Høgskulen på Vestlandet

BACHELOROPPGAVE:

BO21E-21 AUTOMATISK TUNING AV REGULERINGS-PARAMETERE FOR BYGG

Martin Jacobsen
Magnus Kristiansen
Helene Pisani

1. jun. 2021

Dokumentkontroll

Rapportens tittel: BO21E-21 Automatisk tuning av regulerings-parametere for bygg	*Dato/Versjon* 1. jun. 2021/1.0
	Rapportnummer: BO21E-21
Forfatter(e): Magnus Kristiansen Martin Jacobsen Helene Pisani	*Studieretning:* AUTH18
	Antall sider m/vedlegg 51
Høgskolens veileder: Ingvar Henne	*Gradering:* Åpen
Eventuelle Merknader: Vi tillater at oppgaven kan publiseres.	

Oppdragsgiver: GK Norge AS	*Oppdragsgivers referanse:*
Oppdragsgivers kontaktperson(er) (inklusiv kontaktinformasjon): Jan Gunnar Ludvigsen Tlf: 95 00 55 10 E-post: jan-gunnar.ludvigsen@gk.no	

Revisjon	**Dato**	**Status**	**Utført av**
0.11	20.01.21	Første utkast	Helene, Martin, Magnus
0.2	28.01.21	Korreksjon etter møte med veileder	Helene, Martin, Magnus
0.3	28.01.21	Dokument levert for evaluering av veileder	Helene, Martin, Magnus
1.0	31.05.21	Dokument klart for innlevering	Helene, Martin, Magnus

Forord

Denne rapporten er resultat av bacheloroppgave gjort av tre studenter på siste semester av automatiseringslinjen ved Høgskolen på Vestlandet, campus Bergen. Oppgaven er levert av GK.

Da de ulike bacheloroppgavene ble presentert ble vi alle tre fort enige om å velge GK sin. Det virket som en spennende og god mulighet for å lære mer om regulering i praksis. Valget ble styrket ettersom GK er et anerkjent firma.

En stor takk til GK som lagde denne spennende oppgaven og for å ha tatt oss godt imot. Vi har fått vært med på anleggsbesøk, benyttet oss av interne systemer og har gjennom Jan Gunnar Ludvigsen fått møtt dyktige fagpersoner. Det har vært veldig gøy og lærerikt. En ekstra stor takk til den engasjerte veilederen vår Jan Gunnar som har lært og vist oss så mye om systemene til GK. Han har alltid stilt opp og hjulpet oss underveis gjennom hele perioden.

Takk til Vidar Løtveit, Øystein Gaukstad samt Jan Gunnar som lot seg intervjue og delte sin kunnskap og erfaring med oss.

Takk til den interne veilederen Ingvar Henne for god veiledning til skriving av bacheloren.

Sammendrag

Ventilasjonsstyring er et omfattende tema. Flere ulike komponenter spiller sammen om å levere ren, temperert luft i for eksempel kontorlokaler. Hele denne prosessen er styrt av reguleringsteknikk som igjen bygger på bruken av PID-kontrollere. Problemet er å få reguleringsparameterne til å drifte systemet optimalt, noe som ofte er et tilfelle. Det er ulike faktorer som påvirker tilstanden til anlegget som også kan endres over tid. Dette er faktorer som utetemperatur, utskiftning av utstyr eller varierende varmebehov. I tillegg vil også ombygninger av et bygg være med på å endre stabiliteten i reguleringsanlegget.

Vanligvis blir disse reguleringsparameterne stilt inn manuelt gjennom prøving og feiling, basert på erfaring. Dette kan være en omfattende og treg prosess og det blir gjerne ikke brukt nok tid på innregulering av anlegg. Dette kan da føre til at systemet blir ustabilt og pendling vil forekomme. Ved å lage et program, som ved hjelp av loggdata, detekterer pendling vil man kunne avdekke hvor problemet kan ligge. Dersom programmet i tillegg foreslår en endring av reguleringsparametere vil man kunne få et system som optimaliserer tuningen av anlegget.

Programmet er nå en programblokk i GK sitt system hvor den får informasjon fra systemet den opererer i. Her leser den loggdata og der hvor den oppdager pendling vil den foreslå en endring av reguleringsparameterne. En bruker må så godkjenne endringen før de vil bli tatt i bruk.

Innhold

Figurliste

1 Innledning

1.1 Oppdragsgiver

GK er Skandinavias ledende tekniske entreprenør og servicepartner og leverer smarte løsninger innen ventilasjon, kulde, byggautomasjon, elektro, rør og sikkerhet. Med over 3100 ansatte i Norge, Sverige og Danmark har de en omsetning på 6.2 milliarder kroner i året [1]. GKs verdier er inkluderende, nysgjerrig og ansvarlig. Hos GK jobber flere fagfelt sammen for å skape det beste produktet for kunden. De ønsker at varen de leverer skal skape verdi for generasjoner.

Energi brukt til oppvarming og ventilering av kontorbygg, skoler og kjøpesenter står for en stor del av energiforbruket hos byggene. GK ønsker å levere smarte løsninger som kan bidra til lavere energiforbruk og dermed mindre klimagassutslipp.

1.2 Problemstilling

Når en skal sette opp et nytt varme og ventilasjonsanlegg i bygg er det flere reguleringsparametere som må stilles inn. Vanligvis blir disse plottet inn for hånd ved testing og feiling før man finner riktige verdier til det spesifikke systemet. Innreguleringen vil dermed variere fra anlegg til anlegg og fra person til person som utfører jobben.

Selv om en fikk til en god innregulering kan det oppstå hendelser over tid som kan føre til at en bør omjustere på systemet. For eksempel utskifting av utstyr, ombygging eller bare så enkelt som at utetemperaturen endrer seg med årstidene. Når et anlegg kjører med feil regulering, vil det føre til unødvendige kostnader for kunden og slitasje på utstyr. Dette er ikke ønskelig.

1.3 Hovedidé for løsningsforslag

Ved å lage et program som analyserer loggdata fra anleggene, vil man kunne oppdage pendling i reguleringen til de forskjellige prosessene. Vi vil lage et slikt program som henter csv filer fra anleggene som skal sjekkes. Programmet skal i første omgang gå gjennom data fra en tilstrekkelig lang periode for å se om det har oppstått pendling og deretter gi beskjed/alarm om det er dårlig regulering. Når programmet klarer å detektere pendling skal det ved videre analyse av loggdata gi forslag til parametere for regulering slik at systemet blir stabilt. På denne måten vil anlegget sannsynligvis driftes med lavere kostnader og ha mindre slitasje på utstyr, samtidig som at inneklimaet igjen blir best mulig.

2 Kravspesifikasjon

Oppdragsgiver la frem fire mål for oppgaven. Det første er å utarbeide en algoritme som skal avdekke uheldig pendling eller treg regulering, basert på loggdata. Det neste målet innebærer implementasjon av det ferdige programmet. Her vil vår veileder, Jan Gunnar, bistå med hjelp ettersom koden behøver en videre implementasjon til GK sitt system, Niagara. For å redusere feil samt begrense omfanget av arbeid vil dette være fremgangsmåten. For disse målene har det blitt satt visse krav som skal opprettholdes.
De er som følgende:

- Loggdata skal hentes ut fra GK sitt system, Niagara. Dette skal være en CSV-fil.
- Filen skal inneholde et tidsstempel og en verdi.
- Programmet skal hente og lese denne filen.
- Programmet skal kunne sortere informasjon fra hvert datapunkt og prosessere dette.
- Programmet skal ved en gitt verdi i prosent registrere et avvik.
- Resultater skal loggføres av programmet og verdier skal være tilgjengelige for bruk i regulering.
- Koden skal være forståelig og kommentarer skal være presise slik at det ikke oppstår misforståelser ved videre implementering.

Det tredje målet, derimot, omhandler bruk av maskinlæring og ettersom det er et omfattende fagfelt vil ikke dette målet bli prioritert av GK. Ettersom gruppen ikke har tilstrekkelig kunnskap om emnet fra tidligere vil det ikke forekomme en implementasjon av maskinlæringskode. Gruppen vil prøve å sette seg best mulig inn i emnet på egenhånd og formulere muligheten for bruk av maskinlæring i GK sine anlegg.

Uheldig regulering kan føre til enkelte problemer. Det kan være økt energiforbruk samt slitasje på deler i reguleringsanlegget. Det siste målet vil derfor være å gi en utreding på hvordan dette skjer, i motsetning til en prosess hvor drift er optimalisert og pendling er redusert.

3 Analyse av problemet

Oppdragsgivers hovedidé for løsningsforslag		
#	Delmål	Løsningsforslag
1	Utarbeide en algoritme, basert på loggdata, for å avdekke pendling eller treg regulering.	Skrive kode i C# som mottar en fil med reell loggdata.
2	Endre reguleringsparametere ved å implementere algoritmen i GK sitt system.	Vår eksterne veileder, Jan Gunnar Ludvigsen, skal bistå med konvertering fra C# til GK sitt system, Niagara.
3	Benytte maskinlæring for å identifisere hvilke faktorer som påvirker de forskjellige reguleringsprosessene.	Drøfte muligheter og skissere ideer for en reell løsning. Dersom det blir mulig, kan en eventuell implementering av dette forekomme.
4	Utrede konsekvensen av uheldig, kontra gunstig, regulering.	Teste forskjellige reguleringsparametere for å kartlegge effektkost av ulike segmenter i et reguleringsanlegg.

Figur 1: Analyse av problemet

3.1 Utforming av mulige løsninger

3.1.1 Valg av programmeringsspråk

Hovedoppgaven går ut på å lage et program. Hvilket språk vi skulle skrive dette programmet i ble det første spørsmålet vi måtte stille oss. GK bruker selv programmeringsspråket Java. Vi sto mellom å bruke deres eget, Python eller C#. Etter samtaler med veilederen vår ble vi enige om at å sette seg inn i et nytt språk er tidkrevende og tiden vi har til bacheloroppgaven er ikke tilstrekkelig. Løsningen vi kom fram til var at vi skriver programmet i et språk vi mestrer, C#, også vil han hjelpe oss å skrive om programmet slik at det passer i GK sine systemer.

3.1.2 Løsning

Gruppen får opplæring av GK til bruk av programvare som skal brukes når kildekode skal iverksettes som en løsning for GK. Programvarer som skal brukes er Niagara, Centraline og GKCloud. GK skal også vise gruppen et typisk anlegg som koden skal brukes på, slik at gruppen kan lage en best mulig løsning.

Programmets funksjon, kort forklart, er å detektere utslag i verdier basert på loggdata. Dette vil skje ved at programmet mottar en CSV-fil med loggdata. Denne dataen inneholder blant annet en verdi og et tidsstempel.

Videre vil dataen bli sortert hvor så hver verdi vil bli sjekket opp imot en referanseverdi. Referanseverdien vil være en bestemt verdi satt av en bruker, for eksempel en ønsket temperatur. En annen løsning vil være å la programmet basere seg på et gjennomsnitt i stedet for en gitt verdi. Da vil programmet kontinuerlig oppdatere gjennomsnittet for hvert punkt som kommer inn.

Programmet vil så lagre en verdi dersom den har for stor differanse fra referanseverdien. Dette kan være en naturlig årsak og ikke pendling, så ved å sjekke tidsstempelet ved neste utslag mot det forrige kan man så se om det skjer periodisk. Dersom en trend i utslag oppdages vil programmet så foreslå en endring av PI-parametere slik at pendlingen blir redusert.

For å utarbeide denne koden må gruppen sette seg inn i hvordan PID regulering fungerer på VVS anlegg slik at gruppen har en forståelse for dette. Prosessen begynner med å utarbeide en skisse over

hvordan koden kan se ut før selve arbeidet med å skrive koden starter. Etter at første utkast av programmet er ferdig, skal det testes på reell loggdata. Dersom det fungerer godt og klarer å detektere pendling kan vi utvide koden til å foreslå nye reguleringsparametere. Dette skjer i andreutkast av koden. Til slutt vil vi forsøke å få programmet til å oppdatere regulatoren med de nye parameterne.

Når arbeidet med å skrive programmet er godt i gang kan gruppen også begynne å se på krav tre og fire. Det skal da undersøkes hvordan maskinlæringsalgoritmer fungerer og hvordan det kan brukes til å identifisere faktorene som får reguleringen til å drifte. I tillegg skal gruppen også drøfte hvordan dårlig regulering påvirker driftskostnader og komponenter med tanke på slitasje. Dette er noe gruppen kan begynne med når de har sett på anlegg i praksis sammen med GK, og/eller etter første utkast av koden.

3.1.3 Vurderinger i forhold til verktøy og HW/SW komponenter

HW/SW	Beskrivelse
Visual Studio (SW)	Brukes for å skrive C#-koden hvor logikken vil bli brukt i sluttproduktet. Ettersom utdanningsforløpet har lagt opp til undervisning i dette programmet, er det dette som vil bli brukt.
Virtual Box (SW)	Oppretter tilgang til GK sine systemer som Centraline og Niagara via en virtuell datamaskin.
Niagara (SW)	Rammeverk for styring og et grafisk grensesnitt. Programmet, som er basert på Java, utfører både HMI og logikk.
Centraline (SW)	Et integrert system som er bygget på Niagara. Systemet viser de forskjellige anleggene til GK, samt koden som ligger bak reguleringene.
GK Cloud (SW)	Skyløsning for blant annet lagring av historikk og fjerninteraksjon med anleggene. Data fra flere anlegg samles også her.
PLS (Jace 8000) (HW)	IO modul med logikk. Brukes til å kjøre Centraline på anleggene.

3.2 Konklusjon

Ettersom gruppen har grei erfaring innenfor C# og Visual Studio vil dette derfor bli brukt for å utarbeide koden. Sluttproduktet må være tilpasset GK sitt system, men ettersom Jan Gunnar vil bistå med hjelp til videre implementering vil dette ikke være et problem.

I tillegg til å utarbeide logikken for programmet trengs det også en del teori og kunnskap om reguleringsanlegg. En del av denne teorien må tilegnes før selve arbeidet med koden starter. Videre vil man opparbeide en dypere forståelse for temaet og dermed forstå hva som trengs av teori. Utarbeiding av kode og tilegning av teori vil derfor skje simultant om hverandre. Dette er også en god løsning med tanke på tidsomfanget på oppgaven.

Gruppen har valgt å kun bruke ett løsningsalternativ da oppgaven er såpass spesifikk, og ønsket fra GK er å få et program som tidligere beskrevet. Samtidig stiller GK seg ganske fritt til å bistå med hjelp hvis det trengs.

4 Realisering av valgt løsning

4.1 Teori

4.1.1 HVAC

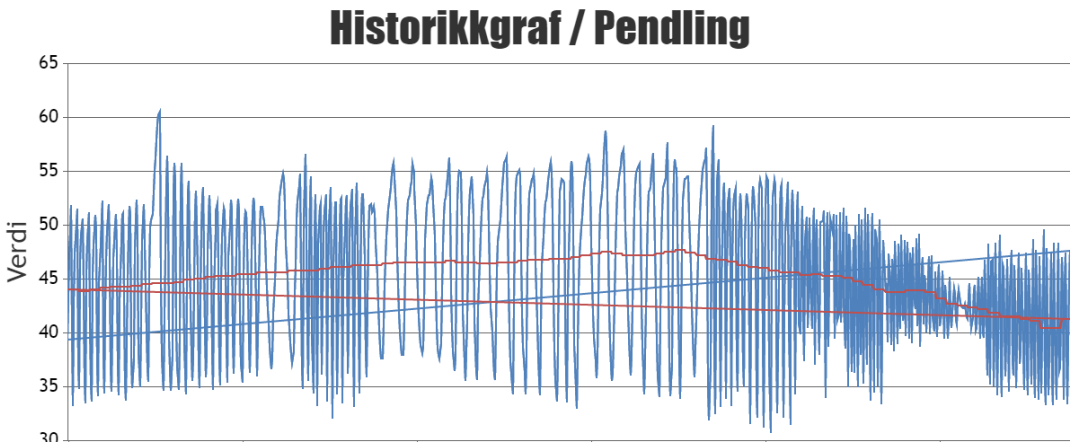
HVAC står for "heat, ventilation and air conditioning". På norsk kjent som VVS: varme, ventilasjon og sanitærteknikk. VVS anlegg brukes i større bygg med masse mennesker, for eksempel skoler, kontorbygg, kjøpesentre eller leilighetskomplekser, og skal bidra med varme, kjøling og ventilasjon.

Et godt tegnet, installert, testet og vedlikeholdt varme- og ventilasjonssystem leverer en behagelig temperatur og god regulert luft for brukerne samtidig som det holder kostnader og energiforbruk nede. God regulert luft betyr at luften skal være ren, og at temperatur, luftfuktighet og viftestyrken skal være innenfor akseptable verdier. [2]

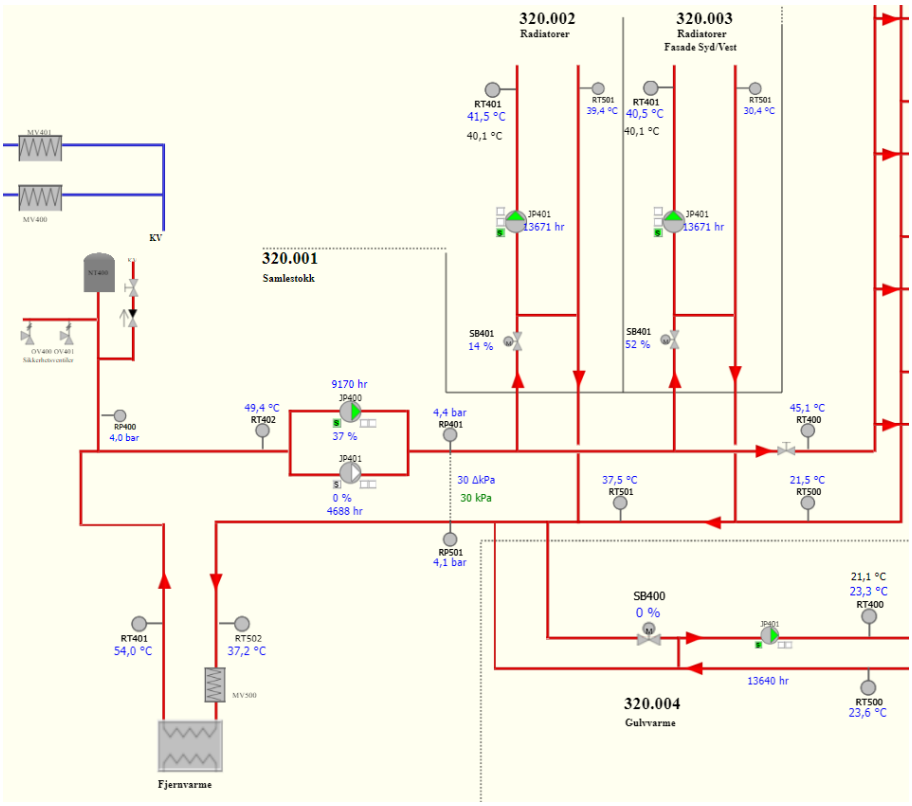
HVAC anlegg består av flere komponenter som jobber sammen for å skape det perfekte inneklima. Hvilke komponenter som blir brukt og hvordan et slikt anlegg ser ut varierer ut ifra klima, designer eller leverandør, budsjett, bygningens alder og arkitektur. [3]

4.1.2 Pendling

Pendling oppstår på grunn av at systemet er ustabilt og man vil få en respons som overreagerer på endringer, og amplituden for avviksverdien blir stor. Ustabilitet kan komme av feil valg av PID parametere. Dette vil programmet vårt kunne løse. Ustabilitet i reguleringssløyfer kan også komme av feildimensjonering av komponenter, denne type ustabilitet vil ikke programmet kunne gjøre noe med. For eksempel hvis akkumulatortanken til et varmeanlegg er for liten, i forhold til forbruket, vil temperaturen bli ujevn og reguleringen til forbrukerne vil sannsynligvis pendle.



Figur 2: Figuren viser et eksempel på pendling. Den røde grafen viser settpunkt og den blå grafen viser måleverdier. (se bort ifra de to rette grafene). Bildet er hentet ut ifra programmet vårt hos GK.



Figur 3: hvis NT400 er for liten vil muligens 320.002 og 320.003 pendle. Bildet er hentet ut ifra GK sitt system.

Utfordringen med VVS systemer er at det er en mekanisk treghet med. For eksempel oppvarming av luft eller at viftehastighet skal oppnå ønsket volum per tid eller trykk. Anleggene er ikke nødvendigvis ustabile hele tiden, men ofte i begrensede tidsintervaller på grunn av ytre påvirkninger slik som utetemperatur, temperatur på fjernvarme eller flere forbrukere på samme reguleringssløyfe.

4.1.3 PID Regulering

Når noe skal reguleres eller styres vil ofte en regulator bli tatt i bruk. Denne bruker ofte PID regulering. PID regulering kan forklares ved at det er tre bidrag som utgjør pådraget til regulatoren. Disse tre bidragene er: proporsjonalleddet, integralleddet og derivasjonsleddet. Dette betyr at man har et ledd som gir et bidrag som korrelerer direkte ved avviksverdien, altså P-leddet. I- og D-leddene er funksjoner av avviket og tiden der I-leddet gir bidrag for summen av avvik over tid og D-leddet gir bidrag for endringen i avviket når avviket oppstår.

I Niagara brukes det uavhengige forsterkninger til hvert enkelt ledd i PID loop action, Kp, Ki og Kd. I-tiden er satt til ett minutt og D-tiden er satt til ett sekund. I andre PID regulatorer blir ofte P brukt som felles forsterkning, og så har man I-tid og D-tid som parametere.

Hvis D-leddet skal legges inn bør D-tiden være lik ventilaktuator-tiden, noe vi ikke har tilgjengelig data på i systemet og det blir derfor vanskelig å utføre kalkuleringen på en god måte. I tillegg sier Skogstads metode at man skal bruke P og I for disse type prosesser. [4] Vi har likevel gjort et forsøk med å bruke Kd, uten å få de ønskede resultatene. Vi fikk ut at avvikene er like store før og etter. Dette er trolig på grunn av at ventilen som blir styrt har en egen regulering integrert i ventilen, slik at denne i praksis har en ukjent og variabel forsinkelse i forhold til ønsket verdi fra regulatoren.

Problemet er altså at man har tidsforsinkelse på responsen, derfor ønskes det en veldig treg regulering. Dette kan oppnås ved riktige innstillinger ut fra hvordan prosessen oppfører seg.

Tabellen under viser konsekvensene av å øke Kp, Ki og Kd:

Parametere	Stigningstid	Oversving	Innsvingningstid	Steady-state error	Stabilitet
Kp	Synker	Øker	Liten endring	Synker	Synker
Ki	Synker	Øker	Øker	Eliminerer	Synker
Kd	Liten endring	Synker	Synker	Ingen effekt	Øker (hvis lav)

Figur 4: Effekt av PID parametere [5]

I henhold til Ziegler-Nichols metode for innregulering, vil periodetiden ved oscillering være utslagsgivende for forholdet mellom Kp og Ki, Kd. Ku er Kp ved kritisk oscillering hvor kun P brukes. Tu er oscillerings-periodetiden.

Kontroll type	Kp	Ti	Td	Ki	Kd
P	0.50Ku	-	-	-	-
PI	0.45Ku	0.80Tu	-	0.54Ku/Tu	-
PID	0.60Ku	0.5Tu	0.125Tu	1.2Ku/Tu	3Ku*Tu/40

Figur 5: Ziegler-Nichols metode

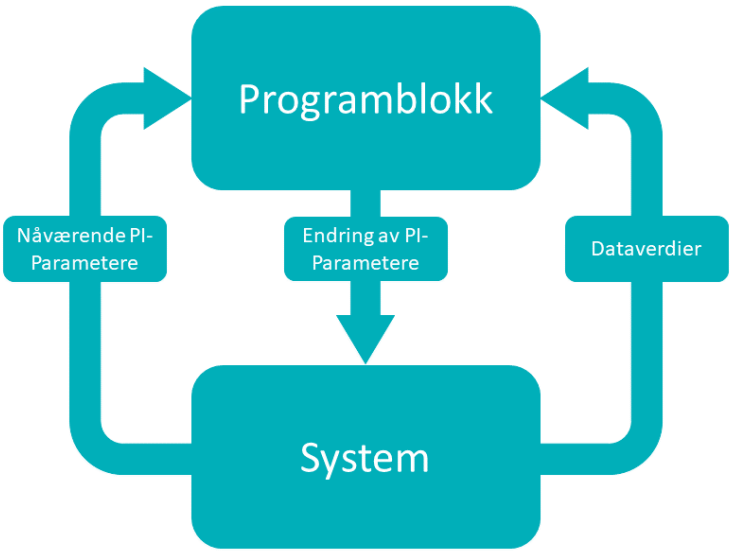
Ziegler-Nichols metode er i midlertidig ikke fornuftig å bruke da prosessene som skal reguleres avhenger av andre prosesser. Slik at å få prosessen som skal reguleres inn til å kritisk svinge ikke vil være praktisk gjennomførbart av et autonomt program som kun ser på enkelt reguleringer, og som er i drift. Metoden krever også at man endrer I-tid og D-tid, noe vi ikke har mulighet for å gjøre.

Skogestads metode er modulbasert og belager seg på prosessparameterne: prosessforsterkning, tidskonstant og tidsforsinkelse. [4] For å ikke forstyrre prosessen bruker vi topp-til-bunnpunkt-tid * 0,63 istedenfor sprangresponstest-tid fra 0-63%. Dette vil gi et godt utgangspunkt for reguleringen, bortsett fra at vi har en ukjent tidsforsinkelse i ventiler med egen regulering.

Denne ukjente tidsforsinkelsen gjør det vanskelig å sette gode parametere med en gang. Dette gjør det mer aktuelt å bruke en utforskende metode for å sette parameterne.

Løsningen vi har valgt å bruke er en utforskende metode hvor vi ganske enkelt senker Ki verdien. Fra Figur 4: Effekt av PID parametere, ser vi at dette vil øke stabiliteten. Ved nok iterasjoner av dette vil systemet bli stabilt, med mindre det viser seg å være feildimensjonert eller at komponenter er ødelagt. I så tilfelle vil ikke regulering løse problemet.

4.2 Kode



Figur 6: Illustrasjon av programmets funksjon

Koden tar inn CSV filer for måleverdier og settpunkt for en gitt PID regulering, den får også input for PID verdier. Hvor langt tilbake i tid som skal sjekkes, velges av bruker, og CSV filene som brukes inneholder data fra gjeldende tidspunkt og tilbake til oppgitt tidspunkt.

Måleverdiene med tidspunkt hentes ut og legges inn i egendefinert klasse kalt datapunkter, som igjen ligger i en liste av datapunkter. Når alle målepunktene er gitt vil datapunktene få et settpunk basert på tidspunktet som stemmer overens. Dette gjøres ved å sortere settpunktene opp mot måleverdiene ved hjelp av tidsstempelet.

Deretter vil tillatt avviksverdi for tillatt måleverdi regnes ut basert på prosentverdi oppgitt av bruker og settpunktet. Datapunktene sjekkes for om målepunktene er innenfor avviksintervallet og blir tilegnet et avviksattributt. Hvert punkt i listen med avviksattributtet vil bli sjekket opp mot punkter i listen som ligger innenfor et tidsintervall valgt av bruker. Her blir måleverdiene til punktene sjekket om de har gått ned til settpunktgrafen, og så videre ut fra settpunktet til en verdi som er utenfor igjen.

For å sørge for at pendling blir oppdaget, sjekkes det etter et gitt antall topper/bunner som går innom settpunktet i løpet av et gitt tidsrom. For å oppnå dette vil et datapunkt med måleverdi som er utenfor tillatt verdigrense sjekkes, både framover og bakover i tid, etter datapunkter som har måleverdi nær settpunktet. Når den finner et slikt punkt vil dette punktet brukes som utgangspunkt for å se etter punkter med måleverdier som er utenfor tillatt verdigrense. Når disse punktene blir funnet blir de lagret i en liste, og prosessen fortsetter fram til man har funnet det gitte antall topper/bunner. Da vil alle punktene i listen i tillegg til startpunktet bli kategorisert som pendling.

For å sikre oss at koden ikke itererer seg ut av listen når den går fra et punkt som er utenfor og sjekker dens nabopunkter for pendling, blir de første få punktene i listen satt til å være ‘ikke utenfor’. Når koden kjøres sjekker den også at den ikke er i nærheten av å være utenfor listen etter hver

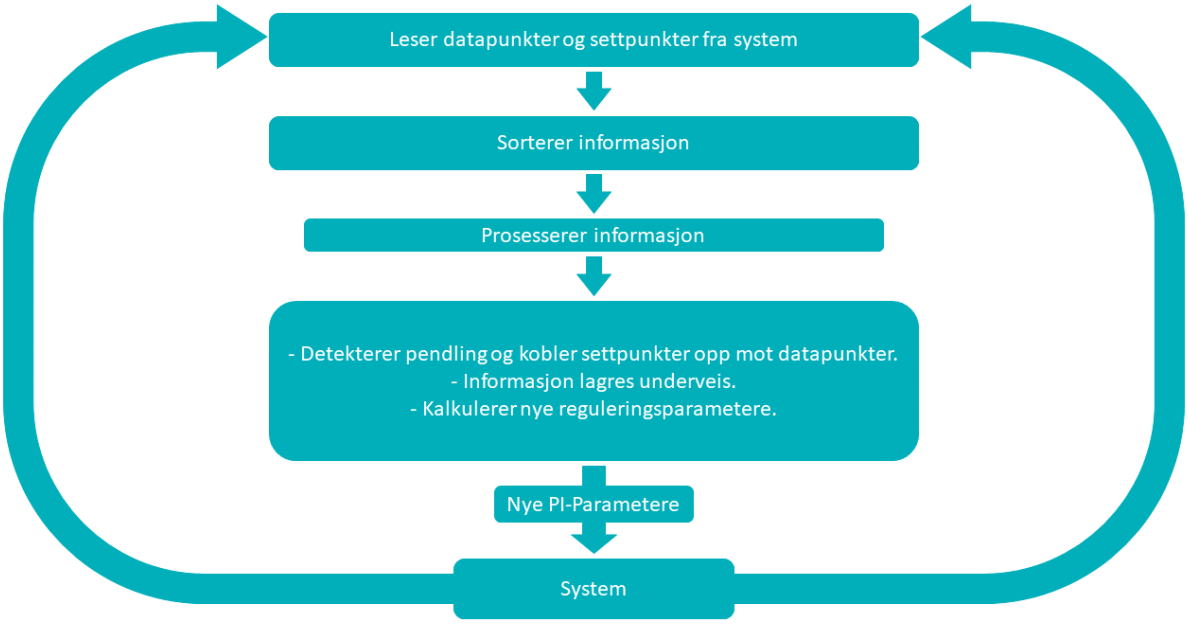
iterasjon. Også rett før listen er slutt vil iterasjonssjekken stoppe, for å ikke sjekke punkter som er utenfor listen.

Punkter som allerede er kategorisert som pendling vil ikke bli sjekket på nytt. Hvis et punkt som sjekkes for pendling ser en topp/bunn som er registrert som pendling vil punktet også registreres som pendling.

Når alle punktene har blitt sjekket blir disse vist i en tekstboks og i en graf, hvis det er pendling, slik at brukeren kan gjøre en vurdering av reguleringen. Samtidig blir det foreslått endring av PID verdier, som brukeren kan velge å sette. Dette er satt opp for at koden selv endrer parameterne når det over tid viser seg at det fungerer godt.

Ved implementering av koden fra C# til Niagara fikk vi en god del hjelp av Jan Gunnar hos GK. Dette gjaldt også ved oppretting av brukergrensesnittet. Det er likevel vi som bestemte hvordan det skulle se ut og hva som skulle være med. Grunnen til at vi fikk hjelp til dette er at vi ikke er kjent med programmeringsspråket, og det ville tatt for lang tid for oss å lære det.

Koden sjekker gjennom loggdata fra de siste x antall timene, og kommer med justeringsforslag dersom det er pendling. Siden det er trege prosesser vil det ikke være fornuftig å sjekke anlegget igjen de neste x antall timer, så programmet kjøres ikke oftere enn x antall timer. Per nå ligger dette inne som én gang i døgnet, men en bruker må likevel inn og akseptere justeringene.



Figur 7: Oversikt over stegene i prosessen

Kildekode ligger som Vedlegg 1.

4.3 Slitasje av utstyr ved feil regulering

4.3.1 Ventilasjonsanlegget

Før vi kan snakke om slitasje på maskindeler er det greit å forstå hvordan et ventilasjonsanlegg kan se ut, og være bygget opp. Et typisk anlegg henter frisk uteluft gjennom en inntaksrist før det så blir

ledet inn til ventilasjonsaggregatet. Her blir luften behandlet, enten om det er kjøling eller oppvarming. Et ventilasjonsaggregat er gjerne tilpasset hvert bygg den er installert hos, men kan bestå av blant annet varmegjenvinnere, filtre, vifter og spjeld. Via kanalveier blir så luften fordelt i bygget gjennom luftventiler. [6]

4.3.2 Hvilke komponenter er mest utsatt for uheldig regulering?

Ettersom det er snakk om slitasje som en konsekvens av en ujevn drift, skal vi kun ta for oss de mest utsatte komponentene. Etter å ha intervjuet tre personer, hvorav to arbeider eller har arbeidet som serviceingeniør for GK, ble vi fortalt at det var noen deler som skilte seg ut. Dette var helst komponenter som var utsatt for mye bevegelse eller start/stopp.

Dersom vi tar for oss et rørsystem med varmtvann, der målet er å varme opp luften som er kommet inn i bygget, så kan vi se hvordan aktuatorene aktivt tilpasser seg etter behovet. Trengs det mindre varmtvann strupes ventilene, og trengs det mer varme åpnes de opp. En slik aktuator, som kan styres ved hjelp av elektronikk, tilpasser seg som en følge av hvilke P, I og kanskje D verdier som er blitt satt i innreguleringen. Dersom disse parameterne ikke er gunstige kan det føre til at åpningsprosenten på en ventil ikke er stabil og aktuatoren sliter med å finne den rette posisjonen. Skjer denne alterneringen nokså hyppig, og over lang tid, kan det medføre større slitasje på aktuatorens deler. Deler som ble nevnt i intervjuet var blant annet pakninger og plasttannhjul. Ryker disse ender det som oftest opp i at hele komponenten blir byttet, og ved en uheldig regulering vil dette skje noe fortere enn hva dens forventede levetid tilsa.

4.3.3 Men hvorfor oppstår uheldig regulering?

Som nevnt over, kan gale parametere på PID-kontrolleren være en årsak for at systemet pendler, men hva skyldes det?

For det første så er et ventilasjonsanlegg et komplekst system med utallige faktorer som spiller inn. Uteluft, varmebruk eller antall personer i et bygg er eksempler på variable faktorer. Tar vi i tillegg med at ventilasjonsstyring er en relativ treg prosess og at selve ventilasjonsaggregatet med rørsystemer, ventiler og spjeld ikke lar seg enkelt beskrives av en simpel matematisk formel, så får vi et overblikk over hvor omfattende det kan være å regulere inn et system.

En annen faktor er den menneskelige faktoren. Personen som har hatt ansvaret for innreguleringen kan ha hatt for liten tid til å finne de rette innstillingene. Personen som har hatt ansvaret for innstalleringen av rørsystemet kan ha feildimensjonert. Eller personen som har hatt ansvaret for å bestille inn komponentene som skulle installeres kan ha bestilt deler som ikke var bra nok tilpasset systemet.

PID-parameterne blir derfor satt ut ifra erfaring og analyse, mens fintuningen blir gjerne bestemt ut ifra prøv og feil – metoden. Ettersom ventilasjonsstyring er en treg prosess med mye variasjon, kan et tilsynelatende bra innregulert system vise pendling ved et annet scenario.

4.3.4 Hvilke tiltak kan være med på å redusere slitasjen?

Dersom maskindeler ofte må byttes ut eller relativt hyppige besøk av servicepersonell forekommer, kan det tyde på at anlegget ikke driftes optimalt. Som nevn er uheldig regulering en årsak for at komponenter fortere blir slitt ut, men det er ikke alltid like lett å oppdage. Er det snakk om vifter eller pumper som svinger i ytelse kan man gjerne høre dette, er det dog snakk om små ventiler eller

spjeld rundt om i bygget kan det være lettere å overse problemet. Logging av anlegget er derfor helt nødvendig for å oppdage feil eller uregelmessigheter.

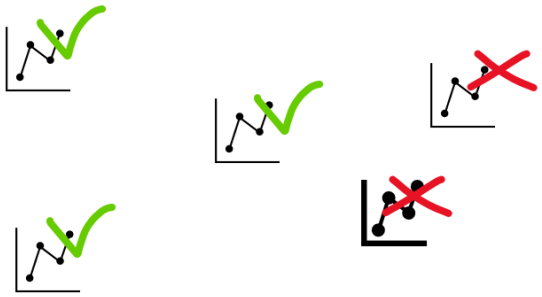
I tillegg til å sørge for at systemet har en jevn og stabil drift så er det viktig med gode vedlikeholdsrutiner og rapportering. Er det gjort ombygginger bør det også tas en ny innregulering for å sørge for at PID-parameterne er tilpasset en eventuell forandring av systemet.

4.4 Maskinlæring

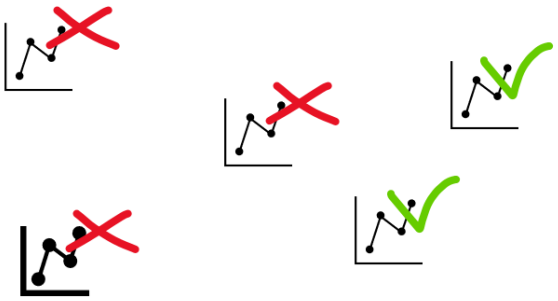
I VVS anlegg er det utrolig mange komponenter som jobber sammen. Når en får pendling så kan det være kun en eller flere av disse som er årsaken til problemet. Hva om det er den samme komponenten som igjen og igjen er synderen? Hvordan kan en finne ut hvilken komponent som gir størst utslag for pendling i systemet? Hva om en kan samle masse data og gi det til en maskin som så basert på denne dataen i fremtiden kan gjenkjenne hvilken komponent det er som skaper trøbbel. Dette er maskinlæring.

Det er slitsomt og tidkrevende for et menneske å skulle gå gjennom hundretalls med loggdata fra ulike anlegg og for å sjekke hvilke som inneholder uregelmessig regulering. En datamaskin derimot kan lese gjennom dem mye raskere. Derfor er maskinlæring så genialt. Maskinlæring går ut på å lære en maskin å tenke som et menneske. Ved å kunne kjenne igjen ting fra tidligere erfaringer og ta framtidige valg basert på disse.

Vi har laget et program som forteller om hvilke logger/komponenter som pendler og ikke. I maskinlæringsbiten må vi se på systemet som en helhet. Figurene under skal illustrere hva vi ønsker å finne ut av. Hvert grafsymbol representerer en komponent i systemet. Grønn hake betyr at komponenten er stabil og har en fin regulering. Rødt kryss betyr uheldig regulering.



Figur 9: Når den uthevede komponenten pendler ser vi at tre holder seg stabil mens en annen også pendler.



Figur 8: Her er det tre komponenter som pendler samtidig og to som ikke er påvirket.

Ved å sammenlikne mange ulike reguleringsscenarioer av det samme systemet kan vi finne ut hvilke komponenter som påvirker hverandre. Hvilke komponenter svinger samtidig og hvor mye svinger de. Er de bare litt ustabile eller blir en helt i utakt for eksempel hver gang utetemperaturen går under 5 grader. Algoritmen skal og se på hva energiforbruket er ved de ulike reguleringsscenarioene. Slik kan en finne ut om god regulering har lavere energiforbruk enn dårlig regulering.

4.4.1 Mulighet for å benytte maskinlæring

Er det mulig å lære en maskin dette? Da det er tidligere data vi trenger for å lære opp maskinlæringsalgoritmen er det viktig å finne ut om den dataen vi har er tilstrekkelig. Vi vet at GK har rikelig med loggdata fra anleggene sine så mengdedata vil ikke være et problem. GK har automatisk

logging av verdier hvert tiende minutt, men har også slik at de kan logge temperaturverdier hver gang de stiger eller synker med 0.1 grader celsius. Det er derfor god dekning på loggene. Det som kan påvirke troverdigheten på loggene har med hvor i rommet ting står plassert. Er for eksempel temperaturmåler og tilluftsventil rett ved siden av hverandre kontra langt unna hverandre vil en få forskjellige verdier på temperaturmåleren.

Et problem med loggene er at ikke alle verdier blir målt alltid. Det begynner å bli mer vanlig å montere energimålere på forskjellige kretser, som viser hvilke vannmengder man har til hvilke tidspunkter, men det er langt fra alle som gjør det. Det kan dermed være noen tilfeller en ikke har nok data til å implementere en algoritme godt nok.

GK har veldig mange forskjellige typer systemer. Det kan derfor være en utfordring om vi kun trener algoritmen opp på en type system. Det kan godt være man bør vurdere å ha flere algoritmer for å dekke variasjonene av anlegg. For eksempel en maskinlæringsalgoritme som kunne oppdaget en sammenheng mellom pendling på temperatur og vanngjennomstrømning på kursene i bygg hvor vi har data på dette. Mens vi andre steder kan se sammenhengen mellom andre ting.

Det er mye informasjon som må følge med hver logg. For eksempel hvor stort rom og hvor mange etasjer komponenten betjener. Eller hvor mange det er av samme komponent i rommet, for eksempel varmeovner. En må se på komponenten i forhold til hele systemet den er med i. Her er det viktig at vi ikke får for mye eller for lite informasjon på hver logg. Er der for lite informasjon kan logaritmen slite med å finne noe sammenheng mellom loggene. Er det derimot for mye informasjon kan det forvirre logaritmen slik at den finner sammenhenger som ikke er relevant for oss. For eksempel kan den finne ut at en varmeovn utgjør størst pendling på oddetallsdager.

4.4.2 Typer maskinlæring

Det finnes mange måter å lære maskiner på. For det første har vi overvåket og ikke-overvåket læring. Overvåket går ut på å vise maskinen eksempler som er markert med det vi vil den skal finne. På den måten vil det bli lettere og gå fortere for maskinen å lære seg å detektere det den skal. I motsetning til ikke-overvåket læring hvor maskinen selv fra starten av må finne ut hva som skiller seg ut i eksemplene [7]. I denne læreprosessen bruker vi overvåket læring da vi har mye informasjon om loggene på forhånd og kan vise maskinen gode eksempler på hva den skal se etter.

Videre kan vi skille mellom aktiv og passiv læring. Ved passiv læring vil maskinen bare kunne observere informasjonen den får og så lære ut ifra det. Ved aktiv læring kan den stille spørsmål underveis eller eksperimentere. Med å bruke aktiv læring kan maskinen vise oss hva den har kommet fram til [8]. Slik at vi kan hjelpe maskinen å fortelle om det den gjør er riktig eller feil. Maskinen kan finne ut hvilke komponent som har størst innvirkning på systemet også kan vi analysere manuelt og verifisere at den har gjort riktig. Slik får vi en sikrere algoritme.

Vi kan velge å lære maskinen offline eller online. Offline vil den få store mengder treningsdata og lære seg opp på disse. Da vil den i framtiden bare basere seg på treningsdataen fra den gangen og ikke utvikle seg noe mer. For å oppdatere en slik algoritme må man igjen gå offline og gi den et nytt treningssett [9]. Det negative med denne metoden er at den krever mye databehandlingsressurser og det tar lang tid å gå gjennom all dataen. I GK får man kontinuerlig ny loggdata som vil variere gjennom hele året og det vil være krevende å skulle oppdatere algoritmen ofte nok for å håndtere disse.

Det ideelle hadde vært om maskinen kunne lære samtidig som den får mer og mer informasjon levert. Det produseres konstant ny loggdata fra systemene til GK og vi ønsker at algoritmen alltid skal være oppdatert på disse. Dette kalles online læring. Når den har lært fra tidligere loggdata kan den kaste den og ta imot ny. Slik vil algoritmen hele tiden lære seg nye systemer og forbedre seg selv.

Når det er satt opp en algoritme og modell må man teste og vurdere hvor god den er. Det er viktig å ha et godt og variert treningssett og det samme gjelder testsettet. Treningssettet er for å trene opp algoritmen mens testsettet er for å teste hvor god den er. Man kan så sette opp en funksjon som samler opp antall feil algoritmen gjør. Det vil si antall systemer der den ikke har funnet riktig komponent som gir størst innvirkning på systemet. Får en for høyt antall feil kan en vurdere teste ut en annen modell og/eller algoritme.

4.4.3 Utvidelser

På sikt kan man utvikle algoritmen til å ta hensyn til flere faktorer for å optimalisere systemene best mulig. For eksempel ved å få maskinen til å analysere værmeldinger, målte temperaturer, strømpriser og bruksmønstre i bygg kan man bruke alle disse til å beregne når en bør bruke energi. Da kan man forutse når det trengs varme eller kulde i bygget og bruke mest energi mens strømprisen er lav. Dette vil være med å senke energiforbruk og -kostnader.

Det vil ikke bare komme de individuelle byggene til gode å bruke strøm mens prisen er lav. Det er jo slik at strømmen er høy når mange bruker strøm. Dersom færre bruker strøm når det er stort forbruk vil det være mindre behov for å bygge ut infrastruktur rundt strøm, noe som kommer samfunnet til gode.

Maskinlæring er et spennende område og det bringer mange goder. Det blir spennende å se utviklingen av det innen varme- og ventilasjonsanlegg.

5 Testing

5.1 Forarbeid

Før vi var ferdig med første utkast var vi ute på anlegg med Jan Gunnar fra GK, for å se på hvordan regulering av anleggene fungerer i praksis og på typiske anlegg GK kan jobbe på. Vi fikk se på komponenter som typisk er i bruk og lært litt om hvordan de fungerer, og hvordan det ser ut når man skal endre parametere på en reguleringssløyfe. Det var lett å se hvordan endringer i verdier kunne føre til at anlegget som var stabilt ble ustabilt når vi endret på P og I parameterne som var stilt inn. Det gikk fra å være en jevn og kontrollert kurve, til å bli en kurve som gikk veldig mye opp og ned. Fordi det er trege prosesser så tok det likevel en god stund før man kunne konstatere endringer i prosessen ut ifra endringer gjort på parameterne, så det er enkelt å tenke seg til hvorfor GK ønsker programmet vi lager i denne oppgaven.

5.2 Loggdata

Under utviklingen av programmet brukte vi ulike loggdata som kom fra anlegg med pendling på, og anlegg som fungerte fint. Ved å bruke anlegg med og uten pendling kunne vi enkelt teste om programmet fanget opp det som skulle fanges opp og presentere det slik vi ønsket. Under denne prosessen arbeidet vi i VS og måtte importere loggfiler manuelt. Det var også har vi fant ut hvordan koden måtte se ut for å mest effektivt og korrekt jobbe seg gjennom loggene.

5.3 Niagara

Når vi hadde fått koden inn i Niagara kunne vi hente loggdata direkte fra skylagring og PID parameterne fra anleggene de befant seg i. det var noe vanskeligere å jobbe med syntaksen da vi ikke var kjent med denne fra før av fordi det er et annet programmeringsspråk. Her fikk vi testet at koden kunne lese av loggdataen automatisk og komme med forslag til oppdaterte PID innstillinger dersom det ble oppdaget pendling i perioden som ble lagt inn, vanligvis de siste 24 timene.

6 Diskusjon

6.1 Tidsplan

Ved innlevering av forstudiet satte vi opp hovedemner i tidsplanen etter det vi først så på som viktig i oppgaven. Ikke bare skulle vi lage en C#-kode men også tilegne oss kunnskap rundt ulike temaer. Kunnskap vi trengte til å kunne lage selve koden og kunnskap til ulike problemstillinger vi skulle svare på i oppgaven.

Vi har arbeidet jevnt hele veien, men kan se i ettertid at vi har jobbet i moduler. Første vi gjorde ble en rask undersøkelse rundt ventilasjonssystemer og GK sine systemer slik at vi kunne ta fatt i kodearbeidet. Videre satte vi opp hovedkonturene av hvordan vi ville koden skulle være. Da denne var på plass kunne vi begynne på neste modul som var å lage førsteutkast av koden. Opprinnelig så vi for oss at vi skulle bli ferdig med denne til påske, men denne var vi allerede begynt å implementere i GK sitt system uken før påske.

Da vi hadde implementert førsteutkastet og i tillegg testet at den fungerte godt hos GK, tok vi fatt på andreutkastet. Denne ble vi raskt ferdig med etter god hjelp fra veileder. De andre punktene i oppgaven om teori har vi jobbet med jevnt samtidig med kodearbeidet. Vi hadde allerede startet å skrive på rapporten i god tid og følte ikke et stort press mot slutten.

Hovedforskjellen fra tidsplanen levert ved forstudie og slik den ser ut i dag er punkt 7 om feil regulering og slitasje. Vi begynte i starten av perioden å søke litt rundt etter artikler om temaet, men vi fikk også høre mye om emnet fra Jan når vi var ute på anlegg. Da fikk vi en ide om at vi kunne intervjue folk som jobber med maskiner og som har sett med egne øyne hvordan slitasje på utstyr kan skje ved feil regulering.

Disse intervjuene skjedde etter planen. Maskinlæringsbiten har vært litt krevende å sette seg inn i da vi hadde bare liten kjennskap til det fra før av. Men denne også ble vi ferdig med til slutt. Vi er fornøyde med hvordan vi har klart å følge plan og tidsfrister.

6.2 Risikoanalyse

I risikoanalysen hadde vi satt opp ulike risikoer med grad av sannsynlighet og konsekvens. Den eneste med høy sannsynlighet var feilkoding. Vi hadde litt feilkoding underveis, men det ble løst.

Videre hadde vi satt moderat sannsynlighet på risikoen for at medlemmer skulle bli smittet av korona eller bli satt i karantene i løpet av perioden. Dette kunne hatt stor konsekvens for arbeidet, men vi har heldigvis holdt oss unna begge deler alle sammen. Uønsket endring/sletting av viktig informasjon i GK sitt system hadde også fått graden moderat i sannsynlighet. Heldigvis har vi unngått dette med tanke på de store konsekvensene som kunne komt av det.

Risikoer med lav sannsynlighet er tap av informasjon/filer og uenigheter/splittelse innad i gruppen. Dette har ikke hendt. Alt i alt har vi ikke støtt på noen store utfordringer eller problemer.

7 Konklusjon

På bakgrunn av analyse av loggdata og regelmessig testing av programkode, samt intervjuer og god diskusjon med vår veileder fra GK, oppsummeres videre viktige resultater og funn fra oppgavens forløp.

7.1 Nødvendigheten ved loggføring av anleggsdrift

Ettersom GK lenge har loggført anleggsdriften sin har de hatt tilgang til å oppdage uheldige reguleringer ved å studere grafer. Grafene viser tydelig når et anlegg ikke kjøres optimalt og gir så et bilde på at en endring bør gjøres. Tidlig i prosessen fikk vi også et innblikk i disse grafene og det førte videre til en tankeprosess hos oss.

Da vi så startet med planleggingen av hvordan oppgaven skulle løses, ble vi fort enige om å basere oss på loggdata. Ettersom oppgaven gikk ut på å opprette en algoritme som skulle detektere og håndtere pendling, var dette et naturlig sted for oss å begynne. Vi fikk så tilgang til GK sine systemer via Jan Gunnar Ludvigsen, og med det fikk vi også tilgang til loggfiler fra samtlige av GK sine anlegg. Ut ifra disse loggfilene kunne vi analysere anleggsdriften ved å studere grafer. Noen anlegg hadde stabile og jevne grafer med lite svingninger, mens andre var mer utsatt for pendling. Hver graf var representert av en rekke datapunkter med et tidsstempel og en verdi. Hver graf var også rettet mot en spesifikk føler, enten om det var en temperatursensor eller åpningsprosenten på et spjeld. Ved å få tilgang til denne dataen som en CSV-fil, gjorde det at vi kunne starte arbeidet med en logisk analyse via databehandling.

Dermed kan vi si at det å føre logg fra et anlegg vil kunne lede til en mer komplett analyse av driften og dens komponenter. Et slikt innblikk vil også kunne redusere slitasjeomfanget til maskindeler ved at det kan oppdages før en komponent potensielt ryker.

7.2 Detektering av pendling

Etter å ha fått tak i de nødvendige loggfilene vi trengte for å utføre en analyse, var det så å få en datamaskin til å forstå hva den skulle se etter. Som nevnt tidligere brukte vi Visual Studios og programmeringsspråket C# til å lage logikken til programmet vårt. Her skrev vi altså koden før den, ved hjelp av Jan Gunnar, ble implementert hos GK.

Der vi, etter å ha fått en innføring av Jan Gunnar, kunne se og oppdage pendling måtte vi finne ut av hvordan programmet skulle detektere det. Vi diskuterte flere metoder, men det som gikk igjen var å registrere verdier som skilte seg ut. Noe som er viktig å få med er at vi i tillegg til loggfiler med datapunkter (sensortype, verdier og tid), hentet vi også loggfiler med settpunkter. Disse settpunktene ble bestemt ut ifra forhold vi ikke hadde tilgang til, men som påvirket anleggets drift. Ut ifra en kombinasjon av settpunkter og registrerte datapunkter kunne vi danne et forhold mellom de to ulike informasjonene. Settpunktet ble brukt som et referansepunkt mens datapunktet viste hvilke verdier som faktisk ble registrert. Vi satt så en grense på hvor stor differanse det kunne være mellom settpunkt og verdi, og fikk skrevet en kode der programmet skulle lagre punktet og dens verdier.

En tilleggsmetode vi benyttet oss av var å registrere alterneringer av grafen. Programmet detekterte altså når en verdi gikk for lavt og for høyt med viss frekvens. Dette kan tyde på at en komponent ikke finner den riktige innstillingen og derfor pendler.

Selv om vi er fornøyde med våre kriterier av hva som skal kategoriseres som pendling eller uheldig regulering, så er det med stor sannsynlighet at det finnes bedre metoder. Programmet vårt kjører en analyse med flere tusen datapunkter på kort tid, og det registrerer hva vi mener bør telles som pendling. Det den kanskje ikke registrerer like enkelt er uheldig regulering som ikke blir vist som pendling. Det kan for eksempel være unødvendig høyt pådrag av luft der temperaturen og luftmengden hadde vært tilstrekkelig ellers.

7.3 Bestemmelse av P og I parametere

Detektering av pendling var en omfattende prosess, men det var likevel ikke hele oppgaven til programmet. I tillegg til å finne ut hvor og når et anlegg hadde uregelmessigheter i driften, var det også et mål å kunne foreslå endringer av P og I parameterne. Programmet skulle altså overta arbeidet med å finne de rette innstillingene til hver komponent. Etter samtaler med Jan Gunnar har vi forstått det slik at innreguleringer skjer via prøv og feil metoden, helt til man finner et nivå man er fornøyd med. Selvfølgelig ligger det mer bak, men her er det snakk om den siste fininnstillingen. Det er også det som vil bli programmets funksjon, rette på en allerede innstilt parameter.

Programmet er per i dag ferdig implementert hos GK hvor den ligger inne som en funksjonsblokk sammen med relevante reguleringsblokker. Den mottar informasjon direkte fra systemet og via et brukergrensesnitt kan man se nåværende parametere, samt foreslåtte endringer. Det er så opp til det tekniske personalet å godta disse endringene for å iverksette dem.

7.4 Videre anbefalinger

Etter å ha arbeidet med denne oppgaven har vi fått et større innblikk i hvordan regulering av spesielt ventilasjonsanlegg fungerer. Begrepet *pendling* har blitt et hverdagsord for oss, samt forståelsen for hva konsekvensene av uheldig regulering er. For å kunne anbefale en videre forbedring av hva vi har oppnådd så er det første som kommer til sinns, maskinlæring. Dessverre har vi lite kunnskap om emnet, men vi er alle enige om at det kunne ha vært en større del av utdanningsforløpet vårt.

7.4.1 Forbedring av kriterier for pendling, basert på loggdata

Som nevnt under 7.3 vil programmet kunne bli forbedret dersom man hadde kategorisert pendling på en mer avansert måte. Dette kan være et fint utgangspunkt hvis man har tenkt å fordype seg i regulering av ventilasjonsanlegg. Er all pendling uheldig regulering og hva kan regnes som uheldig regulering? Er det mulig å forutse pendling og forhindre at det skjer, og kan i så fall maskinlæring være en mulighet?

7.4.2 Forbedring av P og I parametere

Under 7.3 blir fininnstillingen av reguleringsparameterne nevnt. Det blir nå i hovedsak gjort, basert på erfaring, på en prøv og feil metode. En god maskinlæringsalgoritme vil her ha potensiale til å lære seg hvilke komponenter og segmenter som krever visse innstillinger.

7.4.3 Kartlegging av hvilke komponenter som utgjør størst påvirkning på pendling.

Hvis man oppdager pendling i et ventilasjonsanlegg og ved hjelp av loggdata klarer å detektere hvilke komponenter som forårsaker det, vil det være naturlig å endre på de tilhørende parameterne. Hvis det så viser seg at endringen ikke løste problemet, kan det da tyde på at komponenten ikke hadde så stor innvirkning på resten av systemet? Kan en enkelt komponents feilinnstilling medføre følgefeil til andre komponenter, og hvordan kan man detektere denne komponenten eller segmentet av systemet?

8 Referanser

[1] «gk.no,» GK AS, [Internett]. Available: https://www.gk.no/. [Funnet 24 01 21].

[2] S. C. Sugarman, HVAC Fundamentals (3rd Edition), Fairmont Press, Inc., 2016.

[3] ASHRAE, «1. HVAC System Analysis and Selection page,» i *2020 ASHRAE Handbook - HVAC Systems and Equipment (SI Edition)* , American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc. (ASHRAE), 2020, pp. 1.1-1.2.

[4] F. Haugen, Reguleringsteknikk, techteach.no/shop, 2012.

[5] K. H. A. G. C. Yun Li, «PID control system analysis and design,» *IEEE Control System Magazin,* Februar 2006.

[6] N. C, Kompendium i Ventilasjonsteknikk, UIT, 2017.

[7] M. Mohri, A. Rostamizadeh og A. Talwalkar, Foundations of Machine Learning, second edition, MIT Press, 2018.

[8] S. Shalev-Shwartz og S. Ben-David, Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press, 2014.

[9] A. Géron, Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow (secodn edition), Sebastopol, CA 95472: O'Reilly Media Inc., 2019.

Appendiks A Forkortelser og ordforklaringer

PID	Proporsjonal, integral og derivasjon. (Reguleringsparametere)
VVS	Varme-, ventilasjons- og sanitærteknikk
CSV	comma-separated values
VS	Visual Studio

Appendiks B Prosjektledelse og styring

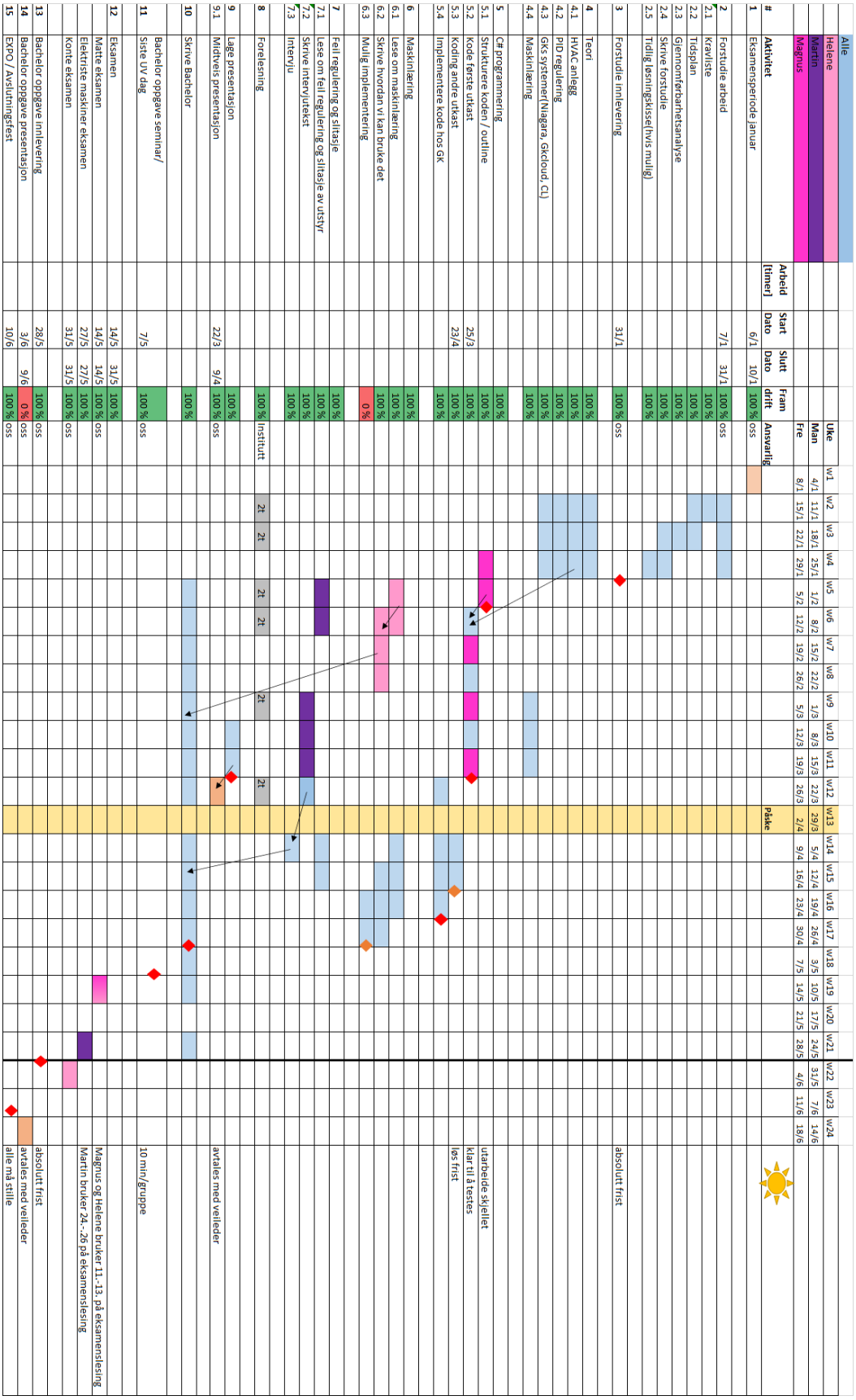
B.1 Prosjektorganisasjon

Helene Pisani er prosjektleder. Hovedansvar for å sørge for å holde frister og avtale møter med veiledere.

Magnus Kristiansen har hovedansvar for utarbeiding av koden og fordeler arbeidsoppgaver knyttet til den.

Martin Jacobsen har hovedansvar for å skrive logg fra møter og lage grafiske fremstillinger til oppgaven.

B.2 Fremdriftsplan



B.3 Risikoliste

Risikovurdering		Sannsynlighet				
		Svært lav	Lav	Moderat	Høy	Svært høy
Konsekvens	Svært stor	5	6	7	8	9
	Stor	4	5	6	7	8
	Moderat	3	4	5	6	7
	Liten	2	3	4	5	6
	Svært liten	1	2	3	4	5

#	Risikoer	Risikovurdering			Mulige tiltak
		Konsekvens	Sannsynlighet	Risikonivå	
1	Gruppemedlemmer blir utsatt for Koronasmitte.	Stor	Moderat	6	Digitale møter.
2	Uønsket endring/sletting av viktig informasjon i GK sitt system.	Svært stor	Moderat	7	Sikkerhetskopiering i forkant og underveis.
3	Feilkoding.	Svært liten	Høy	4	Regelmessig testing av kodesegmenter.
4	Tap av filer og informasjon.	Svært stor	Lav	6	Skyløsninger og i tillegg sikkerhetskopiering på egen datamaskin.
5	Uenigheter/Splittelse innad i gruppen.	Stor	Lav	5	Gode samtaler og jevn arbeidsfordeling.

9 Vedlegg

1. **Kode**

2 Intervjumappe

3 Prosjektdagbok