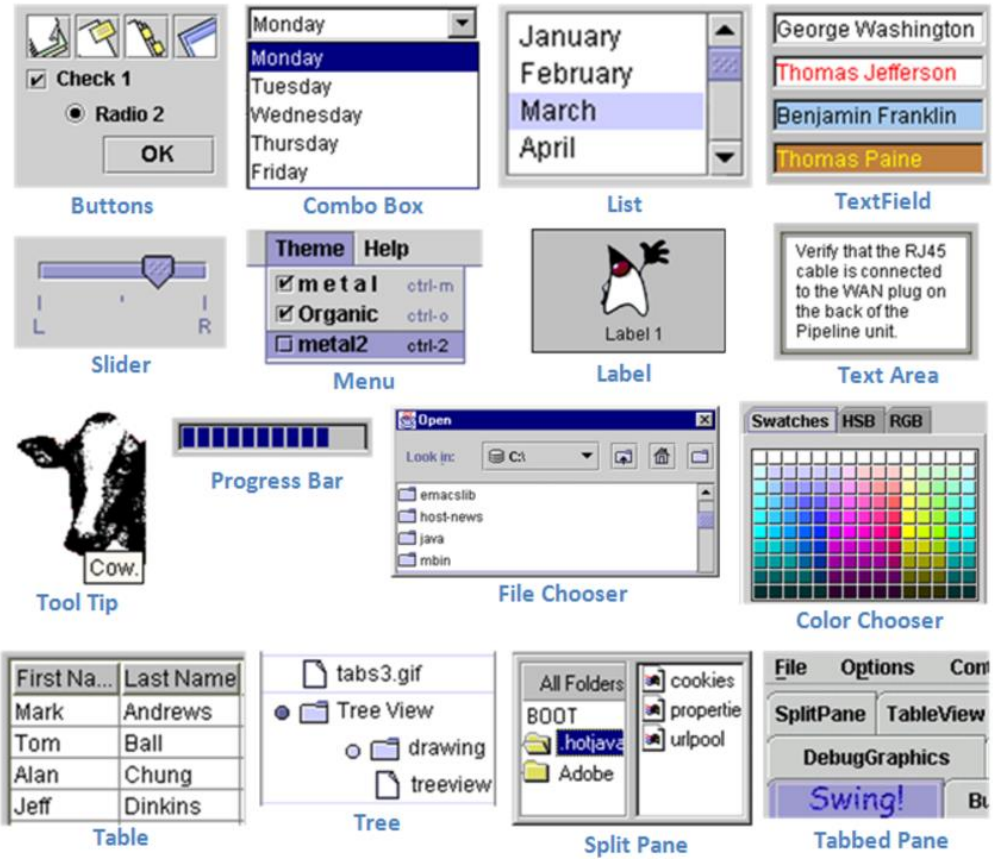
Her blir komponenten plassert i en kontainer som er laget av fem regioner. Disse er «North», «South» «East», «West», og «Center». Hver av disse kan kun inneholde en komponent. Man trenger ikke å plassere en komponent i alle regionene.

BoxLayout.

I BoxLayout plasserer vi komponentene i en enkel rad eller kolonne.

7.3.2 Java Swing.

Swing er en enklare form for API, og dette brukes til å lage GUI for java. Byggemønseret i Swing gjør at det er enkelt å legge grafikk til kode og dette gjør en uten å endre mye på applikasjonen. Ved å bruke swing er det da lett å lage Button, textfield, checkbox, labels tabell osv. [9] Swing gir et enormt mengde med muligheter og gjenbrukbare GUI komponenter.



Figur 2 Java Swing muligheiter [8]

På figuren over kan man se mange av mulighetene man kan bruke i swing.

Noen av hovedfunksjonane til Swing er:

1) Swing er skrevet i reint Java (utenom noen klasser) og er derfor helt bærbart.
2) Komponentene i Swing er lettvektelige, mens komponentane i AWT er tungvektelige (med tanke på systemressurs utnyttelse).
3) Swing komponentene støtter «Pluggable look and feel» i Java. Look and feel er mekanismen til å endre design i et program, som farge, form, layout osv. [10]
4) Swing kan brukes uten mus, og kan opereres kun med tastatur.

5) Swing bruker AWT event-handling klasser og adderer nye klasser i pakken javax.swing.event. Swing bruker og AWTs layout manager, for å styre layout.

Normal måte å byggje på swing på, er å bruke JFrame og JPanel i et bestemt mønster. Prosedyren går ut på å ha en klasse A som igjen arver JPanel. Klasse A fyller seg selv med komponent(tekst, buttons, labels osv), og danner enten hele vinduet eller deler av dette.

En instans av A, blir altså komponent utformet slik man ønsker vinduet skal se ut. Men A vil da ikke lage et vindu som vi kan se og røre med. Dette er da J-Framets oppgave. En instans av JFrame blir da presentert som virkelige vinduet på skjermen vår. I J-Frame, kan man og legge til komponenter slik som i et panel. Forskjellen er likevel at for å skape struktur, så fylles hele vinduet(framen) med panelet A. Hvis man ønsker button nederst i vinduet, blir den laget i panelet a, ikke framen.

7.4 Smarte nett

7.4.1 Innledning

Begynnelsen av den smarte tidssalderen er over oss.

Energibransjen er en av de siste til å ta overgangen fra sin opprinnelige struktur fra slutten av 1800-tallet til et moderne system som gir kraftsselsakpene og deres kunder sanntidsinformasjon som kan føre til bedre beslutningsprosesser og lavere samlet kostnad for kunder.

7.4.2 Smart

Ordet smartGrid ble brukt i begynnelsen av 1990-tallet men begynte å brukes i 2005. i begynnelsen var det særlig fokus på å gjøre det med robus, og dette var trigget av det man hadde opplevd av abrudd i nordamerika.

Visjonen ble da mer adaptiv og responsen ble automatisert ved utfall og unormale situasjoner i kraftsystemet. De siste årene har utslipp av klimagasser fått så stor fokus i Smart grid-konseptet, dette gjennom at man ønsker større elektrisitetprosuksjon av nyere kilder.

I de seneste årene er da Smart Grid brukt som et ord til å omfatte mange problemstillinger. Dette inngår både i planlegging, drift, vedlikehold av det elektriske energisystemet osv.

Smarte nett er ikkje bare kraftnette, men dette gjelder hele systemet. Her benyttet toveivskomunnikasjon til å styre annlegg og apperater som er koblet til nettverk med internett. Da kan man kalle smart nett som en slags forbindelse mellom kraftnett og internett, hvor funksjonen kommer fra nettet til kraftsystemet. [11]

7.4.3 Smarte nett.

Det er en måler i et sikringsskap som leser av mplinger, og dette sendes videre til nettselskapet. Nå til dags må kunden lese dette selv, en gang i sesongen, og avregningen er basert avtaletypen. Smarte målere gjør dette mye mer nøyaktig. Kunden betaler kun for strømmen han har brukt.

Ved å ha elektriske apperater med en ip-adresse, vil da måleren kunne hente informasjonen om strømprisene. Disse apperatene vil da kun skrus på når åprisene er gunstige.

Nettselskapene vil da få mer kontroll over krafsystemet, samtid at dette skjer effetivt og enkelt. Her får de gjerne oversikt over hvor mange som bruker strøm, og hvordan fordele dette utover, og kan da fordeles jevnt mellom. Da er det ikke behov for å bruke store, tykke ledninger.

Når en husstand har en smart strømmåler og elektriske apparater med en IP-adresse, vil måleren kunne hente ut informasjon om strømprisen. Apparatene vil kunne ha innstillinger for kun å skru seg på automatisk når strømprisen er på ideelle nivå.

Det postitive med smarte nett er at vi i dag står overfor en global miljøkrise. Siden behovet for energi økes i framtiden, vil utbygginger av tykke ledninger å slikt være lite miljøvennlig, og smarte nett er da en miljøvennlig prosess. [12]

Med digitalmålerinstallasjon/AMS som når alle husstander 1. januar 2019 blir dette den nye normalen i stedet for unntaket. Den smarte måleren er den første av mange nye intelligente enheter som påvirker kraftselskapene. Inntoget av avansert kommunikasjon, e-handel og maskin-til-maskin (M2M) kommunikasjon vil over tid oppmuntre kraftselskapene over hele landet til å etablere nye forretningsmodeller. Kraftselskapene vil også bli "smartere" med nye ansatte som kan både elkraft og IKT.

7.4.4 Hva er fordelene?

Med nye smarte nettteknologier og sensorer kan kraftselskapene raskt finne ut hvor brudd forekommer i systemet, noe som vil bidra til å gjenopprette forsyningen raskere og til en lavere pris. De vil også kunne trekke elektrisk kraft fra de kraftverkene som produserer på det mest kostnadseffektive nivåene til til enhver tid. Smarte målere i boliger og bedrifter kan analysere bruksmønstre og identifisere ideelle tider for å kjøre tørketrommel i kjeller eller maskiner i bedrifter. Det legger mindre press på de lokale kraftselskapene når de har sine lasttopper.

Kundene slipper all måleravlesing fordi trådløs teknologi kan overføre alle relevante data. Å ha den riktige softwaren og det rette utstyret vil bidra til at smartnettverket blir mer effektivt.

Kunder kontakter vanligvis kun bare kraftselsakpene når de har regningsklager, strømbrudd eller spørsmål om flytting. Det pågår en debatt i bransjen om behovet for å engasjere boligkunder, spesielt fordi det er vanskelig å nå energieffektivitetsmål, og gir betydelig mindre energireduksjon enn det gir å jobbe med større kunder som bedrifter og foretak. NVE gjør undersøkelser rundt temaet forbruker fleksibilitet. Sitat:

«Lokal fleksibilitet som alternativ til nettinvesteringer

«Økende bruk av effektkrevende apparater gjør at forbrukere krever mer effekt fra nettet enn før. Samtidig skjer en økende andel av kraftproduksjonen med fornybare teknologier som ikke kan styres eller lagres etter behovet for kraft (sol, vind, bioenergi etc.). Denne utviklingen øker behovet for fleksibilitet og kan gi knapphet på overføringskapasitet i distribusjonsnettet.

Behov for økt nettkapasitet møtes i dag hovedsakelig gjennom utbygging av nett, selv om kapasiteten kun er anstrengt i et fåtall timer i løpet av året. Alternativt kan forventninger om effektknapphet håndteres via utkobling eller reduksjon av forbruk i disse timene. Dette omtales som forbrukerfleksibilitet. Lagring av elektrisitet (batterier) eller fleksibilitet fra produksjon kan benyttes til samme formål.» [13]

Alle kunder, inkludert private, er avgjørende for digitalisering av elnettet. Uten de vanlige forbrukerne kan et kraftselskap vanskelig forsvare den investeringen som er nødvendig for å bringe distribusjonsnetttet til teknologinivået som er nødvendig for å møte kundenes nåværende og fremtidige behov. Det er ikkje bare måleravlesing som skal digitaliseres, det handler også i stor grad om at myndighetene ønsker at infrastrukturen kraftselskapene utgjør må opp på et høyere nivå for å møte kundenes behov og krav og tilrettelgge for ny industriutvikling. Toveis strømflyt og kommunikasjon blir stadig viktigere for moderne liv og næringsliv.

Tenk på telefonindustrien. For femten år siden hadde vi ikke mobiltelefoner, og ikke mange ville ha drømt om at de ville stenge fasttelefonen i eget hjem. I dag har mange bare mobiltelefoner, og mange bruker dem først og fremst til teksting. Mobilkundene velger abonnent som passer deres behovTelekomindustrien forvandlet seg helt på mindre enn to tiår. Mange ting akselererte disse endringene, inkludert deregulering av monopol og teknologi, men også kundenes øsnke om betre og meir komfortable tjenester. Dette må vi prøve å videreføre under utrullingen av det digitale strømnettet.

7.4.5 Hvordan gjør vi det?

Kraftselskapene og nettselskapene står klar til å starte sin egen digitale revolusjon, men hvordan? Brudd på telefonlija kan være ganske irriterende, men sjeldan livstruende, men det er ikke mulig å ikkje ha strøm. Kraftbransjen er tungt regulert, og endringer skjer derfor langsomt. Det globale elektriske nettverket har blitt kalt verdens mest kompliserte maskin, og det ble bygget for å håndtere kunder fra det 20. århundre, ikke det 21. Store investeringer i infrastruktur vil alltid vises igjen på sluttbrukernes faktura og det gjør at vi må jobbe ekstra hardt for å motivere og få med oss disse.

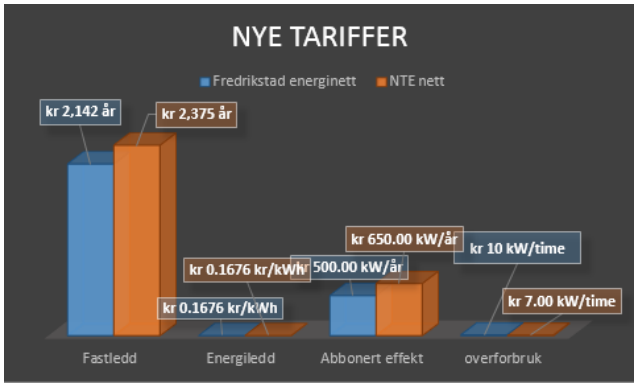
Energi Norge, som er en paraplyorganisasjon for nettselskap og kreftleverandører har skrevet en veileder for å komme i gang. Her er et lite utdrag.»

Arbeidsmetodikk

«*I samfunnet generelt introduseres vi stadig for nye løsninger og teknologier. Dette gjelder også for nettselskapene, men det er viktig at nye løsninger rettes mot de mest grunnleggende problemstillingene man står overfor i eget nett - de som gir størst nytteverdi og bidrar til økt kostnadseffektivitet. Enkelte problemstillinger gjelder både nasjonalt og lokalt, eksempelvis økende effektuttak, økende gjennomsnittsalder på nettet, krav til forsyningssikkerhet, nye krav til spenningskvalitet og krav til datasikkerhet og personvern. Andre problemstillinger vil variere mellom ulike nettselskaper, ut fra geografisk sammensetning, vær og klimatiske forhold, nettstruktur og tekniske løsninger, kundesammensetning og nye problemstillinger som intermitterende laster og plusskunder / lokal energiproduksjon. Uavhengig av hvilke problemstillinger man står overfor kan det være hensiktsmessig å ta utgangspunkt i en problemløsningsmodell bestående av fire hovedfaser slik det er illustrert i figuren nedenfor. De fire fasene er nærmere beskrevet i etterfølgende underkapitler.» [14]*

7.4.6 NYE TARIFFER, KUNDESEGMENTERING SPØRREUNDERSØKELSE

Fremtiden for smart energi er nå. Over hele landet snakker forbrukerne oftere om smartnettverket, fra forbedret energieffektivitet, innkjøp av smarte energiprodukter, raskere respons på strømbrudd, økt pålitelighet og penger –spart på månedlige energiregninger.



Figur 3 Nye tariffer

Mer sofistikerte tariffer gir kundene fleksibilitet i hvordan de velger å dempe sitt energiforbruk. Denne fleksibiliteten fører til mer deltakelse i å styre eget forbruk fordi kunder kan bestemme hvilke tiltak som passer best til deres behov. Vi kan få webportaler på nettbrett og mobil som styrer når det lønner seg å lade elbilen eller varme varmtvannet. Vi kan også tenke oss videre at batteriteknologi kan gjøre det mulig å lade opp batterier hjemme når prisen for strøm er lav og så bruke den når prisen er høy. Dets amme gjelder for eksempel for de nye plusskundenen som lager strøm sjølv med solceller og vindmøller, de trenger software som styrer produksjonen og leverer enten til eget bruk, lade eget batteri eller lvere inn på nettet når prisen er riktig. Krafselskapene på sin side vil med dagens IKT kunne ha elektonisk kommunikasjon med kunden og tilgang til hva som skjer å ha en mye betre kontroll med laststyring enn i dag. [15]

Elektrisitetskunder sitter altså i krysset mellom big data, nettportaler, nettmodernisering investeringer, innovative smarte elektriske apparater og programmer. Med alle fremskritt innen elektrisk kraft har smarte nett-aktører mulighet til å engasjere kunder, forme relasjoner og produsere programmer som tilfredsstiller kundenes ønsker for strømforvaltning. I USA jobber de mye med hvordan de skal markedsføre det smarte nettet til folk og en organisasjon kalt SGCC har med utgangspunkt i tanken om at en engasjert forbruker er en fornøyd forbruker ønsket og forløse dette engasjementet med å se på motivasjon og holdninger de enkelte kundegruppene har. Kundene er delt opp i fem segmenter som vi skal komme tilbake til. [16] De har laget en undersøkelse og vi skal gjengi noe av den. Bakgrunnen for dette fokuset frå SGCC og amerikanske myndigheter er at forbrukerne sitter mitt i veikrysset der big data, apper, utbygging av det smarte nettet og og nye, innovative elektiske apparater og maskiner med dertil hørende programvare og den viktige rolla forbrukerne kommer til å ha.

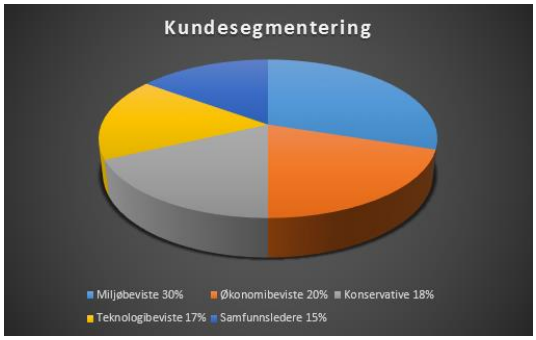
Undersøkelsen viser at forbrukerne tar vel imot fordelene med de nye smarte energiteknologiene, og mange er interessert i nye tariffer og elektronisk kundeservice som blir tilgjengelige. Undersøkelsen testet tre store fordeler med smart energiteknologi:

- Forbedret pålitelighet,
- Redusert klimagassutslipp
- Hjelpe forbrukerne med å spare penger ved å muliggjøre bedre energistyring.

I hvert tilfelle sier minst 86 prosent av forbrukerne at de tre nevnte momentene er viktige. Resultatene viser også en høy interesse for effektbasserte tariffer. For eksempel uttalte 3 av 5 respondenter at de ville sannsynligvis ville ønske å prøve effektariffer. Der får kunden en fast månedlig pris, men blir straffet om den bruker for mye effekt og vil få rabatt om den bruker lite effekt. I tillegg sa 25-50 prosent av forbrukerne i de ulike gruppene at de kunne ønske å prøve andre modeller for prissetting også. Studien viser at forbrukerne favoriserer smarte energiprogrammer og tariffer som tilbyr enkelhet og bekvemmelighet, og som lar kunden styre forbruket framfor å bli styrt av kraftselskapet.

Kundene ble delt opp slik:

1. Miljøbeviste
2. Økonomibeviste
3. Konservative
4. Teknologibeviste
5. Samfunnsledere og «eliten»



Figur 4 kundesegmentering

Man kan spørre hvorfor kundesegmentering er viktig innen energibransjen. Tross alt antar de fleste kraftselskapene at de fleste forbrukere er like når det gjelder energiforbruk. [18]Kundene tar kun kontakt når strømmen er ute og for spørsmål vedrørende strømregningen. Kunder har variert bakgrunn og forbruker energi forskjellig. Som et resultat virker ikke en tilnærming der man behandler alle kunder likt. SGCCs forsking viser at forbrukerne har ulike ønsker og behov. Segmentering gir større muligheter for å engasjere forbrukerne bedre og gir innsikt i hva elforbrukerne ønsker mest av smarte tariffer og smart teknologi. Vi tenker at en ligende undersøkelse kan hjelpe myndigheter og bransje her hjemme til vår utrulling av smarte nett.

7.4.7 SMARTE ANSATTE

Smarte nett trenger også smartere ansatte.

For å forberede arbeidstakere på AMS-utrullingen effektivt, må opplæring og videreutdanning gjelde for bestemte oppgaver og være relevant for den enkeltes stilling. Det motsatte er å gi ufokusert, generell informasjon som vanligvis går tapt etter at en person forlater et seminar eller en workshop. Kraftselskapene må identifisere hvilke stillinger som er mest berørt foran utrullingen av smarte nett og organisere opplæring og yte støtte rundt de ansatte og deres verdifulle bidrag nå når jobbsituasjonen blir endret. Nye og endrede stillingsbeskrivelser for den ansatte gjør at vi må belyse hva arbeiderne skal gjøre mer, bedre eller annerledes, og hvilke systemer som må på plass for å støtte den ansatte i denne overgangen.

7.4.8 SMARTE BYER

Eksperimentelle pilotprosjekter for spesifikke smarte teknologiinitiativer

Et eksempel på utvikling i byer finne vi i San Diego og Jacksonville i Florida. [17] De har samarbeidet med GEs Business Innovations Unit for lokale smarte gatelyspiloter, et program som fokuserer på å vise hvordan smart belysning vil være gunstig for energibransjen og den overordnede infrastrukturen for fremtidige "smarte byer". Begge byene har utplassert 4.000 smarte LED-gatebelysninger utstyrt med overvåkingskameraer for å finne parkeringsplasser. De smarte LED-gatelysene gir også bedre energieffektivitet, og rapporter viser at hver by har spart ca $ 350 000 årlig på strøm. Teknologien som er utprøvd vil bli værende og det jobbes med å lansere dette for andre kraftselskap og andre byer.

7.4.9 SMART KOMMUNIKASJON

Dagens avansere teknologi med mobiilt bredbåd og raske prosessorer gjør det mulig for kraftselskapene å nå kundene gjennom toveis kommunikasjon. AMS-teknologien er imidlertid ikke optimalisert for utsendelse av informasjon. Kraftselsakp som søker en støøre informasjonsutveksling, har begynt å implementere avanserte prisresponsive og lastkontrollprogrammer ved hjelp av eksisterende bredbånds- og mobilnett. Bredbånd gjør det mulig å implementere mer robust toveiskommunikasjon enn å kun bruke AMS-teknologien i målerene NVE krever.

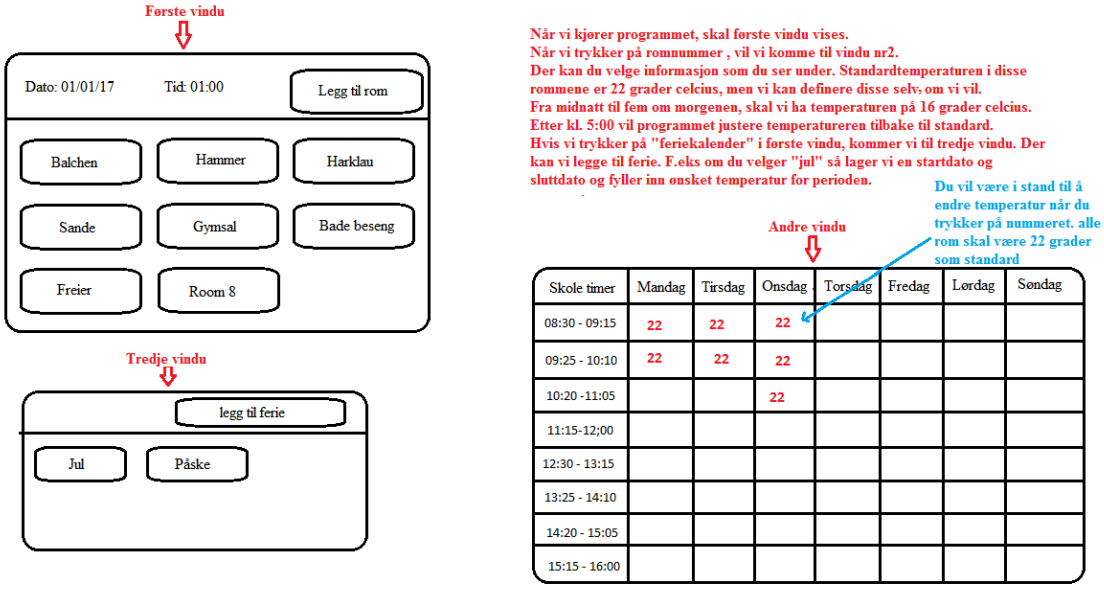
8.0 Programmering

8.1 Oppgave og bakgrunn.

Oppgava som vi fikk fra SFE var å lage en programert styring for en skole. I programmeringa skal vi lage et program i Java hvor vi styrer ulike rom etter ulike temperaturer. Dette gjør vi basert på hvilket bygg det er, til dømes vil vi gjerne ha det litt varmare i et svømmebasseng. Romma i skolene kan være: svømmebasseng, ulike klasserom, gymsal, gang, toalett, kantine osv. I selve programmet har vi tenkt å bruke to ulike GUI vinduer. Idet vi trykker på det ene vindauget, kommer vi inn til det andre vinduet. I vindu nummer 1 har vi oversikt over ulike rom, og da vi trykker på dette, kommer vi til vindu nummer 2. I dette vinduet kommer vi til å skrive informasjon som timeplan, dato og klokkeslett over når rommene er i bruk. I timeplanen har vi gjerne oversikt over hvilke timer vi har, og hvor mye temperaturen skal reguleres, og en modus for både natt- og dagmodus. Dette var vår tankegang før vi starta med det hele under forprsjektet.

8.2 Planlegging.

Etter at vi hadde tenkt en periode, så prøvde vi å lage en slags tankefigur med tekst på hvordan vi hadde tenkt å få oppgåva til å se ut. Etter å ha brukt noen timer på tegningen, endte vi med de tankene som ses på figuren under.



Figur 5 Forklaring på programstruktur

Det endte med å komme med tre ulike vinduer (ved hjelp av java GUI) for vi syntes det virket mer oversiktleg og enklere. Idet man kjører programmet, så tenkte vi å få opp et vindu med klokkeslett og dato. Her får vi da oversikt over alle rom på skolen. Vi valgte å gi rommene navn etter klasserommene i HISF, men dette kan selvfølgeleg endres enkelt i programmet om man ønsker. Trykker vi på et av rommene, så skal vi komme til et vindu nummer to. Her har vi da oversikt over alle dager i uka, og undervisningstimer. Vi setter en standard romtemperatur på 22 grader, og når man kommer i dette rommet, kan man regulere denne temperaturen om man ønsker. Vi bruker og skoletimene som vi har ved HISF som et eksempel i programmet vårt, men dette kan og endres i programmet.

Fra midnatt og til 5 om morgenen, tenker vi å ha ein automatisk endring til 16 grader celcius. Dette er da nattmodusen i programmet. Vi satte denne funksjonen fordi at systemet vi skal lage til skulen, og skal vere bærekraftig, og dermed regulerer vi ned for å synke ned kostnadene. Etter fem varmer vi opp skulen, for å gi lærerane, elever og andre ansatte et trivelig og normalt temperatursnivå til læring.

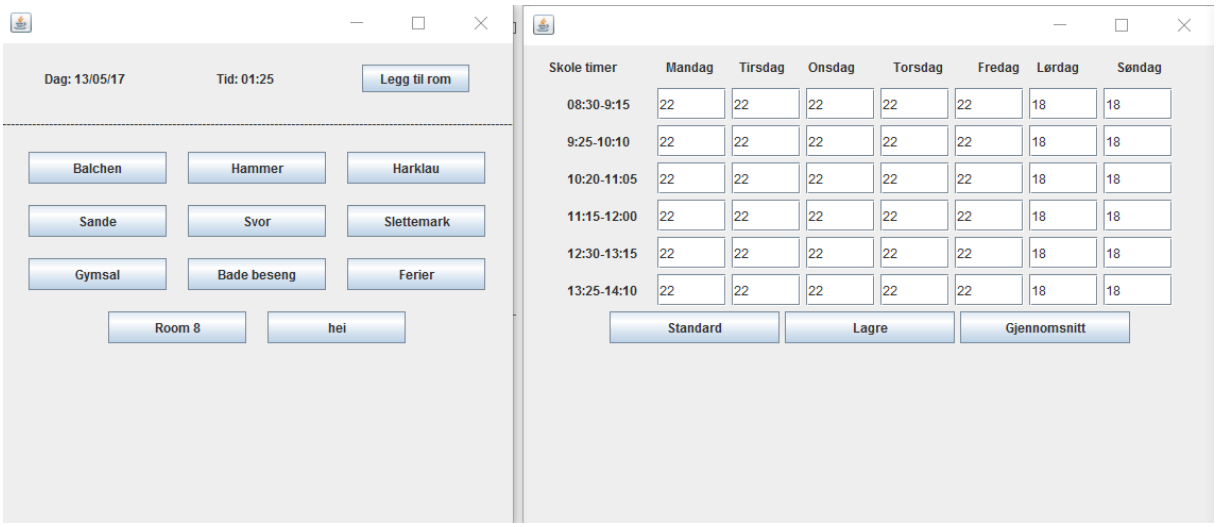
Vi tenker og å lage en feriefunksjon, med tanke på at skolene ikke blir brukt så mye når det er ferie. Dette gjør vi da i første vindu, og legger en rom som vi kaller for ferie. Her tenker vi at idet man trykker på rommet, så velges ferien, som f.eks «juleferie», og setter en startdato og en sluttdato på ferien.

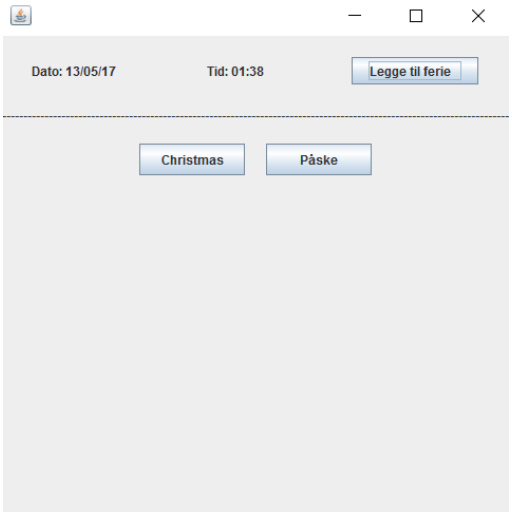
8.3 Teorisamling og samling av naudsynt informasjon.

Ved hjelp av tidlegere undervisning i Programmering 1 og 2, så hadde vi noe bakgrunnsinfo innen dette emnet. Likevel måtte vi samle mer informasjon for å få til dette. Dette gjorde vi ved å lese i kilder, mye på internett ved bruk av kilder og ved å se på nyttige videoer på YouTube. [18]

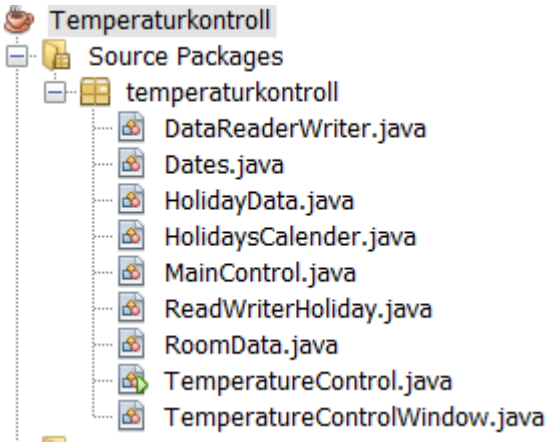
Etter at vi har samlet all informasjon som vi trengte i prosjektet, så var vi klar til å begynne med oppgaven. Samlingen av teorien tok noen få veker, men for å utføre det praktiske, trengte vi den teortiske kunnskapen bak emnet.

8.4 Sluttprodukt.





Figur 6 Ferdig Javaprogram.



Figur 7 Filinformasjon av program.

I figur 7 så ser man et lite utklipp for de ulike javafilene i vårt gui program. I temperaturControlWindow har vi laget de ulike label funksjonene som de kan se på figur 4. Disse er de ulike dagane, tidspunkter, standardfunksjon på 22 grader celcius og en gjennomsnittsknapp. Gjennomsnittsknappen regner da ut snittet av alle disse verdiene i løpet av de ulike skoletimene.

I RoomData oppretter vi de ulike rommene og skriver koden til å opprette nye rom. Dette utføres ved å trykke «Legg til rom» på vindu 1.

I MainControl oppretter vi de ulike egendefinerte romma som vi laget i Java. Disse er f.eks «Balchen» som kan sees i Figur 6. Vi bestemte oss for å opprette navn etter de ulike rommene på Høyskolen for å ha et eksempel å gå etter.

Den største utfordringen vår var å lage ferierommet. Dette klarte vi til slutt, ved å opprette HolidaysData og HolidaysCalender. Etter litt undersøkelse, fikk vi det til slutt, og vi klarte å

lage en «feriefunksjon». Ved å trykke på «Ferier», så kommer vi til et tredje vindu og der satte vi funksjoner for start-og sluttdato. Vi klarte således å lage ferdig programmet helt som vi tenkte på starten. Programmkoder kan finnes under vedlegg 1.

9.0 Drøfting

Vi har i oppgaven vår laget et eget energidisplay. Smarte nett vil bli fremtidens kraftnett, og dens hensikt er å utnytte styre og lagre energien på en bedre måte. Smarte nett omfatter ganske mye. I vår case fra SFE har vi valgt å fokusere på å lage en energidisplay og software som vi gjorde i java for å styre strømbruken i en skole.

Vi har også kontanktet Ulf Møller fra Energi Norge. Han har da laget en veileder for kraftbransjen. Noen av anbefalingene hans, er at vi burde sette i gang prosjekt innen smartteknologi, før myndighetene kommer med forskrifter og reguleringer. Dette har med tidsbruk å gjøre, siden det ofte tar lang tid hvis det ikke settes i gang. I følgje Ulf Møller vil et demonstrasjonsporgram strekke seg over flere åro og dersom man utsetter realisering av resultater og geninster til hele programmet er avsluttet, vil det både føre til reduksjon av økonomiske gevinster og kontinuerlig forbedringsarbeid ikke får slått rot i orgainsasjonen.

Grunnen til at vi tar opp dette er at det viser at vårt lille prosjekt både bør og realiseres tidlig selv om SFEs Smart Valley prosjekt nødvendigvis ikke er ferdig enda. Selv om vi har fokusert på en enkel skole, vil vi at NVE kommer med tiltak som gjør at kommuner, fylkeskommuner og stat blir tvunget til å lansere lignende piloter.

Dette vil i sin tur føre til at det blir et større marked for software selskaper, og tilbydere av elektriske apperater og maskiner som knyttes til det smarte nettet.

10.0 Konklusjon

Ved å se tilbake på hovedmålet, så ser vi at vi har nådd det. Vi har laget et energidisplay program i java, hvor vi styrer strømbruken i en skole. Vi har også sett på utfordringene og mulighetene AMS utrullingen gir.

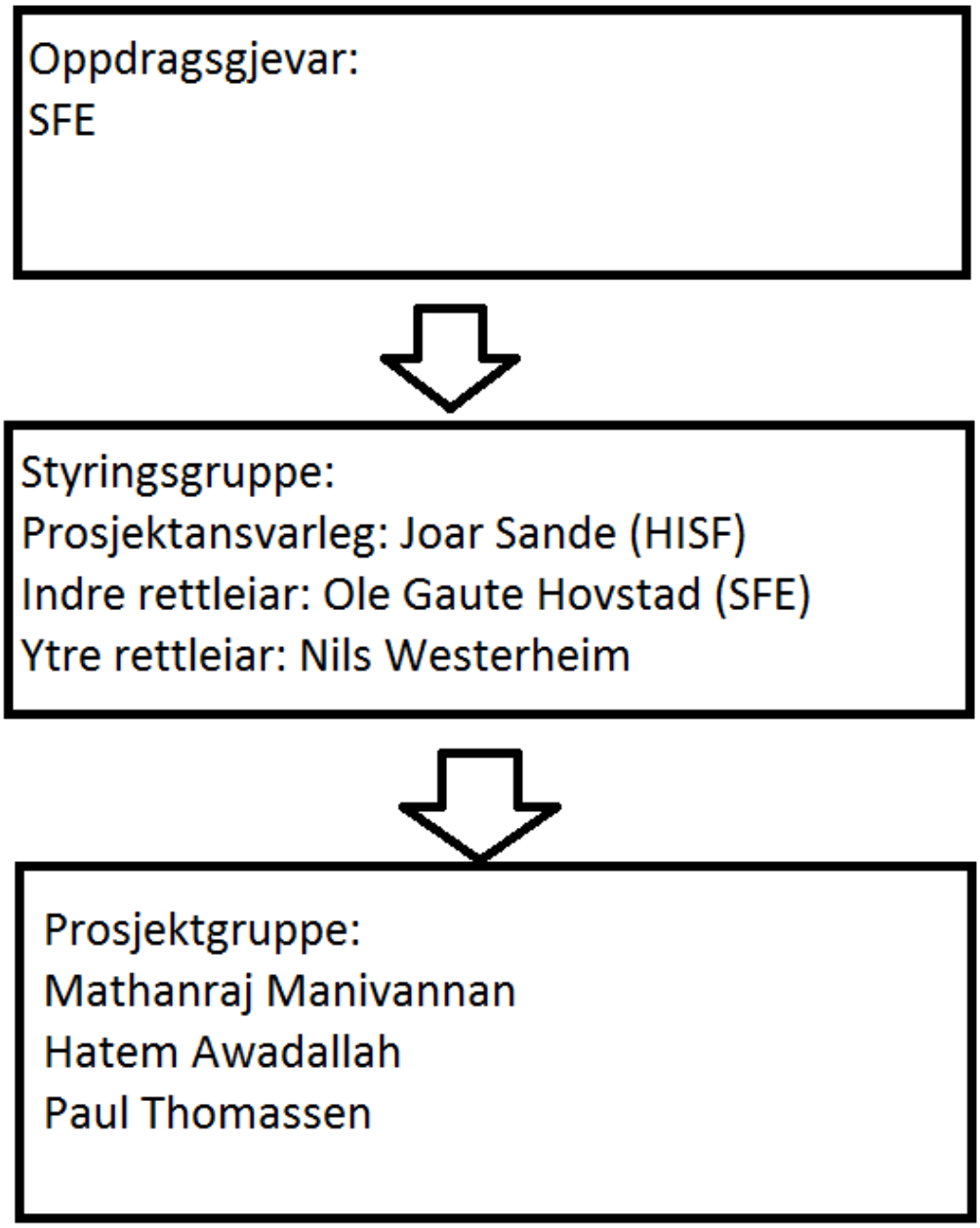
Programmet, som vi har laget, er brukervennlig og enkelt å bruke. Ved å starte programmet, så kan man enkelt endre temperaturer, legge til rom, sette en feriefunksjon og regne gjennomsnitt.

Vi ser at samfunnet står overfor store utfordringer med utviklingen av den nye infrastrukturen smarte nett krever. Politikerne må fatte vedtak som NVE følger opp. Kraftbransjen må fokusere på utdanning av nytt personell, og de sammen med private elektrobransjer må se på de nye markedene som åpner seg når alle elektriske apperater for en IP-adresse.

11.0 Administrasjon av prosjekt

11.1 Organisering

Figur 8 viser organisasjonsturkutruen i prosjektet. Prosjektet vårt er drivet av tre ulike nivåer, oppdragsgiver, styringsgruppe og en prosjektgruppe.



Figur 8 Prosjektstruktur

11.1.1 Oppdragsgiver.

Etter litt spøring, fikk vi vår oppgave fra SFE. SFE,

Sogn og fjordane Energi, er en vasskraftprodusent, og levandør av kraft.

Hovedkontoret deres ligger i Sandane. Man kan lese mer om SFE på deres egne nettsider, http://www.sfe.no/.

Figur 9 logo sfe

11.1.2 Styringsgruppen.

Styringsgruppen er de som har det overordna ansvaret i prosjektet vårt, og virker som ledere. Under arbeidet vårt, så ble vi veiledet av de, og styringsgruppen bestemmer viktige avgjørsler under prosjektet.

Styringsgruppen er satt sammen av:
Indre rettleder: Ole Gaute Hovstad, kontaktpersonen vår hos SFE.
Ytre rettleder: Nils Westerheim, prosjektansvarlig.
Prosjektleiar: Hatem Awadallah, representerer prosjektgruppen.

Kontaktinformasjon rettledere:

Navn:	E-post:	Nummer:
Nils Westerheim	nils.westerheim@hisf.no	95281089
Ole Gaute Hovstad	ole.gaute.hovstad@sfe.no	916 40 231

11.1.3 Prosjektgruppen.

Prosjektgruppen har ansvaret for å utføre prosjektet. Vi skal dermed gjøre både det teoretiske og det praktiske arbeidet bak dette. Denne gruppen består av tre personer, to av de tilknyttet til automasjonsfaget, og den eine innen elkraftfaget. Ingen av oss har noen bakgrunn relatert til prosjektet ut i frå tidlegere studier og erfaringer, så dette ville bli en utfordring.

Kontaktinformasjon prosjektgruppe:

Navn:	E-post:	Telefon:
Paul Thomassen	thomassen7@gmail.com	46988460
Mathanraj Manivannan	z94mathan@live.com	46851237
Hatem Awadallah	hatemnor@hotmail.com	90751669

Møtene og fordelingene fungerte som vi ønsket. Det har vore jevnlig kontakt mellom oss tre, men Paul har og hatt jobb i perioden, so det var litt vanskelig å holde møter. Likevel klarte vi å holde oss på rett spor, og vi klarte å opprettholde et godt sammerbeid. Problemer, som vi har møtt underveis, har vi diskutert oss i mellom, og når vi satte fast har vi spurt våre rettledere.

11.2 Ansvarfordeling.

I prosjektet har vi delt oppgaven i ulike delar. Dette er fordi at vi har ulike linjer vi har studert ved Høgskulen i Førde, og derfor syntes vi dette var best for å bruke kunnskapen vår så godt som mulig. Vi lot likevel oss utfordre selv til å studere begge studieretningane, både elkraft og automasjonstudiet, slik at alle får kjennskap i begge fagområdene og elementene i prosjektet. Hovedansvaret for de ulike emnene ble likevel fordelt til oss tre i mellom, for å ha kontroll på emnene i prosjektet. Dette henger sammen med kunnskapsområdene våre og erfaringer.

Hatem Awadallah har vært prosjektleder, det er han som foreslår møter og det administrative knytt til prosjektet. Men vi fikset fremdriften i forhold til vår plan, ressursplan og

arbeidsfordeling oss i mellom. Hatem har og hovedansvaret for programmeringsbiten av prosjektet.

Mathanraj Manivannan har hatt ansvar for rapportskrivingen, nettsted og programmeringsbiten. Han har ansvar for å skrive rapporten, men selvsagt hjelp fra de to andre i gruppen. Her ligger også ansvar for de ulike dokumentene knytt til prosjektet. Dessuten jobber både Hatem og Mathanraj med det teoretiske bak programmeringdelen, likeså utførelsen.

Paul Thomassen har også hatt ansvar for det teoretiske grunnlaget i prosjektet. Siden erfaringen hans ligger innen Elkraft, så har han sett nærmare på AMS biten av prosjektet. Det vil og seie at teorien innebærer å vurdere, drøfte, presentere teori som er relevant for prosjektet. I tillegg har han vært med å skrive rapporten.

Selv om vi la hovedansvaret på rapporten til em person, var dette jobbet sammenn i gruppa. Dette gjelder også andre dokument som vi har levert innen perioden, og det var et felles ansvar. Alle personer i gruppen har et ansvar å dokumentere hva en har jobbet med i perioden, og å hente informasjon. Vi samlet alt dette i et skjema som de kan se under vedlegg 2. Informasjonen, som man synes er viktig, blir diskutert innen gruppa under møter, slik at alle for et innsyn i emnene og forståelse. Ansvarfordelingene, som er nevnt ovenfor, er og hvordan vi arbeidet. Men til syvende og sist, så har vi hatt et delt ansvar innen hele prosjektet, og derfor var det avhengig av at alle delene blei samlet til ett. Dette gjorde også at vi lærte mye fra hverandre, og et bedre sammensatt rapport.

11.3 Arbeidsmetode.

Prosjektet vår var samansatt, så vi trengte å øke vår kunnskap, både teoretisk og praktisk. Dette gjorde vi gjerne ved å søke innen internett, bøker og lærere. Det meste lærte vi gjerne mest ved å søke i Google, og å finne relevante artikler i prosjektet vårt. Dessuten har tidlegere prosjekter i bibsys vært til stor hjelp. Emner innen AMS var vanskelig å hente fra SFE, så vi har og kontaktet flere levandører for å få mer informasjon om emnet. Ved å få flere

synspunkt, førte det til at vi lærte mer om emnet. Vi har holdt oss mest til primære kilder, men ser at det var nødvendig å bruke sekundære kilder for å få mer informasjon.

Mye av programmeringsbiten lærte vi ved å spørre faglærer ved hjelp når vi sat fast, ellers klarte vi å løser det meste ved å studere selv. I det praktiske arbeidet har vi da hatt fokus på å utvikle programmet som SFE ba oss om å lage. Det har vore fokus på å gjøre dette riktig fra bunnen av, så vi endte med å ikke ha mye feilsøking på slutten. Dette var ganske hjelpsom, siden dette kunne ført til stress ved slutten av prosjektet. Dermed fikk vi små tekniske problem når det gjelder programmeringsbiten.

De beslutningene og valgene vi tar, har vi både diskutert med oss selv i gruppen, og diskutert med våre rettledere. Dette kommer til hjelp ved at vi ser på erfaringane våre, og til å drøfte val av løsninger.

Til å bygge rapporten, har vi benyttet av oss av kilder som «Rapportskriving – Råd og vink » [19] og vi har sett på tidlegere rapporter «AUTOMATISERT MODELL AV VASSBEHANDLINGSANLEGG». [20] Ved å følge disse rådene, ble det mye enklere å holde flyt og en rød tråd gjennom hele teksten. Det ble enklere for oss å skrive, følgje ein strukur og gi informasjon som gjorde dette mye enklare.

11.4 Møteplan.

I utgangspunktet, hadde vi planlagt å ha møter hver fjortende dag. Etter en periode, merket vi at dette ikke var nødvending, siden vi egentlig utvekslet informasjon via sosiale medier, og så at alle gjorde slik man skulle gjøre. Ved at vi var effektive, så fant vi ut av det var ikke behov for å ha møter så hyppig som vi i utgangspunktet trodde, og dermed hendte det ofte at det var møter hver tredje uke, der vi diskuterte hvor langt vi har kommet, og hva vi skulle gjøre videre. Det var dannet møter, når en av oss i gruppen følte behov for det, eller når en i styringsgruppen ba om det, for å ha en oppdatering. Møter kan det finne under vedlegg 3, 4 og 5.

Uformelle møter i gruppa vart gjerne hyppige, det var da vi møtes på skolen og jobbet med teorisamling, oppgaveløsning og diskusjon. Dette var gjort noen ganger hver uke, og var de «møtene» vi fikk mest utslag på. Noen få av disse møtene, gjerne de av det viktigare slaget, kan lesast meir på nettstaden vår.

Vi tenkte å sende oppdateringer til våre rettledere og oppdragsgiver via nettstaden vår. Ellers ble også mye informasjon delt via e-poster. I nettsiden ligger all nødvendig informasjon kring prosjektet vårt, hvordan vi lagde programmet, planleggingen bak det og informasjon om møter. Dessuten kan De finne forprosjektet og slikt der og.

11.5 Gantskjema

I starten av prosjektet så lagde vi et gantskjema som har einoversikt på hvordan vi hadde tenkt å jobbe i prosjektet. Det var greit å ha en slags plan på hva vi skulle gjøre utover semesteret, og gantskjemaet, som vi laget tidleg, hjalp oss med å holde orden på saker. Vi brukte excel og lagde gantskjema. Gantskjemaet kan finnast under vedlegg 6. Vi satte og noen minipæler, og disse er gjerne i form av tidsfrister som måtte nås. Disse er satt av høgskolen, og det er mål som vi måtte nå i prosjektperioden.

Da vi sammanlignet med endelig gantskjema, så ser vi at det er noen få forskjeller, og at det ikke gikk som vi hadde planlagd. Dette kommer av kollisjon av andre fag, jobb, sykdom og andre årsaker. Likevel fikk vi gjort alt innen tidsfristene, og det gjekk således ganske greit. Vi ble likevel ferdig med prosjektet tidlig som vi trodde, men det skriftlige arbeidet tok lengre tid enn planlagd.

11.6 Dokumentstyring

Til å skrive dokumenter, har vi brukt google docs. Dette fant vi ut var effektiv, siden det hjalp oss med å holde styr på å skrive ting samtidig, og å redigere filene i en og samme fil. Google docs er en tekstbehandlingprogram som en kan endre på nettet [21]. Når det gjelder å ha styr på dokumentene våre, så hadde vi så og si ingen problem med det, og holdt dette systematisk og under kontroll.

11.7 Tidsbruk

Prosjektet gir oss 20 studiepoeng, og det er beregnet at hver student skal ha 500 timer til rådighet. Det gikk mest tid på å samle informasjon til å løse oppgavene som vi fikk utdelt og å skrive rapport fordi en som regel aldri vet helt hvor lang tid det tar. De 500 timane omfatter

helt fra begynnelsen av prosjektet, samling av informasjon, utføre oppgaven, rapportskriving og til sluttprodukt. Vi lagde en nøye presisjon på hvor mange timar vi brukte i eit exceldokument. Her innførte vi antall timer, etter vi ble ferdig med jobbingen for en dag. Etterpå samlet vi disse i et enkel dokument for seg selv, med beregnet og hypotetisk tidsbruk hver for oss. Etterhvert, som arbeidet har utviklet seg, så ser vi at vi brukte mer timer enn vi hadde planlagd. Totalt brukte vi circa 350 timar på prosjektet, og dette kan leses nærmare på vedlegg 7 under. Dette ble mye meir enn det vi trodde på forhånd, der vi antok totalt timebruk på mellom 220-235 timer.

11.8 Nettside

Til nettside har vi benyttet oss av wix.com. Dette er en gratis tjeneste, og det var enkelt å lage. Her har vi lagt både forprosjekt, gantskjema, tidsplan, presentasjoner og alt som er relatert med prosjektet. Dessuten har vi laget ein liten logg med innlegg på dee viktigeste møtene våre, og ei liten oppdatering på fremdriften i prosjektet. Vi valgte denne netttjenesten, fordi den var enkelt å bruke, og nettsiden var enkelt å lage. Dette gjorde også at vi ikke stresset med resten av prosjektet, siden det var lett å bruke.

Det var lett å bruke, gratis, og hadde gode brukarveiledninger. Oversiktlige funksjonar, gjorde også redigeringen gøy og enkelt. Nettsiden vår kan ses her:
https://z94mathan.wixsite.com/smartvalley

11.9 Risikostyring.

I forprosjektet vårt så ble det gjennomført en risikovurdering av prosjektet. Dette skrev vi i et eget exceldokument for seg selv. Det kan leses under vedlegg 8. Der fokuserte vi veldig mye på faresignal som kunne oppstå når vi jobbet med prosjektet. Dette gjaldt i hovedsak det prosjektfaglige arbeidet. I en risikoanaylse, so kartlegg en over mulige hendelser som kan inntreffe i prosjektet [22]. Vi har valgt å skrive ulike tiltak, og kalkulerte risikoane etter konsekvens og sannsyn. Desse er da rangert frå 1-5 og går i denne skalaen: «låg», «svært låg», «middels», «ganske stor» og «stor».

Risikofaktor = Konsekvens * Sannsyn

Vi tenkte å bruke en risikofaktor på 9. Dette er fordi om vi har middels sannsyn (både sannsyn og konsekvens) så blir det 9, og dessuten kom vi frem til i gruppa at 9 virka best med hensyn på risikofaktorene vi fikk utregnet. Riskofaktorer som vi fekk under 9, ble ikke skrevet noen form for tiltak imot, men det betyr ikke at vi ignorerte det i videre arbeid med prosjektet.

Når vi likevel arbeidet med prosjektet, så møtte vi ikke noen form for problem med hensyn til risikovurderningene som vi presenterte. I mange tilfeller har det vore vanskelig å møte kvarandre grunna jobb, sykdommer og andre fag. Disse enkelthendelsene som forseintkomingar, har vi ikkje hatt problem med, så dette var ikke noen problem. Oppsto det problem i gruppen, ble det diskutert innen gruppen, og vi fikk et gått gruppemiljø.

11.10 Budsjett.

Før vi startet prosjektet, var det ikke nødvendig å utarbeide noen form for budsjettplan. Modellen vi lager er basert på et program som vi utvikler i java, og dette er gratis. Nettsiden, vi oppretta, var gratis så her ble det heller ikke brukt noen kroner. Teorisamlingen var også gratis, i høve til alt all informasjon vart samlet fra internett, forelesninger, bedrifter og lærere. Plakatkostnadene er dekket av skolen, så det samlet beløpet og det beløpet vi antok var 0 kr.

11.11 Prosjektevaluering.

11.11.1 Gruppe og kommunikasjon

Gruppa vår jobbet med det prosjektadministrativet i forrige semester, så vi var trygge på hverandre i starten. Siden vi har jobbet sammen før, var det enklere for oss å jobbe sammen i gruppe igjen. Gruppen vår mener at dette har fungert bra, og vi har fått til det vi hadde tenkt til fra starten. Enkelte uenigheter i gruppa har vi diskutert saman, og komme frem til løsninger ved å dele ulike synspunkter.

Totalt sett synes vi at det at vi fungerer saman som ei gruppe og at vi klarer å drøfte sammen til en løsning, har hjulpet oss til å komme frem til løsningene våres i prosjektet.

Vi har opplevd ulike utviklinger, og alle har fått ulike erfaringer og mye kunnskap innen prosjektarbeid og prosjektstyring.

11.11.2 Utfordringer

Fagområene vi måtte sette oss inn i var veldig omfattende og det var behov for mye energi for å sette seg inni det. Vi fikk mange utfordringer, og dette gjaldt å sette seg inn i grunnleggende teoretiske emner. Dette gjelder både å sette seg inn og studere om smartteknologi, AMS og å få all kunnskapen som var behov for å programmere. Men siden programmeringsspråket er java, så hadde vi mye kunnskap fra før, så det var noe bakgrunnsstoff.

I alle prosjekt vil det alltid bli utfordringer, og disse fikset vi i løpet av proseesssen. Det å ha utfordringer, og å finne ut hvordan de løses, eren læringsprosess som er ganske nyttig. Gruppen vår mener da at dette ble veldig viktig for prosjektet.

11.11.3 Hva har vi lært?

Vi har lært mye både indvuiduelt og som gruppe å jobbe med dette prosjektet. Vi setter oss inn i fagområder, og emner som vi hadde noe kunnskap i, men ikke mye. Dette gjorde at vi lærte ganske mye, fikk utfordringer, og gjorde oppgaven ekstremt spennende. Fag som vi har hatt tidlegere som Programmering, elektro, Ingeniørfaglig systememne, Ingeniørfaglig innføringsemne høgspennsanlegg og fordypingsfag i elkraft har blitt nyttet her, og ved å kombinere disse fagfeltene sammen, så føler vi har fått mye utbytte. Dette gjelder elkraftstudiet og automasjonstudiet ved Førde.

Med å jobbe med hovedprosjektet, har vi også lært hvordan man utfører et større prosjekt. Vi har riktignok hatt mindre prosjekt i studiet, så det var noe erfaring innen emnet. Det å jobbe med rettledere og partnere, har vært helt annerledes. Gruppa har da blitt flinkere med det prosjektfaglige, og vi spurte mye hjelp fra dem via epost.

Til syvende og sit er gruppa ganske fornøyd med det endelige produktet, modellen og rarrporten som er utarbeidd.

12.0 Referanseliste:

1.Reknes, A, H. Gytri, A. Ola. Sambygdingane blir prøvekaninar for framtidas energiforbruk[Internett] [hentet 16.05.17]. Tilgjengelig fra: https://www.nrk.no/sognogfjordane/vesle-hyen-blir-testbygd-for-framtidas-energiforbruk-1.12570974

2. Smartgrid, The Norwegian Smartgrid Centre, [henta 16.03.17] Tilgjengelig frå: http://smartgrids.no/senteret/about-smartgrid/

3. Nettnytte, nye muligheter for kraftnettet ved innføring av AMS, lese 16.03.17, henta frå: http://hovedprosjekter.hig.no/v2014/tol/elektro/ams/AMS/FORSIDE.html

4. Lie, Ø Slik vil nettselskapene holde nettleia oppe i fremtiden [Internett] sist oppdatert 6.12.13 [hentet 17.03.17] Tilgjengeleg frå: http://www.tu.no/artikler/slik-vil-nettselskapene-holde-nettleia-oppe-i-fremtiden/233406

5. Rouse M, Mckenzie C. Definition Java [Internett]. [Henta 06.05.17]. Tilgjengelig fra: http://searchmicroservices.techtarget.com/definition/Java

6. Rouse M. Object-oriented programming (OOP) Java [Internett]. [Henta 06.05.17]. Tilgjengelig fra: http://searchmicroservices.techtarget.com/definition/object-oriented-programming-OOP

7. Rouse M. Java virtual machine(JVM) [Internett]. [Henta 06.05.17]. Tilgjengelig fra: http://www.theserverside.com/definition/Java-virtual-machine-JVM

8. Java Programmering Tutorial Programming Graphical User Interface (GUI) [Internett] [Henta fra 07.05.17]. Nanyang Technological University Tilgjengelig fra: https://www3.ntu.edu.sg/home/ehchua/programming/java/j4a_gui.html

9. Swing [Internett] [Henta 09.05.17] Norges teknisk-naturvitenskapelige universitet Tilgjengelig fra: https://www.ntnu.no/wiki/display/tdt4100/Swing

10. Look and feel [11.05.17] Wikipedia, the free encyclopedia Tilgjengelig fra: https://en.wikipedia.org/wiki/Look_and_feel

11 Smart Grids – nøkkelen til et fleksibelt energisystem [internett] [Henta 27.04.17] tilgjengelig fra: https://www.sintef.no/projectweb/smartgrids/smart-grids--nokkelen-til-et-fleksibelt-energisystem/

12 Smarte nett og husholdninger [internett] [henta 28.04.17] tilgjengelig fra: http://ungenergi.no/miljoteknologi/bygg/smarte-nett-og-husholdninger/#point0

13 Ny teknologi og forbrukerfleksibilitet [Internett] NVE [Henta 14.04.17] sist oppdatert 04.05.17 tilgjengelig fra: https://www.nve.no/elmarkedstilsynet-marked-og-monopol/sluttbrukermarkedet/ny-teknologi-og-forbrukerfleksibilitet/

14. Veileder for design og utvikling av Smart Nett-prosjekter i distribusjonsnettet [Internett] [Henta 15.04.17] Energi Norge, Tilgjengelig fra: https://www.energinorge.no/contentassets/89751508098c44fb95fafcac64075a37/veileder-for-smarte-nett.pdf

15. Lie, Ø Da kundene måtte betale for effekt i stedet for forbruk, gikk strømforbruket ned med 20 prosent [Internett] sist oppdatert 3.7.14 [henta 04.04.17] tilgjengelig fra: https://www.tu.no/artikler/da-kundene-matte-betale-for-effekt-i-stedet-for-forbruk-gikk-stromforbruket-ned-med-20-prosent/223269

16 SGCC's Survey Reveals Awareness and Attitudes Toward Smart Grid-Enabled Programs and Technologies [Internett] Smartgrid CC Atlanta USA Hentet [06.04.17] tilgjengelig fra: http://3593f84chf852yw5d4c5emoe.wpengine.netdna-cdn.com/wp-content/uploads/2015/04/FINAL__Consumer-Pulse-Wave-5.pdf

17 San Diego to Deploy World's Largest Smart City IoT Platform with Current, powered by GE [Internett] Marketwired USA henta [02.05.17] tilgjengelig fra: http://www.marketwired.com/press-release/san-diego-deploy-worlds-largest-smart-city-iot-platform-with-current-powered-ge-2197840.htm

18. Java Swing Tutorial: Introduction and goals [Internett] YouTube Tilgjengelig fra: https://www.youtube.com/watch?v=I41_sdkuzOY&list=PLY-H7Nl6qqgntwySdPesG8eLJRpdjin19

19. Rapportskriving - Råd og vink [Internett] Norges teknisk-naturvitenskapelige universitet [Henta 28.04.17] Tilgjengelig fra: https://www.ntnu.no/kt/studier/rapport

20. Grønsdal T, Olsbø B, Rauset K Automastisert modell av vassbehandlingsanlegg [Bacheloroppgåve]. Førde: Høgskulen i Sogn og Fjordane: 2016, 87 s. Også tilgjengelig fra: https://brage.bibsys.no/xmlui/bitstream/handle/11250/2393075/Vassbehandlingsanlegg.pdf?sequence=1&isAllowed=y

21. How to use Google Docs [Internett] Google [Henta 11.02.17] Tilgjengelig frå: https://support.google.com/docs/answer/7068618?co=GENIE.Platform%3DDesktop&hl=en

22. Internkontroll – Informasjonsikkerhet [Internett] Difi [Henta 16.05.17] Tilgjengelig frå: http://internkontroll.infosikkerhet.difi.no/risikostyring/risikovurdering

13.0 Figurliste

Innholdsliste 14.0 Vedlegg

Programkode:

Temperaturkontroll

DataReadWriter.java:

```
package temperaturkontroll;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.io.FileWriter;

public class DataReaderWriter {

    ArrayList<RoomData> rD = new ArrayList<RoomData>();

    ArrayList<HolidayData> hD = new ArrayList<HolidayData>();

    ArrayList<RoomData> ReadData() throws FileNotFoundException,
IOException {

        String csvFile = "roomdata.csv";
        String line = "";
        String cvsSplitBy = ",";

        try (BufferedReader br = new BufferedReader(new
FileReader(csvFile))) {

            while ((line = br.readLine()) != null) {

                String[] cards = line.split(cvsSplitBy);

                int[] arr = new int[42];
                int c = 0;
                for (int i = 2; i < 44; i++) {
                    arr[c] = Integer.parseInt(cards[i]);

                    c++;
                }
                RoomData rd = new RoomData(Integer.parseInt(cards[1]),
cards[0], arr);
                rD.add(rd);
            }

        } catch (IOException e) {
            e.printStackTrace();
        }

        return rD;

    }
```

```
    void WriteData(ArrayList<RoomData> rt) throws Exception {

        FileWriter fileWriter = null;
        fileWriter = new FileWriter("roomdata.csv");

        try {
            for (int i = 0; i < rt.size(); i++) {
                int[] arr = rt.get(i).getArr();
                //Write a new student object list to the CSV file
                fileWriter.write(rt.get(i).getName());
                fileWriter.write(",");
                fileWriter.write(String.valueOf(rt.get(i).getRoomNum()));

                for (int j = 0; j < arr.length; j++) {

                    fileWriter.write(",");

                    fileWriter.write(String.valueOf(arr[j]));

                }

                fileWriter.write("\n");
            }
        } catch (Exception e) {
            System.out.println("Error in CsvFileWriter !!!");
            e.printStackTrace();
        } finally {

            try {
                fileWriter.flush();
                fileWriter.close();
            } catch (IOException e) {
                System.out.println("Error while flushing/closing fileWriter
!!!");
                e.printStackTrace();
            }

        }
    }

}
```

Dates.java:

```
package temperaturkontroll;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
```

```
import javax.swing.JPanel;
import javax.swing.JTextField;

/**
 *
 *
 */
public class Dates extends JFrame implements ActionListener {

    JPanel panel;

    public Dates(ArrayList<HolidayData> hD,HolidayData hdd) {

        super("");
        System.out.println(hdd.getTemp());

        setLayout(new BorderLayout());
        this.panel = new JPanel();
        this.panel.setLayout(new FlowLayout(1, 20, 20));
        add(panel, BorderLayout.CENTER);

        JLabel jl1 = new JLabel("                                                    Ferie
informasjon                                          ");

        this.panel.add(jl1);

        JLabel jl2 = new JLabel("----------------------------------------
--------------------------------------------------------------------------
-------------------");

        this.panel.add(jl2);

        JLabel jl3 = new JLabel("Startdato:");
        jl3.setPreferredSize(new Dimension(130, 30));


        this.panel.add(jl3);

        JTextField jt0 = new JTextField(hdd.getStartDate());
        jt0.setPreferredSize(new Dimension(200, 30));
        this.panel.add(jt0);

        JLabel jl4 = new JLabel("Sluttdato:");

        jl4.setPreferredSize(new Dimension(130, 30));


        this.panel.add(jl4);

        JTextField jt2 = new JTextField(hdd.getEndDate());
        jt2.setPreferredSize(new Dimension(200, 30));
        this.panel.add(jt2);



         JLabel jl6 = new JLabel("Temperatur");

        jl6.setPreferredSize(new Dimension(130, 30));


        this.panel.add(jl6);
```

```
        JTextField jt3 = new JTextField(String.valueOf(hdd.getTemp()));
        jt3.setPreferredSize(new Dimension(200, 30));
        this.panel.add(jt3);


        JLabel jl5 = new JLabel(" ");

        jl5.setPreferredSize(new Dimension(130, 30));


        this.panel.add(jl5);

        JButton b1 = new JButton("Lagre");
        b1.setPreferredSize(new Dimension(200, 30));
        b1.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {


                try {
                    HolidayData hddd=new
HolidayData(hdd.getHoliday(),jt0.getText(),jt2.getText(),Integer.parseInt(j
t3.getText()));

                    hD.add(hddd);

                    ReadWriterHoliday rwh=new ReadWriterHoliday();
                    rwh.WriteData(hD);
                } catch (Exception ex) {

Logger.getLogger(Dates.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
        });

        this.panel.add(b1);
        setSize(500, 500);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent evt) {

        String m = JOptionPane.showInputDialog("Ferienamn:");

        JButton b1 = new JButton(m);
        b1.setPreferredSize(new Dimension(90, 30));

        this.panel.add(b1);


        this.panel.revalidate();
        validate();
    }

}
```

HolidayData.java:

```
package temperaturkontroll;
```

```
public class HolidayData {

    String holiday;
    String StartDate;
    String EndDate;

    int temp;

    public HolidayData(String holiday, String StartDate, String EndDate,
int temp) {
        this.holiday = holiday;
        this.StartDate = StartDate;
        this.EndDate = EndDate;
        this.temp = temp;
    }

    public HolidayData(String n)
    {
        holiday=n;
        StartDate="";
        EndDate="";
    }
    public String getHoliday() {
        return holiday;
    }

    public void setHoliday(String holiday) {
        this.holiday = holiday;
    }

    public int getTemp() {
        return temp;
    }

    public void setTemp(int temp) {
        this.temp = temp;
    }


    public String getStartDate() {
        return StartDate;
    }

    public void setStartDate(String StartDate) {
        this.StartDate = StartDate;
    }

    public String getEndDate() {
        return EndDate;
    }

    public void setEndDate(String EndDate) {
        this.EndDate = EndDate;
    }


}
```

HolidaysCalender.java:

```
package temperaturkontroll;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

/**
 *
 *
 */
public class HolidaysCalender extends JFrame implements ActionListener {

    int count = 2;
    JPanel panel;
    DateFormat df = new SimpleDateFormat("dd/MM/yy");
    DateFormat df1 = new SimpleDateFormat("hh:mm");

    Date dateobj = new Date();

    String sdate;
    String edate;

    ArrayList<HolidayData> hD = new ArrayList<HolidayData>();

    int s;

    public HolidaysCalender() throws Exception {
        super("");
        setLayout(new BorderLayout());
        this.panel = new JPanel();
        this.panel.setLayout(new FlowLayout(1, 20, 20));
        add(panel, BorderLayout.CENTER);

        JButton button = new JButton("Legge til ferie   ");
        JLabel jl = new JLabel("Dato: " + df.format(dateobj) + "
");
        JLabel jl21 = new JLabel("Tid: " + df1.format(dateobj) + "
");

        ReadWriterHoliday rwh = new ReadWriterHoliday();
        hD = rwh.ReadData();
        //sadd(jl,BorderLayout);
        this.panel.add(jl);

        this.panel.add(jl21);
```

```
        this.panel.add(button);

        JLabel jl2 = new JLabel("----------------------------------------
------------------------------------------------------------------------
------------------");
        //sadd(jl,BorderLayout);
        this.panel.add(jl2);

        button.addActionListener(this);
        //setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JButton b1 = new JButton("Christmas");
        b1.setPreferredSize(new Dimension(100, 30));
        b1.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {
                HolidayData hddd = hD.get(0);
                Dates d = new Dates(hD, hddd);

                d.setVisible(true);
            }
        });

        this.panel.add(b1);
        b1 = new JButton("Påske");
        b1.setPreferredSize(new Dimension(100, 30));
        b1.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {
                HolidayData hddd = hD.get(1);
                Dates d = new Dates(hD, hddd);

            }
        });
        b1.setPreferredSize(new Dimension(100, 30));

        this.panel.add(b1);

        s = count;

        if (hD.size() > count) {

            while (s < hD.size()) {

                b1 = new JButton(hD.get(s).getHoliday());
                b1.setPreferredSize(new Dimension(130, 30));

                b1.addActionListener(new ActionListener() {

                    public void actionPerformed(ActionEvent e) {

                        try {

                            Dates d = new Dates(hD, hD.get(s - 1));
                            d.setVisible(true);

                        } catch (Exception ex) {

Logger.getLogger(MainControl.class.getName()).log(Level.SEVERE, null, ex);
                        }

                    }
```

```
                });

                this.panel.add(b1);

                s++;

            }
        }
        count = s - 1;

        setSize(500, 500);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent evt) {

        String m = JOptionPane.showInputDialog("Fresie navn:");
        // System.out.println(m);
        if ((m != null) && (m.length() > 0)) {

            count++;
//If you're here, the return value was null/empty.
//setLabel("Come on, finish the sentence!");
            JButton b1 = new JButton(m);
            b1.setPreferredSize(new Dimension(90, 30));
            HolidayData hdd = new HolidayData(m);

//hD.add(hdd);
            b1.addActionListener(new ActionListener() {

                public void actionPerformed(ActionEvent e) {
                    Dates d = new Dates(hD, hdd);
                    d.setVisible(true);
                }
            });
            this.panel.add(b1);

            //this.panel.add(new JButton("Room "+count));
            this.panel.revalidate();
            validate();
        }
    }
```

MainControl.java:

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import java.awt.FlowLayout;
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
```

```
public class MainControl extends JFrame implements ActionListener {

    int count = 7;
    JPanel panel;
    DateFormat df = new SimpleDateFormat("dd/MM/yy");
    DateFormat df1 = new SimpleDateFormat("hh:mm");

    Date dateobj = new Date();
    int s;

    ArrayList<RoomData> rD = new ArrayList<RoomData>();

    public MainControl() throws Exception {
        super("");
        setLayout(new BorderLayout());
        this.panel = new JPanel();
        this.panel.setLayout(new FlowLayout(1, 20, 20));
        add(panel, BorderLayout.CENTER);

        JButton button = new JButton("Legg til rom");

        JLabel jl = new JLabel("Dag: " + df.format(dateobj) + "
");
        JLabel jl21 = new JLabel("Tid: " + df1.format(dateobj) + "
");

        DataReaderWriter drw = new DataReaderWriter();
        rD = drw.ReadData();
        //sadd(jl,BorderLayout);
        this.panel.add(jl);

        this.panel.add(jl21);

        this.panel.add(button);

        JLabel jl2 = new JLabel("----------------------------------------
----------------------------------------------------------------------
-------------------");
        //sadd(jl,BorderLayout);
        this.panel.add(jl2);

        button.addActionListener(this);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JButton b1 = new JButton("Balchen");
        b1.setPreferredSize(new Dimension(130, 30));
        b1.addActionListener(new ActionListener()  {

            public void actionPerformed(ActionEvent e) {

                try {
                    TemperatureControlWindow tc = new
TemperatureControlWindow(rD, 0);
                    tc.setVisible(true);
                } catch (Exception ex) {

Logger.getLogger(MainControl.class.getName()).log(Level.SEVERE, null, ex);
                }

            }
        });
```

```
        this.panel.add(b1);
        b1 = new JButton("Hammer");
        b1.setPreferredSize(new Dimension(130, 30));
        b1.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {
                    TemperatureControlWindow tc = new
TemperatureControlWindow(rD, 1);
                    tc.setVisible(true);
                } catch (Exception ex) {

Logger.getLogger(MainControl.class.getName()).log(Level.SEVERE, null, ex);
                }

            }
        });

        this.panel.add(b1);
        b1 = new JButton("Harklau");
        b1.setPreferredSize(new Dimension(130, 30));

        b1.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {
                    TemperatureControlWindow tc = new
TemperatureControlWindow(rD, 2);
                    tc.setVisible(true);
                } catch (Exception ex) {

Logger.getLogger(MainControl.class.getName()).log(Level.SEVERE, null, ex);
                }

            }
        });

        this.panel.add(b1);

        b1 = new JButton("Sande");
        b1.setPreferredSize(new Dimension(130, 30));
        b1.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {
                    TemperatureControlWindow tc = new
TemperatureControlWindow(rD, 3);
                    tc.setVisible(true);
                } catch (Exception ex) {

Logger.getLogger(MainControl.class.getName()).log(Level.SEVERE, null, ex);
                }

            }
        });

        this.panel.add(b1);
```

```
        b1 = new JButton("Svor");
        b1.setPreferredSize(new Dimension(130, 30));

        b1.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {
                    TemperatureControlWindow tc = new
TemperatureControlWindow(rD, 4);
                    tc.setVisible(true);
                } catch (Exception ex) {

Logger.getLogger(MainControl.class.getName()).log(Level.SEVERE, null, ex);
                }

            }
        });

        this.panel.add(b1);

        b1 = new JButton("Slettemark");
        b1.setPreferredSize(new Dimension(130, 30));

        b1.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {
                    TemperatureControlWindow tc = new
TemperatureControlWindow(rD, 5);
                    tc.setVisible(true);
                } catch (Exception ex) {

Logger.getLogger(MainControl.class.getName()).log(Level.SEVERE, null, ex);
                }

            }
        });

        this.panel.add(b1);

        b1 = new JButton("Gymsal");
        b1.setPreferredSize(new Dimension(130, 30));
        b1.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {
                    TemperatureControlWindow tc = new
TemperatureControlWindow(rD, 6);
                    tc.setVisible(true);
                } catch (Exception ex) {

Logger.getLogger(MainControl.class.getName()).log(Level.SEVERE, null, ex);
                }

            }
        });

        this.panel.add(b1);
```

```
        b1 = new JButton("Badebeseng");
        b1.setPreferredSize(new Dimension(130, 30));

        b1.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {
                    TemperatureControlWindow tc = new
TemperatureControlWindow(rD, 7);
                    tc.setVisible(true);
                } catch (Exception ex) {

Logger.getLogger(MainControl.class.getName()).log(Level.SEVERE, null, ex);
                }

            }
        });

        this.panel.add(b1);

        b1 = new JButton("Ferier");
        b1.setPreferredSize(new Dimension(130, 30));
        b1.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {
                    HolidaysCalender tc = new HolidaysCalender();
                    tc.setVisible(true);
                } catch (Exception ex) {

Logger.getLogger(MainControl.class.getName()).log(Level.SEVERE, null, ex);
                }

            }
        });
        this.panel.add(b1);
        s = count + 1;

        if (rD.size() > count) {

            while (s < rD.size()) {

                b1 = new JButton(rD.get(s).getName());
                b1.setPreferredSize(new Dimension(130, 30));

                b1.addActionListener(new ActionListener() {

                    public void actionPerformed(ActionEvent e) {

                        try {
                            TemperatureControlWindow tc = new
TemperatureControlWindow(rD, s - 1);
                            tc.setVisible(true);
                        } catch (Exception ex) {

Logger.getLogger(MainControl.class.getName()).log(Level.SEVERE, null, ex);
                        }
```

```
                    }
                });

                this.panel.add(b1);

                s++;


            }
        }
        count=s-1;

        setSize(500, 500);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent evt) {

        String m = JOptionPane.showInputDialog("Rom navn:");

        if ((m != null) && (m.length() > 0)) {


        count++;

        int[] arr = new int[42];
        for (int i = 0; i < arr.length; i++) {
            arr[i] = 22;
        }
        RoomData rd = new RoomData(count, m, arr);
        rD.add(rd);
        JButton b1 = new JButton(m);
        b1.setPreferredSize(new Dimension(130, 30));
        b1.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                try {
                    TemperatureControlWindow tc = new
TemperatureControlWindow(rD, count);
                    tc.setVisible(true);
                } catch (Exception ex) {

Logger.getLogger(MainControl.class.getName()).log(Level.SEVERE, null, ex);
                }

            }
        });

        this.panel.add(b1);

        //this.panel.add(new JButton("Room "+count));
        this.panel.revalidate();
                validate();
    }}

}
```

ReadWriterHoliday.java:

```
package temperaturkontroll;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;

/**
 *
 *
 */
public class ReadWriterHoliday {

    ArrayList<HolidayData> hD = new ArrayList<HolidayData>();

    ArrayList<HolidayData> ReadData() throws FileNotFoundException,
IOException {

        String csvFile = "holidaydata.csv";
        String line = "";
        String cvsSplitBy = ",";

        try (BufferedReader br = new BufferedReader(new
FileReader(csvFile))) {

            while ((line = br.readLine()) != null) {


                String[] cards = line.split(cvsSplitBy);



                HolidayData hd=new
HolidayData(cards[0],cards[1],cards[2],Integer.parseInt(cards[3]));

                hD.add(hd);


            }



        } catch (IOException e) {
            e.printStackTrace();
        }

        return hD;

    }

    void WriteData(ArrayList<HolidayData> rt) throws Exception {

        FileWriter fileWriter = null;
        fileWriter = new FileWriter("holidaydata.csv");
```

```
        try {

            for(int i=0;i<rt.size();i++){
            fileWriter.write(rt.get(i).getHoliday());
                fileWriter.write(",");
                fileWriter.write(rt.get(i).getStartDate());

                    fileWriter.write(",");

                    fileWriter.write(rt.get(i).getEndDate());

                    fileWriter.write(",");

                    fileWriter.write(String.valueOf(rt.get(i).getTemp()));



                fileWriter.write("\n");
            }
        } catch (Exception e) {
            System.out.println("Error in CsvFileWriter !!!");
            e.printStackTrace();
        } finally {

            try {
                fileWriter.flush();
                fileWriter.close();
            } catch (IOException e) {
                System.out.println("Error while flushing/closing fileWriter
!!!");
                e.printStackTrace();
            }

        }
    }

}
```

RoomData.java:

```
package temperaturkontroll;


public class RoomData {

    int roomNum;
    String name;
    int [] arr;


    RoomData()
    {

        arr=new int[42];
    }


    RoomData(int rn,String n,int [] ar)
    {
        arr=new int[42];
```

```
        roomNum=rn;
        name=n;
        arr=ar;

    }

    RoomData(int rn,String n)
    {
        arr=new int[42];
        roomNum=rn;
        name=n;


    }


    public int getRoomNum() {
        return roomNum;
    }

    public void setRoomNum(int roomNum) {
        this.roomNum = roomNum;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int[] getArr() {
        return arr;
    }

    public void setArr(int[] arr) {
        this.arr = arr;
    }


    public void setStandard()
    {

        for(int i=0;i<arr.length;i++)
        {
            arr[i]=22;
        }

    }


}
```

TemperatureControl.java:

```
import java.text.SimpleDateFormat;
```

```
import java.time.LocalTime;
import java.util.Date;

/**
 *
 *
 */
public class TemperatureControl {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws Exception {
        // TODO code application logic here

   MainControl mc=new MainControl();

   mc.setVisible(true);
   LocalTime now = LocalTime.now();
   System.out.println(now);
  // LocalTime limit = new LocalTime( "15:30" );
//Boolean isLate = now.isAfter( limit );
String currentTime = new SimpleDateFormat("HH").format(new Date());
int time=Integer.parseInt(currentTime);
if(time>0 && time<5)
{

}
else
    System.out.println(22);
//String timeToCompare = "00:00";

//boolean x = currentTime.equals(timeToCompare);


// DataReaderWriter drw=new DataReaderWriter();
   //drw.ReadData();

    }

}
```

TemperatureControlWindow.java:

```
package temperaturkontroll;

/**
 *
 *
 */
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
import javax.swing.JFrame;
import javax.swing.JButton;
```

```
import javax.swing.JPanel;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import java.awt.FlowLayout;
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

public class TemperatureControlWindow extends JFrame implements
ActionListener {

    int count = 7;
    JPanel panel;
    DateFormat df = new SimpleDateFormat("dd/MM/yy");
    DateFormat df1 = new SimpleDateFormat("hh:mm:ss");

    Date dateobj = new Date();

    DataReaderWriter rw = new DataReaderWriter();
    JTextField[] jt = new JTextField[42];
    ArrayList<RoomData> rD = new ArrayList<RoomData>();

    public TemperatureControlWindow(ArrayList<RoomData> rd, int num) throws
Exception {
        super("");
        rD = rd;
        setLayout(new BorderLayout());
        this.panel = new JPanel();
        this.panel.setLayout(new FlowLayout(1, 5, 5));
        add(panel, BorderLayout.CENTER);

        //  JButton button = new JButton("Add Room");
        JLabel j0 = new JLabel("Skole timer ");
        j0.setPreferredSize(new Dimension(100, 30));
        JLabel j1 = new JLabel("  Mandag ");

        j1.setPreferredSize(new Dimension(70, 30));

        JLabel j2 = new JLabel("Tirsdag");

        j2.setPreferredSize(new Dimension(60, 30));

        JLabel j3 = new JLabel("Onsdag ");
        j3.setPreferredSize(new Dimension(75, 30));

        JLabel j4 = new JLabel("Torsdag ");
        j4.setPreferredSize(new Dimension(75, 30));

        JLabel j5 = new JLabel("Fredag ");
```

```
j5.setPreferredSize(new Dimension(50, 30));

JLabel j6 = new JLabel("Lørdag ");
j6.setPreferredSize(new Dimension(70, 30));

JLabel j7 = new JLabel("Søndag");
j7.setPreferredSize(new Dimension(70, 30));

//        JLabel j8=new JLabel("08:30-9:15");
//sadd(jl,BorderLayout);
this.panel.add(j0);

this.panel.add(j1);
this.panel.add(j2);

this.panel.add(j3);

this.panel.add(j4);

this.panel.add(j5);
this.panel.add(j6);

this.panel.add(j7);
//    this.panel.add(j8);
int count = 0;
for (int i = 0; i < 42; i++) {
    if (i % 7 == 0) {
        if (count == 0) {
            JLabel j9 = new JLabel("08:30-9:15    ");
            j9.setPreferredSize(new Dimension(80, 30));

            this.panel.add(j9);
        }

        if (count == 1) {
            JLabel j9 = new JLabel("9:25-10:10    ");
            j9.setPreferredSize(new Dimension(80, 30));

            this.panel.add(j9);
        }

        if (count == 2) {
            JLabel j9 = new JLabel("10:20-11:05    ");
            j9.setPreferredSize(new Dimension(80, 30));

            this.panel.add(j9);
        }

        if (count == 3) {
            JLabel j9 = new JLabel("11:15-12:00    ");
            j9.setPreferredSize(new Dimension(80, 30));

            this.panel.add(j9);
        }

        if (count == 4) {
            JLabel j9 = new JLabel("12:30-13:15    ");
            j9.setPreferredSize(new Dimension(80, 30));

            this.panel.add(j9);
        }
```

```
                if (count == 5) {
                    JLabel j9 = new JLabel("13:25-14:10    ");
                    j9.setPreferredSize(new Dimension(80, 30));

                    this.panel.add(j9);

                }

                if (count == 6) {
                    JLabel j9 = new JLabel("14:20-15:05    ");
                    j9.setPreferredSize(new Dimension(80, 30));

                    this.panel.add(j9);

                }

                if (count == 7) {
                    JLabel j9 = new JLabel("15:15-16:00    ");
                    j9.setPreferredSize(new Dimension(80, 30));

                    this.panel.add(j9);

                }
                count++;
            }

            String currentTime = new SimpleDateFormat("HH").format(new
Date());
int time=Integer.parseInt(currentTime);
if(time>=0 && time<=5)
{
        int[] arr = rd.get(num).getArr();
            jt[i] = new JTextField(String.valueOf(16));
            jt[i].setPreferredSize(new Dimension(65, 30));
            this.panel.add(jt[i]);

}
else
{
            int[] arr = rd.get(num).getArr();
            jt[i] = new JTextField(String.valueOf(arr[i]));
            jt[i].setPreferredSize(new Dimension(65, 30));
            this.panel.add(jt[i]);

}
        }

        JButton b1 = new JButton("Standard");
        b1.setPreferredSize(new Dimension(160, 30));
        b1.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {
                change(num);

            }
        });

        this.panel.add(b1);

        b1 = new JButton("Lagre");
```

```
        b1.setPreferredSize(new Dimension(160, 30));
        b1.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                int[] arr = new int[42];
                for (int i = 0; i < 42; i++) {
                    arr[i] = Integer.parseInt(jt[i].getText());
                }

                rD.get(num).setArr(arr);
                try {
                    rw.WriteData(rD);
                } catch (Exception ex) {

Logger.getLogger(TemperatureControlWindow.class.getName()).log(Level.SEVERE
, null, ex);
                }
            }
        });

        this.panel.add(b1);

        b1 = new JButton("Gjennomsnitt");
        b1.setPreferredSize(new Dimension(160, 30));
        b1.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {

                int[] arr = new int[42];
                int total = 0;
                for (int i = 0; i < 42; i++) {
                    total = total + Integer.parseInt(jt[i].getText());
                }
                try {
                        JFrame frame = new JFrame();

                    JOptionPane.showMessageDialog(rootPane, "Gjennomsnittet
er: " + (total / 42));
                } catch (Exception ex) {

Logger.getLogger(TemperatureControlWindow.class.getName()).log(Level.SEVERE
, null, ex);
                }
            }
        });

        this.panel.add(b1);

        //setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(670, 500);
        setVisible(true);
    }

    public void change(int n) {

        try {

               int[] arr = new int[42];
                for (int i = 0; i < 42; i++) {
                    arr[i] = 22;
```

```
                }


            rD.get(n).setArr(arr);

            setVisible(false);

            TemperatureControlWindow ab = new
TemperatureControlWindow(rD,n);
            ab.setVisible(true);
        } catch (Exception ex) {

Logger.getLogger(TemperatureControlWindow.class.getName()).log(Level.SEVERE
, null, ex);
        }
    }

    public void actionPerformed(ActionEvent evt) {

        String m = JOptionPane.showInputDialog("Rom navn:");

        JButton b1 = new JButton(m);
        b1.setPreferredSize(new Dimension(90, 30));

        this.panel.add(b1);

        //this.panel.add(new JButton("Room "+count));
        count++;
        this.panel.revalidate();
        validate();
    }

    /*  public static void main(String[] args) {
        TemperatureControlWindow acojfar = new TemperatureControlWindow();
    }*/
}
```

Detaljert plan:

Dato	sum antal timar	Tema
04.01.2017	4	Introduksjon, snakke om vårt prosjekt
05.01.2017	3	Snakkar, og finn ut at vi vil forsette med prosjektet og planlegger korleis vi vil jobbe med prosjekt
06.01.2017	5	Skriving av prosjektbeskrivng
09.01.2017	2	skriving av prosjektbeskriving.
11.01.2017	2	Fullfører prosjektbeskriving
21.01.2017	3	Oppretting av nettstad.
29.01.2017	4	Diskusjon av oppgåve
03.02.2017	3	Diskuterer oppgåve, og sender mail til SFE
07.02.2017	3	Diskusjon av oppgåve, sender mail til fleire bedriftar
14.02.2017	4	Forprosjektrapport
15.02.2017	4	Forprosjektrapport skriv kvar for oss.
16.02.2017	3	Forprosjektrappport
18.02.2017	5	Teorisamling
21.02.2017	4	Samling av teori
23.02.2017	5	Teorisamling
26.02.2017	5	Teorisamling
28.02.2017	4	Teorisamling
02.03.2017	5	Teorisamling
03.03.2017	4	Teorisamling
06.03.2017	4	Programmering (lager plan) og AMS jobbing
09.03.2017	4	Diskutere programmering og AMS jobbing
13.03.2017	5	Begynner programmering og AMS jobbing
15.03.2017	6	Møte, sjekker framdrift med oppdatering
18.03.2016	5	Programmering og AMS jobbing
22.03.2017	7	Programmering og AMS jobbing
26.03.2017	5	Programmering og AMS jobbing
28.03.2017	5	Programmering og AMS jobbing
31.03.2017	6	Programmering og AMS jobbing
01.04.2017	5	Jobber med midtvegspresentasjon
02.04.2017	4	Finn ut kva vi vil seie av arbeidet vårt
04.04.2017	5	Lager presentasjon
05.04.2017	6	Presenterer og jobbar vidare med presentasjon
07.04.2017	5	Programmering og AMS jobbing
11.04.2017	5	Programmering og AMS jobbing
14.04.2017	6	Programmering og AMS jobbing
17.04.2017	6	Programmering og AMS jobbing
19.04.2017	4	Programmering og AMS jobbing
21.04.2017	5	Programmering(fullfører programmering) og AMS jobbing
23.04.2017	6	Mathan og Hatem fullfører prgrammering byrja å skrive rapport.
23.04.2017	6	AMS (Paul)
25.04.2017	6	Rapport
25.04.2017	6	AMS (Paul)
28.04.2017	6	Rapportskriving
28.04.2017	6	AMS (Paul)
02.05.2017	8	Rapportskriving
02.05.2017	5	AMS (Paul)
05.05.2017	6	Rapportskriving
05.05.2017	6	AMS (Paul)
06.05.2017	6	Jobber med hovedpresentasjon + rapport
07.05.2017	7	Oppdateringsmøte
08.05.2017	6	Rapportskriving (hatem og Mathan), Paul (Ams)
09.05.2017	6	Rapportskriving (hatem og Mathan), Paul (Ams), Pressemelding
10.05.2017	7	Styremøte, oppdatering, rapportskriving.
12.05.2017	6	Plakat + rapport
13.05.2017	8	Rapportskriving
15.05.2017	8	Rapportskriving
16.05.2017	8	Rapportskriving
18.05.2017	8	Rapportskriving

Møter

Prosjektbeskriving

Januar 6. 2016

Personar til stades: Mathan og Paul.

Møte 1 12:30-16:00

I dag møtast Paul og Mathan til å skrive prosjektbeskriving. Hatem var dessverre ikkje her, grunna han jobba i Bergen. Dette skuldast faget styrt praksis. Men Mathan og Paul lojale og jobba med oppgåva. Vi starta med å planleggje kva vi skulle skrive, men sidan vi hadde god nok erfaring fra forrige semester, so var vi allereie ganske greit i mål. Vi skreiv prosjektbeskrivinga ferdig som skulle inn den 12.01.17.

Det meste skreiv vi på dagen, men det som sto igjen, diskuterte vi over nett, og fann ut kva vi skulle levere. Vi fann ut at neste møte sannsynligvis ikkje kjem til å vere før en stund, fordi det er ein stund til vi skal levere forprosjektrapport. Ellers kan De finne både prosjektbeskrivinga og ganttskjema vedlagt på nettsida.

Forprosjektrapport.

Februar 14, 2017

Personar til stades: Mathan Paul, og Hatem

Møte 2 12:30 - 16:00 Grupperom Haarklou

I dag møtast alle saman på grupperommet Haarklou. Dette var første gongen vi hadde anledning til å møte kvarandre fysisk, og vi diskuterte alle idéeane vi hadde so langt. Vi begynte med ein gong og skrive ned alt vi har samla i januar månad og i noko av februar og begynte å forme forprosjektrapporten vår.

Det kan takast med at denne websida var opna 28 januar 17. Paul har sendt diverse mailer til SFE og andre selskap og skal soleis komme fram til kva han skal jobbe med spesifikt i prosjektet i løpet av nokre få dagar. Vi begynner å skrive på forprosjektrapporten som skal inn førstkommande fredag, og etter vi har skrive ferdig til klokka 16, ser det ut som vi har eit godt grunnlag til å skrive ho ferdig på fredag.

På slutten av møtet vart det gitt ut ulike oppgåver til kvar av oss, som vi skal gjere ferdig til torsdag, slik at vi kan få med dette i forprosjektrapporten.

Forprosjektrapport II
Februar 16, 2017

Personar til stades: Mathan Paul, og Hatem

Møte 3 13:00-16:00 Grupperom Haarklou

I dag samlast vi til møte for å fullføre forprosjektrapporten. Hatem følte seg sjuk, men han klarte å komme seg til skulen likevel, og det satte Paul og Mathan pris på:). Ved hjelp av å skrive på google docs, vart dette effektivt, og vi kunne redigere og skrive tekst samtidig. Dette kunne alle tre gjere, på det same dokumentet, og soleis gjekk det mykje fortare.

Etter å ha skrive det ferdig, las vi gjennom det ein gong. Då vart det som i alle tilfeller, litt meir korrektur i teksten, men vi endte med å fullføre teksten i dag. Med informasjonen som Paul fekk over mail, so er han no ganske sikker på kva han skal jobbe med, og vi fekk spesifisert oppgåva. Vi skreiv dette og i forprosjektrapporten som vi leverte inn.

Etter vi vart ferdige, merka vi at vi er ganske klare med å begynne skikkeleg med oppgåva, so dette ser bra ut (i hvert fall i denne augneblink!

Programmering av oppgåve.
Møte 4
Mars 6, 2017

Personar til stades: Mathan og Hatem

Grupperom Haarklou 11:00-12:00
I dag bestemte vi for oss å endeleg begynne med programmeringa. Mathan og Hatem møtte kvarandre, og utarbeida eit plan. Denne planen skisserte vi på paint, og vi fekk dermed ein klar idê på korleis vi ville bygge opp dette. Vi bestemte oss på å lage ein plan som vi brukte til å utarbeide oppgåva. Å lage ein plan er viktig, slik at vi ikkje jobbar utan å ha ein framgangsmåte. Dessverre så gløymte vi å bestille rom i dag, og vi vart jaga ut og måtte finne eit nytt rom!

Dahl 12:00-13:15
Vi forsatte å jobbe med planleggjinga og klarte å komme fram til kva vi ville lage i programmet vårt i java. Vi har då gitt kvarandre ulike oppgåver, og dette skal vi då fullføre til vi møtast neste gong. Dagen gjekk veldig bra og vi har då fordelt kvarandre oppgåver.

Samlingsstund
Mars 22, 2017

Personar til stades: Mathan Paul, og Hatem

Møte 5 Grupperom Haarklou 16:00-18:00

I dag møtast alle på grupperommet Haarklou. Vi samanlikna det vi har gjort, og ser at vi har begynt å komme mot eit mål i programmeringa. Mathan og Hatem har jobba med programmeringa og det ser ganske bra ut. Paul driv vidare og undersøkjer AMS biten, og ringer rundt omkring og samlar informasjon frå ulike kraftverk. Vi diskuterte kor langt vi har kome, og såg nærmare på dei ulike fristane vi skulle nå.

Vi ser at det skal vere ein midtvegspresentasjon i byrjinga av mai, og kontakta Joar om å flytte tida tidlegare på dagen fordi det ikkje passar for alle. Vidare sendte vi og melding til Ole Gaute som er oppdragsgjevaren vår frå SFE. Vi samanlikna resultata innan programmeringa, og ser kor langt vi har kome innan det.

No er det meir i tankane på kva vi skal presentere foran klassen i byrjinga av april, og vise kva vi har gjort til no til dei.

Rapportskriving
May 9, 2017

Personar til stades: Mathan Paul, og Hatem

Møte 6 Sivle 10.15-18.00
Idag skriv vi rapport, og vi er på god veg til å vere ferdige. Meste av arbeidet er gjort, og ser ut som vi skal vere ferdige godt før tidsfristen. All det teoretiske og praktiske arbeidet er ferdig, og vi treng berre å skrive ned dette no.

Innkalling til styringsmøte

Tema : Møteinnkalling om status i prosjektarbeidet til studentene ved HISF

Mottakar : Faglærar, Oppdragsgjevar og medlemmer av studentgruppa.

Hei!

Vil med dette kalle inn til statusmøte for prosjektet vårt.

Møtetidspunkt : Mandag 10.05.17 2017.

Foreslår møte tidspunkt frå kl 09:30-09.50. Reknar med du Joar kjem med tilbakemelding på om det passer.

Mvh Paul Thomassen for prosjektgruppa.

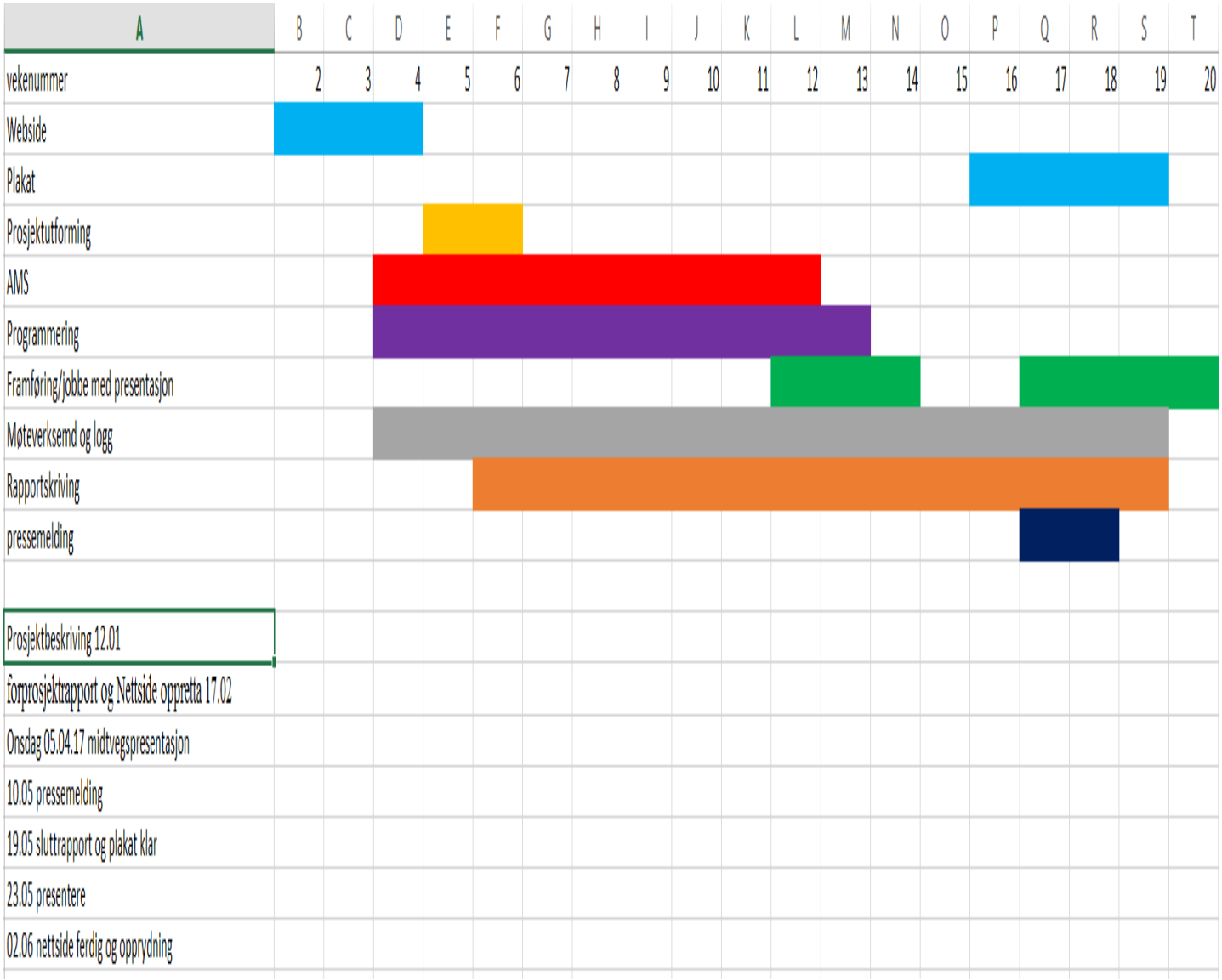
Referat Styremøte

Deltakarar: Joar Sande, Mathan Manivannan, Paul Thommassen, Hatem Awadallah, Ole Gaute Hovstad og Sven Arild Kjerpeset.

Vi starta møte med å informere alle partar om kva vi har jobba med i prosjektet. Paull fekk meir innsikt i kva han skal skrive om, og fekk oppklart kva han skal jobbe med dei siste dagane. Mathan og Hatem var allereie ferdig med programmeringsbiten, og informerte vidare til SFE-rettleiarane om kva dei har jobba med og at dei er godt igang med prosjektet. Joar informerte oss om viktige tidsfristar og korleis vi skulle bruke den siste tida vår, vidare fekk vi beskjed om å sende rapporten til han om vi trenger hjelp. Møtet varte i 20 min frå 09:30-09:50.

Referent: Mathanraj Manivannan

Gantskjema:


A
B C D E F G H I J K L M N O P Q R S T
vekenummer
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Webside
Plakat
Prosjektutforming
AMS
Programmering
Framføring/jobbe med presentasjon
Møteverksemd og logg
Rapportskriving
pressemelding
Prosjektbeskriving 12.01
forprosjektrapport og Nettside oppretta 17.02
Onsdag 05.04.17 midtvegspresentasjon
10.05 pressemelding
19.05 sluttrapport og plakat klar
23.05 presentere
02.06 nettside ferdig og opprydning

Planlagd timesbruk

Timebruk:

Planlegging: 20 t ¨

Forprosjektrapport 10-15 t

Teorisamling: 50-60 t

Utføring og praktisk arbeid: 60 t

Rapportskriving: 70-80 t

Totalt timesbruk: 235 t

Reel tidsbruk

Introduksjon og plannlegging: 21 t

Nettstad : 3t

Forprosjektrapport og diskusjon om oppgåve: 22t

Teorissamling: 32 t

Utføring og praktisk arbeid, Programmering 60 t, Mathanraj, Hatem 72 t programmering, Paul AMS 101 t.

Rapportskriving Mathanraj 85 t, Hatem 76 t og Paul 50 t

Møter, pressemelding, powerpoint, oppdatering av nettstad og plakaat 36 t

Total timesbruk:

Mathanraj Manivannan = 259 t, Hatem Awadallah 262t, Paul Thomassen 265 t.

Risikoanalyse

	A	B	C	D	E	F	G	H	I	J
1	Risikoanalyse:		Årsaker	Konsekvens	Sannsyn	Risikofaktor	Risiko	Tiltak		
2		1	Stjeling av materiale	Stor	Låg	5	Låg			
3		2	Ikkje nå tidsfristar	Stor	Middels	15	Høg	Jobbe tidleg og effektiv		
4		3	Dårleg kommunikasjo	Stor	Svært låg	10	Middels	Bruke sosiale medier, kommunisere ofte med kvarandre med skype telefon.		
5		4	Ikkje fullføre program	Stor	Svært låg	10	Middels	Studerer programmering, spør faglærar om vi sit fast.		
6		5	Offer for virusprogran	Stor	Låg	5	Låg			
7										
8										
9										
10										

Pressemelding

Styringssystem med AMS

Paul Thommasen Mathanraj Manivannan og Hatem Awadallah

Vi har i vår oppgåve laga ein automatisert styring ved bruk av java. Dette prosjektet omhandla Smart Valley som SFE har jobba med. Gruppa vår er både frå automasjon og elkraft, so i elkraftbiten har vi sett på AMS (automatiserte straummålera) og i automasjonsbiten har vi sett på programmering. Smart valley handla kort sagt om å effektivisere straumbruken ved skulane.

Hatem Awadallah, hatemnor@hotmail.com 90751669
Heimeside: https://z94mathan.wixsite.com/smartvalley

Plakat

Innleiing:
Hovedmålet vårt er å lage eit program for eit digitalt system som har ei framtidsretta løysing. Her får ein illustrert dei automatiserte målarane sine funksjonar og korleis ein får brukt styringssystemet til å spare energiforbruket til kunden nettselskapa oppnår ei jamnare energibelastning i straumnettet.

Mål:
Hovedmålet vårt er å lage eit program for eit digitalt system som har ei framtidsretta løysing. Her får ein illustrert dei automatiserte målarane sine funksjonar og korleis ein får brukt styringssystemet til å spare energiforbruket til kunden nettselskapa oppnår ei jamnare energibelastning i straumnettet.

Metode:
Systemet skal kunne styrast med bruk av timeplan der ein kan kople inn og ut dei ulike klasseromma inkludert symjebassenget. Dette programmet tenkjer vi å lage i JAVA, der brukaren sjølv kan regulere temperaturen etter sine eigne ønskjer. Dette vil då dekkje automasjonsbiten av prosjektet.
I elkraftbiten av prosjektet, vil vi bruke smart-teknologi(saman med AMS) til å utvikle ein tariffmodell som er meir rettferdig for den kunden som vil være aktiv med å oppnå spart forbruk og å være miljøbevisst.

Resultat:
Oppgåva som vi fekk frå SFE var å lage ei programert styring for ein skule. I programmeringa har vi laga eit program i Java kor vi styrar ulike rom etter ulike temperaturar. Dette har vi gjort vi basert på kva for eit bygg det er, til dømes vil vi gjerne ha det litt varmare i eit symjebassenget. Romma i skulane er: Symjebassenget, ulike klasserom, gymsal svømmebassenget osv.
Sjølve programmet er laga ved hjelp av Java GUI. I AMS biten har vi sett på smarte løysingar som vil gje oss ny kunnskap og kva som skjer rundt oss. Dette henger saman med effektiv utnytting av straum slik at vi skapar forbrukarfleksibilitet.

Smart Valley

Oppdragsgiver:
SFE

Styringsgruppen:
Joar Sande
Nils Westerhei
Ole Gaute Hovstad

Gruppemedlemer:
Hatem Awadallah
Mathanraj Manivannan
Paul Thomassen

Prosjektbeskrivelse

«Styringssystem med AMS»

Målet vårt er å lage eit prosjekt/modell evt. analyseprosjekt som er samfunnsnyttig og som viser forbrukarfleksibilitet i å formidle til det framtidige energisystemet. Sidan vi hadde dette prosjektet før jul, so har vi tileigna kunnskap innan dette før jul.

Hovedmål:

Hovedmålet vårt er å lage eit modell til Smart Valley, for å effektivisere varmen og redusere kostandene. Dette gjer vi ved bruk av AMS og programmering.

Delmål:

Lage ein modell som simulerer skulen so skilen si varmeløysing med både klasserom og symjebasseng.

- De vurderar sjølv om det skal vere ein virtiuell modell eller ein fysisk modell.
- De vel sjølv kva teknologi de vil bruke for å lage dette. (Elektronikk, HTML, Java, LabView, osv...)
- De må og vurdere kor kompleks de vil at den skal vere, t.d. om de skal inkludere varmetap og ta omsyn til utetemperatur og klima/ årstid.

Lag ei styring som optimaliserar energikostnad og nettleige ved å kople inn og ut varmesoner og elektrokjel.

- De må forstå mekanisme i energibransjen, med produsentar, karftbørs, kraftleverandørar og nettselskap, og korleis energipris og nettleige er bygd opp.
- I dag har nettleiga ein fast formel, fastbeløp + energipris x kWh + effektpris x KwH, men vi reknar med at effektprisen i framtida vil kunne variere pr time.
- De må ta omsyn til skulen sin timeplan for dei fem ulike varmesonene pluss timeplan for symjebassenget
- De vel sjølv kva andre parametrar de vil ta omsyn til, for å lage ei best mogleg
- Dersom de t.d. tar omsyn til utetemperatur kan historikk og varsel hentast frå eKlima eller likande.
- Kanskje de og vil legge til rette for forbrukarfleksibilitet, der kunden kan selje kapasitet til nettselskapet, gjennom at nettselskapet får redusere effektgrensa ved behov...? De vel sjølv kva teknologi de vil bruke for å lage dette. (Elektronikk, HTML, Java, LabView, osv...)

Lag eit brukargrensesnitt som viser status, og lar brukaren endre parametrar. Brukargrensesnittet må minimum vise status

- Det bør vere mogleg å endre timeplanar, grense for maks effekt og liknande.
- De vel sjølv kor mykje anna de vil legge inn. (Rapportar for energi og effekt, energikostnad, spart energikostnad osv...)
- De vel sjølv kva teknologi de vil bruke for å lage dette (Elektronikk, HTML, Java, Labview osv.)

Rammer:

- 125 timar kvar person, fra januar-mai, usikker på ressursar og kostandar foreløpig. Vi har jobba med denne oppgåva i forrige semester
- Vi har allereie avgjort korleis vi skal jobbe med dette og funne ut korleis vi tenkjer å programmere dette vidare. Paul har drøfta med oppdragsjevar og komme fram til korleis han skal løyse elkraftbiten i oppgava.

Faseinndeling:

- Vi tenkjer å ha eit prosjekt som har både sitt preg av automasjon og elkraft, og prøva å få til ei ordning på 50/50 ordning på det. Vi jobbar vidare med det vi jobba med forrige semester, og skal lage programmet til oppgåva i java og jobbe vidare med AMS biten.

- Vi tenkjer forsatt på ein modell på 40 % programmering, 20 % displlay, 20 % webside/virtuell og resten 20 %.

Oppgåveavgrensing:

Det er frie tøyler og vi følgjer dei rammane som vi har fått av vår oppdragsgjevar og SFE.

Organisering:

Gruppeleiar: Paul Thomassen (men vi tenkjer å byte på rolla som gruppeleiar)

Mathanraj Manivannan og Hatem jobbar med programmering, og Paul har hovedansvar for AMS biten.

Oppdragsgjevar: SFE

Stryingsgruppe: SFE og Joar Sande, Nils Westerheim.

Paul - hovedansvar på webside og AMS biten, Hatem og Mathanraj, har hovedansvar for programmering.

Gjennomføring og framdriftsplan:

Vi leggjer ved eit gantskjema, som forklarar dette nærmare

Kostnad og budsjett:

Usikker på kostander endå, og lar det dermed stå open.

Risikoanalyse og kvalitetssikring:

- Vi ønskjer og ha eit oppfølgingsplan som har status på kor langt vi har komme i prosjektet, som vi evaulerar kvar veke.
- Ufordringar som kan oppstå er manglande kunnskap, informasjon og dårleg tid.
- Viss det blir dårleg kjemi i gruppa.
- Viss ein av oss blir langtidssjuk.
- Vi kvalitetsikrar dokumenta ved å nummere dei, og tar ein backup av det.
- Dårleg kommunikasjon mellom oss og bedrift, og andre verksemd.

Korleis forebyggje dette?

- Vi må ha faste møtedagar kvar veke.
- Motivere kvarandre, ha det sosialt og gje ros til kvarandre.
- Sørgje for at vi følgjer Gannt skjema og evaluere prosjektet, slik at vi holde framdriftsplana våre.
- Oppsummeringsmøte på fredag der vi går gjennom kva vi gjorde denne veka , og lagar planen for neste veke. På denne måten for vi ei god gjennomføring.