Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for teknisk kybernetikk

Håvard Bonde
Amund Grasbakken
Magnus Grøterud
Kjetil Halse

# PLS som autopanel for dieselaggregat i autonomt strømforsyningsanlegg

Bacheloroppgave i Automatisering og robotikk
Veileder: Cevdet Islek
Mai 2022

NTNU
Kunnskap for en bedre verden

Håvard Bonde
Amund Grasbakken
Magnus Grøterud
Kjetil Halse

# PLS som autopanel for dieselaggregat i autonomt strømforsyningsanlegg

Bacheloroppgave i Automatisering og robotikk
Veileder: Cevdet Islek
Mai 2022

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for teknisk kybernetikk

NTNU
Kunnskap for en bedre verden

| | | |
|---|---|---|
| **Oppgavetittel (norsk og engelsk):**<br>PLS som autopanel for dieselaggregat i autonomt strømforsyningsanlegg<br>PLC as an autopanel for a diesel generator in an autonomous power supply | | |
| **Forfattere:**<br>Bonde, Håvard **email: haavabon@stud.ntnu.no**<br>Grøterud, Magnus **email: magngrot@stud.ntnu.no**<br>Halse, Kjetil **email: kjethals@stud.ntnu.no**<br>Sæther Grasbakken, Amund **email: amundsg@stud.ntnu.no** | | |
| **Prosjektnummer:**<br>E2210 | **Innleveringsdato:**<br>20.05.22 | **Gradering:**<br>Åpen |
| **Studium:**<br>Bachelor i ingeniørfag, elektro | | |
| **Studieretning:**<br>Automatisering og robotikk | | |
| **Veileder internt:**<br>Cevdet Islek<br>Universitetslektor ved NTNU<br>**Institutt:**<br>Institutt for teknisk kybernetikk | | |
| **Oppdragsgiver:**<br>Getek AS<br>**Kontaktperson:**<br>Helge Engebø<br>Faglig ansvarlig / Installatør<br><br>Arnt Lysø<br>Prosjektleder | Wago Norge<br><br>Tor Erik Næbb<br>Forretningsutvikler Automasjon<br><br>Jim Roger Kvello<br>Distriktssjef Midt & Nord | |
| **Stikkord norsk:**<br>PLS, dieselaggregat, automasjon, programmering, brukergrensesnitt | **Stikkord engelsk:**<br>PLC, diesel generator, automation, programming, user interface | |

# Forord

Denne rapporten er skrevet av fire elektroingeniørstudenter innenfor studieretning automatisering og robotikk, og tilsvarer sluttresultatet på vårt 3-årige bachelorløp ved Norges Teknisk- og Naturvitenskapelige Universitet. Oppgaven er gjennomført av undertegnede i samarbeid med Getek AS og Wago Norge, og veiledet av universitetslektor Cevdet Islek.

Denne rapporten henvender seg i hovedsak til NTNU, Getek AS og Wago Norge. Andre parter er også velkommen til å lese rapporten hvis det er av interesse.

Vi vil sende en spesiell takk til Helge Engebø fra Getek AS for gode tilbakemeldinger, veiledning, og et godt samarbeid gjennom hele prosjektperioden. Vi vil også takke Jim Roger Kvello og Tor Erik Næbb fra Wago Norge, og Arnt Lysø fra Getek for veiledning, utlevering av maskinvare og lisenser til programvare. Vi ønsker i tillegg å takke vår veileder Cevdet Islek for veiledning og raske svar på spørsmål, og Wago Support for gode svar på spørsmål angående deres produkter.

Rapporten er strukturert slik at det er en fordel å lese kapitlene i kronologisk rekkefølge, men hovedinnholdet er i all hovedsak nevnt i kapittel 4: Systemstruktur. PDF-versjonen av rapporten inneholder en klikkbar innholdsfortegnelse, henvisninger og kilderefereringer i dokumentet, slik at det skal være lett å finne frem i dokumentet.

Håvard Bonde

Amund Grasbakken

Magnus Grøterud

Kjetil Halse

Trondheim, 20.05.2022

Sted/dato

# Sammendrag

Oppdragsgiver Getek AS har i samarbeid med Wago Norge, laget en oppgave som går ut på å erstatte styresystemet i Getek sine autonome strømforsyningsanlegg (APS). Dette gjennomføres ved bruk av Wago sin program- og maskinvare.

Det har i prosjektet blitt utviklet et system på en av Wago sine programmerbare kontrollere for overvåking og styring av dieselaggregat i APS-enheter. Det er utviklet et brukergrensesnitt for aggregatet som brukes på en trykkfølsom skjerm på anlegget. Styringen av aggregatet bygger hovedsakelig på en sekvens som er styrt ut ifra et kjøresignal. Sekvensen starter dieselaggregatet automatisk, iverksetter lading av en batteribank, og stopper aggregatet i henhold til kjøresignalet. Sekvensen er tilgjengelig for overvåking i brukergrensesnittet.

Det er laget et alarmsystem som i interaksjon med sekvensprogrammet, detekterer alarmer og utfører nødvendige aksjoner. I tillegg til varsling er det implementert to typer alarmaksjoner når en alarmsituasjon oppstår: vanlig eller umiddelbar stopp av aggregat. Brukergrensesnittet og et SMS-system vil gi varsler når alarmer og feilsituasjoner oppdages, og informerer om feilen. Det er mulig å kvittere aktive alarmer ved hjelp av både SMS og brukergrensesnitt.

Brukergrensesnittet er i utgangspunktet laget for bruk på små berøringsskjermer. Det er programmert på en modulær måte og inneholder egne vinduer tilknyttet motor, generator, alarm, SMS og diverse innstillinger. Det er laget en toppmeny som alltid vises, hvor man derifra kan navigere videre til ønsket vindu. Brukergrensesnittet inneholder mange innstillinger, slik at programmet lettere kan tilpasses ulike APS-anlegg.

SMS-systemet i kontrolleren består av to hoveddeler: sending og mottakelse av SMS. Gruppen har utviklet en kommunikasjonsprotokoll slik at brukere kan sende kommandoer via SMS inn til systemet. Dette kan blant annet være kvittering av alarmer og forespørsel om statusmelding. SMS-er inn til systemet verifiseres ved sjekk i en intern kontaktlogg, slik at ikke hvem som helst kan utføre aksjoner. Varsling av alarmer sendes automatisk til numrene som er lagret i denne kontaktloggen. Alarmvarslingen er utstyrt med et filter slik at ulike kontakter kan få ulike varslinger.

Gjennom prosjektet har gruppen i tillegg fått et godt innblikk innen prosjektstyring og samarbeid. Gruppen sitt produkt er testet på fysisk APS-anlegg og overleveres som et komplett system i henhold til oppgavebeskrivelsen.

# Innhold

**Forord** iii

**Sammendrag** v

**1 Innledning** 1
1.1 Introduksjon . . . 1
1.2 Oppdragsgiver . . . 1
1.3 Bakgrunn . . . 1
1.4 Problemstilling . . . 2
1.5 Avgrensninger . . . 2
1.6 Terminologi . . . 3
1.7 Rapportens oppbygging . . . 4

**2 Metode og utstyr** 5
2.1 Forskrifter og standarder . . . 5
2.1.1 NEK400 . . . 5
2.1.2 IEC 61131-3 Standard . . . 5
2.1.3 Maskindirektivet . . . 5
2.2 Programvare . . . 6
2.2.1 e!COCKPIT . . . 6
2.2.2 PC Schematic . . . 6
2.2.3 LaTeX/ Overleaf . . . 6
2.2.4 Excel . . . 6
2.3 Maskinvare . . . 7
2.3.1 PFC200 (750 - 8207) . . . 7
2.3.2 Strømforsyning (787 - 1212) . . . 7
2.3.3 DI - Digital Inngangsmodul (750 - 430) . . . 8
2.3.4 DO - Digital Utgangsmodul (750 - 530) . . . 8
2.3.5 AI - Analog Inngangsmodul (750 - 459) . . . 9
2.3.6 RTD - Motstandsmåler (750 - 463) . . . 9
2.3.7 Energimåler (750 - 494) . . . 9
2.3.8 Endemodul (750 - 600) . . . 10
2.3.9 HMI-skjerm - TP600 (762-4303) . . . 10
2.3.10 Releer . . . 10

**3 Teori** 11
3.1 Programmerbar Logisk Styring (PLS) . . . 11
3.2 Autopanel . . . 11
3.3 Dieselaggregat . . . 12
3.4 Autonom strømforsyning - APS . . . 13
3.5 PLS-programmering . . . 14
3.5.1 Programmeringsspråk . . . 14
3.5.1.1 Strukturert Tekst (ST) . . . 14
3.5.1.2 Sekvensprogrammering (SFC) . . . 14
3.5.1.3 Ladderdiagram (LD) . . . 14

3.5.1.4 Funksjonsblokkdiagram (FBD) . . . 15
3.5.1.5 Continous Function Chart (CFC) . . . 15
3.5.2 Deklarering av variabler . . . 15
3.5.2.1 Datatyper . . . 16
3.5.2.2 Remanente variabler . . . 16
3.5.3 Pekere . . . 17
3.5.4 Alarmkonfigurator . . . 17
3.5.4.1 Alarmklasser . . . 18
3.5.4.2 Alarmgrupper . . . 18
3.6 Lineær interpolasjon . . . 19

**4 Systemstruktur 20**
4.1 Autopanel som skal byttes ut . . . 20
4.2 Generell oppbygging . . . 20
4.2.1 Oppgavebehandling i PLS . . . 20
4.2.2 Programmering . . . 22
4.2.2.1 Variabelnavn . . . 22
4.2.2.2 Variabelliste . . . 23
4.2.3 Inn- og utgangssignaler . . . 24
4.2.4 Diverse programlogikk . . . 25
4.2.4.1 Driftstimer . . . 25
4.2.4.2 Glødetid . . . 25
4.2.4.3 Temperaturoverstyring . . . 25
4.2.4.4 Valg og skalering av analoge innganger . . . 26
4.3 Aggregat-sekvens . . . 27
4.3.1 Kjøreflagg . . . 28
4.3.2 Oppstart motor . . . 29
4.3.3 Nytt startforsøk . . . 29
4.3.4 Innkobling av lastkontaktor . . . 30
4.3.5 Stopp av motor . . . 30
4.3.6 Nødstopp . . . 31
4.4 Alarmsystem . . . 32
4.4.1 Struktur . . . 32
4.4.2 Alarmklasser . . . 33
4.4.3 Alarmgrupper . . . 33
4.4.4 Alarmobjekt . . . 34
4.4.5 Funksjonsblokker . . . 35
4.4.5.1 FB: Digital alarmdeteksjon . . . 35
4.4.5.2 FB: Analog alarmdeteksjon . . . 36
4.4.5.3 FB: Kommunikasjonsfeil . . . 36
4.4.6 Alarmdeteksjon . . . 37
4.4.6.1 Digital alarmdeteksjon . . . 37
4.4.6.2 Analog alarmdeteksjon . . . 37
4.4.6.3 Kommunikasjonsfeil . . . 38
4.4.6.4 Strømbrudd . . . 38
4.4.6.5 Start- og stoppfeil . . . 39
4.4.7 Alarmlampe . . . 39
4.4.8 Kvittering av alarmer . . . 40

4.4.8.1 Kvitter valgt alarm . . . 40
4.4.8.2 Kvitter alle alarmer . . . 41
4.4.8.3 Nullstill kvittering . . . 41
4.5 SMS-system . . . 42
4.5.1 Struktur . . . 43
4.5.2 SMS: Hovedsekvens . . . 44
4.5.3 Håndtering av innkommende SMS . . . 45
4.5.4 Håndtering av utgående SMS . . . 46
4.5.5 Kommunikasjonsprotokoll . . . 48
4.6 Brukergrensesnitt - HMI . . . 50
4.6.1 Hierarki . . . 50
4.6.2 Modulært design . . . 51
4.6.3 Fargetema . . . 52
4.6.4 Toppmeny . . . 53
4.6.5 Hamburgermeny . . . 54
4.6.6 Pop-up: Logg Inn, Tastatur og Talltastatur . . . 55
4.6.7 Vinduer . . . 56
4.6.7.1 Oversikt . . . 56
4.6.7.2 Motor . . . 57
4.6.7.3 Generator . . . 58
4.6.7.4 Motor Innstillinger . . . 59
4.6.7.5 Generator Innstillinger . . . 60
4.6.7.6 Alarm . . . 61
4.6.7.7 Alarmtest . . . 62
4.6.7.8 SMS Kontakter . . . 62
4.6.7.9 SMS Innstillinger . . . 64
4.6.7.10 Innstillinger . . . 65
4.6.7.11 Avanserte Innstillinger . . . 66
4.7 Batterilading vha. værprognose . . . 69
4.7.1 Mappestruktur . . . 70
4.7.2 Pythonapplikasjonens GUI . . . 71
4.7.3 Værprognose i PLS-en . . . 72

**5 Diskusjon** **73**

**6 Administrativt** **79**
6.1 Timeoversikt . . . 79
6.2 Økonomi . . . 80

**7 Konklusjon - Erfaring** **82**
7.1 Videreutvikling . . . 83
7.1.1 Koble utstyr i APS-anlegget til PLS-en . . . 83
7.1.2 Maskinvare . . . 83
7.1.3 SMS-system . . . 83
7.1.4 Batterilading ved hjelp av værmelding . . . 83
7.1.5 Skylagring og datavisualisering . . . 84

**Referanser** **85**

**A Vedlegg - Maskinvareoversikt**

**B Vedlegg - Styrestrømsskjema** **B-1**

**C Vedlegg - I/O-liste**

**D Vedlegg - Sekvensdiagram**

**E Vedlegg - Kommunikasjonsprotokoll**

**F Vedlegg - Funksjonsblokker og funksjoner, dokumentasjon** **F-1**
F.1 FC: FORMAT_STRING . . . . . F-2
F.2 FC: TIME_in_ms_to_REAL_in_mins . . . . . F-3
F.3 FB: BOOL_ALARM . . . . . F-4
F.4 FB: Alarm_Analog_FB . . . . . F-5
F.5 FB: Check_comfault_459 . . . . . F-7
F.6 FC: Scale_AI . . . . . F-8
F.7 FC: Calc_percent_of_total . . . . . F-9

**G Vedlegg - Alarmoversikt**

**H Vedlegg - Oversikt over HMI**

**I Vedlegg - Timeoversikt, Gant- og S-diagram** **I-1**
I.1 Vedlegg - Regnearkinnstillinger . . . . . I-1
I.2 Vedlegg - Gantdiagram, ordinær vs. ny plan . . . . . .
I.3 Vedlegg - Gantdiagram, ny plan . . . . . .
I.4 Vedlegg - S-diagram . . . . . I-4
I.5 Vedlegg - Timeoversikt . . . . . I-5

**J Vedlegg - Poster** **J-1**

# Figurliste


2.1 PFC200 (750-8207) [29] . . . 7
2.2 24 VDC strømforsyning [29] . . . 7
2.3 Digital inngangsmodul [29] . . . 8
2.4 Digital utgangsmodul [29] . . . 8
2.5 Analog inngangsmodul [29] . . . 9
2.6 Motstandsmåler [29] . . . 9
2.7 Energimåler [29] . . . 9
2.8 TP600 [29] . . . 10
2.9 24 VDC rele fra Finder [12] . . . 10

3.1 Virkemåte til en dieselmotor [8]. . . . 12
3.2 Blokkskjema av generelt APS system [7]. . . . 13
3.3 Adresse vs. innhold . . . 17
3.4 Flytskjema av virkemåte til de brukte kvitteringsmetodene. . . . 18
3.5 Lineær interpolasjon . . . 19

4.1 Autopanelet som skal erstattes [25]. . . . 20
4.2 "Task Configuration" - 4 ulike tråder i PLS, kjøres med bestemte intervall. 21
4.3 Aggregat flytskjema - Oppstart av motor, lading av batteri og stopp av motor. Se vedlegg D for et mer detaljert sekvensdiagram for aggregatet. . . 27
4.4 Ladderdiagram for kjøreflagg med alarmstatus og startsignal . . . 28
4.5 Alarmsystemstruktur . . . 32
4.6 Alarmklasseoppsettet for Error_PRI_1 (Feilalarm med høyest prioritet). . 33
4.7 Oppsettet av generator-alarmgruppen . . . 33
4.8 Funksjonsblokk for digital alarmdeteksjon. . . . 35
4.9 Funksjonsblokk for analog alarmdeteksjon . . . 36
4.10 Funksjonsblokk for deteksjon av kommunikasjonsfeil . . . 36
4.11 Eksempel på alarm deteksjon med boolske innganger. . . . 37
4.12 Eksempel på alarm deteksjon med grense-innganger. . . . 38
4.13 Alarmdeteksjon av start- og stoppfeil . . . 39
4.14 Alarmlampe logikk. 1 = Sann, 0 = Usann, tom = likegyldig . . . 39
4.15 FB: Funksjonsblokk for SMS-modemet . . . 42
4.16 SMS filstruktur . . . 43
4.17 SMS Hovedsekvens . . . 44
4.18 Håndtering av innkommende SMS . . . 45
4.19 Håndtering av utgående SMS . . . 46
4.20 Søk av telefonlogg . . . 47
4.21 SMS eksempel . . . 48
4.22 Legge til ny kontakt i telefonlogg over SMS . . . 49
4.23 HMI-hierarki . . . 50
4.24 Illustrasjon av konseptet rundt bruk av rammer ("Frames"). . . . 51
4.25 Eksempel på bruk av maler ("templates"). . . . 51
4.26 Fargetema tabell, fra makrobasert Excel-ark . . . 52
4.27 HMI, toppmeny . . . 53
4.28 HMI menysystem, Hamburgermeny . . . 54

4.29 Logg-inn-vindu original (V) vs. modifisert (H) . . . . . 55
4.30 HMI tastatur og talltastatur, original (bunn) vs. modifisert (topp) . . . . . 55
4.31 HMI-vindu: Oversikt . . . . . 56
4.32 HMI-vindu: Motor . . . . . 57
4.33 HMI-popup: Overstyr temperaturalarm . . . . . 57
4.34 HMI-vindu: Generator . . . . . 58
4.35 HMI-vindu: Motor Innstillinger . . . . . 59
4.36 HMI-popup: Interpolering glødetid . . . . . 59
4.37 HMI-vindu: Generator Innstillinger . . . . . 60
4.38 HMI.vindu: Alarm . . . . . 61
4.39 HMI-vindu: Alarmtest . . . . . 62
4.40 HMI-vindu: SMS . . . . . 63
4.41 HMI-popup: Endring i SMS-kontaktlogg . . . . . 63
4.42 HMI-vindu: SMS innstillinger . . . . . 64
4.43 HMI-vindu: Innstillinger . . . . . 65
4.44 Eksempel på mørk modus . . . . . 65
4.45 HMI-vindu: Avanserte innstillinger side 1 . . . . . 66
4.46 HMI-vindu: Avanserte innstillinger side 2 . . . . . 67
4.47 HMI-vindu: Avanserte innstillinger side 3 . . . . . 67
4.48 HMI-popup: Valg av analog input . . . . . 68
4.49 HMI-vindu: Avanserte innstillinger side 4 . . . . . 68
4.50 Værprognose: systemarkitektur . . . . . 69
4.51 Værprognose: filstruktur . . . . . 70
4.52 Værprognose: GUI med eksempel . . . . . 71
4.53 HMI-vindu: Værprognose . . . . . 72

5.1 APS-anlegg for fysisk testing . . . . . 77
5.2 Provisorisk oppkobling av PLS for fysisk test. Se vedlegg A for maskinvare på DIN-skinne. . . . . 77
5.3 Provisorisk oppkobling av PLS i skap for fysisk test. . . . . 78

# Tabelliste

3.1 Tabell over mye brukte datatyper i programmet. . . . . 16
3.2 Oversiktstabell for oppførselen til variabler etter online-operasjon. (X: verdien huskes) [10]. . . . . 16

4.1 Tabell over datatyper og prefikser som er benyttet i programmet. . . . . 22

6.1 Tabell over den økonomiske løsningen benyttet i prosjektet. *Er erstattet med en nyere modell . . . . . 80
6.2 Tabell over den økonomiske løsningen benyttet for dagens løsning. . . . . 81

# Kodeliste

3.1 Pekere eksempelskode . . . . . 17
4.1 Eksempel på innhenting av verdier fra inngangskort . . . . . 24
4.2 Skriv til utgangskort . . . . . 24
4.3 Valg og skalering av analoge innganger . . . . . 26
4.4 Valg av analoge innganger . . . . . 26
4.5 Alarmobjekt . . . . . 34
4.6 Deklarering av alarmer . . . . . 34
4.7 Array/tabell med pointers/pekere til alarmobjekter . . . . . 34
4.8 Deteksjon av kommunikasjonsfeil . . . . . 38
4.9 Deteksjon av strømbrudd . . . . . 39
4.10 Skript for å kvittere alarmer . . . . . 40
4.11 Utdrag av kode som kvitterer valgt alarm . . . . . 40
4.12 Skript for styring av alarmikon i HMI . . . . . 41
4.13 Kvitterer alle alarmer . . . . . 41
4.14 Nullstiller alle kvitteringer . . . . . 41

# 1. Innledning

## 1.1 Introduksjon

I dette kapittelet vil det bli klargjort bakgrunn for bacheloroppgaven. Det er definert et mål, i form av en problemstilling og dens avgrensninger. I forbindelse med dette vil oppdragsgiver bli presentert, og hvorfor de har valgt ut akkurat denne oppgaven. Mye av informasjonen her er hentet fra en forprosjektrapport, som ligger vedlagt i PA-mappen.

## 1.2 Oppdragsgiver

Getek AS [6] tilbyr strømforsyningssystemer som, i størst mulig grad, baserer seg på fornybar energi. De tilbyr både nettilknyttede og frittstående ("off-grid") systemer. I hovedsak retter de seg mot proff/industrimarkedet. Allerede så tidlig som på 1980-tallet startet Getek med å levere solcelleanlegg til hyttekunder. Derifra har det eskalert og blitt mer og mer lagt over til autonome anlegg, såkalt APS ("Autonomous Power Supply"), som kombinerer sol, vind og diesel som energikilder.

En viktig del av Getek sin virksomhet i dag er bygging og utleie av frittstående autonome strømforsyningsanlegg for steder hvor kunden skal drifte viktig strømkrevende utstyr, men hvor elverkets kabler er utenfor rekkevidde.

I tillegg til Getek AS, har Wago Norge underveis i prosjektet bidratt med maskin- og programvare, opplæring og brukerstøtte. Wago Norge blir dermed sett på som leverandør og en viktig bidragsyter for prosjektet. Wago Norge er et datterselskap av det tyske selskapet Wago, som er en stor produsent og leverandør av komponenter for elektriske tilkoblinger og komponenter for desentralisert automatisering.

## 1.3 Bakgrunn

Bakgrunn for prosjektet og problemstilling er at Getek ønsker å videreutvikle kontroll, overvåkning og visualisering på sine autonome strømforsyningsanlegg (kap. 3.4). Getek ønsker i denne sammenheng å bruke Wago sine produkter. Et slikt autonomt anlegg består typisk av en batteribank (bly eller litium), solcellepaneler med regulator, inverter, lader og et saktegående dieselaggregat. I noen tilfeller er det også brukt vindturbin som energikilde.

Strømforsyningsanlegget består altså av flere forskjellige komponenter som har mulighet for overvåking og styring, men det er forskjellige kommunikasjonsprotokoller og få ting er sammenkoblet. Hovedhensikten til Getek er å kunne sammenkoble alt utstyret på én plat-

tform istedenfor egne innlogginger og brukergrensesnitt til hver komponent. Oppdragsgiver ønsker seg også et system som har mer fleksibilitet og et bedre brukergrensesnitt.

## 1.4 Problemstilling

Dieselaggregatet er i dag overvåket og styrt av et autopanel. Panelet er spesiallaget for styring av generatorer, og har dermed mange funksjoner som Getek ikke benytter seg av. Brukergrensesnittet på panelet er ganske begrenset så oppdragsgiver synes at muligheten til å tilpasse informasjon er for dårlig og de ønsker mulighet til å koble til flere sensorer og utstyr til panelet. Da har ikke alltid autopanelet nok innganger og utganger, som fører til ekstrautstyr som må legges til.

Ved å bruke utstyr fra Wago skal det utvikles et styresystem som erstatter autopanelet og lettere kan tilpasses hver kundes behov. Dette skal gjøres med en Wago PLS, med dedikerte inn- og utgangskort, i tillegg til styring og overvåking fra HMI både eksternt og lokalt på APS. Brukergrensesnittet skal visualisere verdier og driftsstatus med innebygd alarmsystem og innstillinger for aggregatet.

Oppdragsgiver ønsker muligheten for toveis kommunikasjon med anlegget over SMS. Om det oppstår alarmer skal dette varsles på telefon, og man skal ha mulighet til å kvittere alarmene. Den modulære PLS-en gjør det mulig å koble sammen mye utstyr på samme plattform og det er tilrettelagt for å utvide systemet senere.

Som bonus til oppgaven ønsker oppdragsgiver å få utviklet en slags prognose, som regulerer bruk av dieselaggregatet i forhold til lading fra solcellepanel og vindturbin. Denne oppgaven vil bli sett på som en bonus, og skal ikke prioriteres i stor grad.

## 1.5 Avgrensninger

Oppgaven begrenser seg til å erstatte autopanelet som styrer aggregatet. Dette innebærer styringen av dieselaggregatet og et nytt berøringspanel med HMI. Kjøresignal blir hentet eksternt i form av en inngang på enheten og det skal ikke gjøres noe på resten av utstyret APS-anlegget består av.

En annen gruppe jobber for Getek med kommunikasjon og loggføring av data til Wago sine skytjenester. Denne bacheloroppgaven tar derfor ikke for seg oppretting av kommunikasjon mellom de forskjellige komponentene i APS-anlegget eller loggføring til databaser ved hjelp av skytjenester.

## 1.6 Terminologi

| | |
|---|---|
| **ACK** | "Acknowledgement", kvittering av alarm. |
| **AI** | "Analog Input". |
| **AO** | "Analog Output". |
| **APS** | "Autonomous Power Supply", autonom strømforsyning. |
| **API** | "Application Programming Interface". En API-server sender tilbake data ved forespørsel, ofte i form av JSON. |
| **Bibliotek** | En samling av funksjoner og koderessurser. |
| **Bit** | Et siffer i to-tallsystemet, tilsvarende enten 0 eller 1 (Lav / Høy) |
| **DI** | "Digital Input", digital inngang. |
| **DO** | "Digital Output", digital utgang. |
| **Flagg** | En variabel som brukes som et digitalt signal. Et flagg er enten høyt eller lavt. |
| **Føler** | En sensor brukt for å måle ulike målesignal. |
| **GUI** | "Graphical User Interface". Grafisk brukergrensesnitt. |
| **GVL** | "Global Variable List", liste med variabler som er tilgjengelig over alt. |
| **HMI** | "Human-Machine Interface", brukergrensesnitt. |
| **I/O** | "Input/Output", inngang/utgang. |
| **IDE** | "Integrated Development Environment", kodeprogramvare. |
| **Inverter** | Konverterer likestrøm til vekselstrøm, omformer på norsk. |
| **JSON** | "JavaScript Object Notation". Data pakkes inn på en strukturert måte. |
| **LSB** | "Least Significant Bit", minst "viktig" bit i et dataord. |
| **OPC UA** | En maskin til maskin kommunikasjonsprotokoll [3]. |
| **PT1000** | En motstand som varierer med temperatur, med resistansverdi lik 1000 Ω ved 0 ° C. |
| **RPM** | "Revolutions Per Minute", omdreininger per minutt, turtall på motor. |
| **Strømtrans-formator** | Transformerer/skalerer en strøm til en annen strøm, etter et fast forhold. Ofte brukt til å måle større strømmer. |

## 1.7 Rapportens oppbygging

Rapporten er delt opp i 9 hovedkapitler:

1. **Innledning**
2. **Metode og utstyr**
3. **Teori**
4. **Systemstruktur**
5. **Diskusjon**
6. **Administrativt**
7. **Konklusjon**
8. **Referanser**
9. **Vedlegg**

Hovedrapporten er skrevet i henhold til NTNU sine anbefalinger i "Veiledning rapportskriving", som er vedlagt i PA-mappen. Rapporten er skrevet ved bruk av LaTeX, og kilder som er benyttet for å skrive rapporten er sitert i teksten ved hjelp av LaTeX sin innebygde referanseverktøy BibTex. Alle kilder er listet i slutten av rapporten i en bibliografi. Annen informasjon som rapporten bygger på er lagt ved som vedlegg i slutten av rapporten, eller vedlagt i PA-mappen.

Hovedinnholdet i rapporten er skrevet i kapittelet om systemstruktur, der de forskjellige delene av oppgaven beskrives i flere delkapitler. Informasjonen her oppsummeres til slutt i et diskusjon- og konklusjonskapittel.

# 2. Metode og utstyr

## 2.1 Forskrifter og standarder

### 2.1.1 NEK400

NEK 400 spesifiserer krav til prosjektering, utførelse og verifikasjon av elektriske installasjoner. Kravene skal sikre personer, husdyr og eiendom mot farer og skader som kan oppstå ved bruk av installasjonen, og sikre at installasjonen er egnet for den forutsatte bruk [14]. Normen er basert på Forskrift om elektriske lavspenningsanlegg (FEL). Ikke alt i normen må følges, men det skal dokumenteres gode faglige begrunnelser til hvorfor det ikke er fulgt.

### 2.1.2 IEC 61131-3 Standard

IEC 61131-3 er en internasjonal standard som spesifiserer grunnleggende programvarearkitektur og programmeringsspråk for kontrollprogrammet i PLS-er. Ifølge IEC [11] består den oppdaterte standarden fra 2013 av 4 programmeringsspråk: Instruksjonsliste (IL), Strukturert tekst (ST), Ladderdiagram (LD) og funksjonsblokkdiagram (FBD).

### 2.1.3 Maskindirektivet

Maskindirektivet er en forskrift om maskiner. Den har som formål å sikre at maskiner og sikkerhetskomponenter konstrueres og bygges slik at arbeidstakere, forbrukere, dyr og gjenstander er vernet mot skade på liv og helse og ikke blir utsatt for uheldige belastninger. Maskindirektivet angir i all hovedsak funksjonskrav og det er en viss valgfrihet med hensyn på tekniske løsninger [19].

## 2.2 Programvare

### 2.2.1 e!COCKPIT

e!COCKPIT er en IDE som Wago benytter for å programmere sine PLS-er. Programvaren er bygd på Codesys, som er et utviklingsmiljø for programmering av kontrollapplikasjoner i henhold til den internasjonale industristandarden IEC 61131-3. Programvaren støtter språkene Ladderdiagram (LD), "Sequential Function Chart" (SFC), Funksjonsblokkdiagram (FBD), "Continous Function Chart" (CFC) og Strukturert Tekst (ST). I tillegg til dette er det også mulighet til å programmere brukergrensesnitt direkte i programmet, slik at bruker enkelt kan visualisere verdier fra PLS-en.

### 2.2.2 PC Schematic

PC Schematic er en programvare brukt til tegning og dokumentering av elektriske, pneumatiske og hydrauliske systemer. Ofte blir det brukt til tegning av koblingsskjemaer, rekkeklemmetabeller og tabeller over interne forbindelser. Programvaren støtter eksportering til PDF- og DWG-filer.

### 2.2.3 LaTeX/ Overleaf

LaTeX er et gratis tekstredigeringsspråk som er godt egnet for å skrive faglige og vitenskapelige dokumenter [23]. Hovedrapporten i prosjektet er skrevet i LaTeX i programmet Overleaf, som er et skybasert tekstredigeringsprogram på nett. Programmet lar flere brukere være inne på et og samme dokument, slik at det er enkelt å kunne redigere samtidig. Overleaf er også i samarbeid med NTNU, slik at NTNU-studenter får tilgang til en utvidet versjon kalt "Overleaf Professional" [20].

### 2.2.4 Excel

Microsoft Excel er et dataprogram produsert av Microsoft og er en del av Microsoft Office-pakken. Programmet blir ofte brukt som regneark og prosjektledelse, men det er også mulighet for å lage større og dynamiske regneark. Det er mulighet for å programmere funksjoner ved hjelp av VBA (Visual Basic for Applications), som er et hendelsesdrevet programmeringsspråk. Excel er i dag Turing-komplett, som betyr at bruker kan kalkulere hva som helst uten bruk av andre programmeringsspråk [31].

## 2.3 Maskinvare

For å gjennomføre prosjektet, har det vært nødvendig med en del maskinvare. Dette kapittelet vil gå gjennom komponentene som har blitt brukt, og forklarer kort om hvorfor komponentene ble valgt. Hovedmaskinvaren består i grove trekk av en PLS (kap. 3.1), og tilleggsmoduler for måling og utsending av signaler. Felles for alle I/O modulene er at de er koblet fysisk til PLS-en, og kommuniserer over en intern buss [16]. For mer oversikt over maskinvare se vedlegg A.

### 2.3.1 PFC200 (750 - 8207)

PFC200 - 750-8207 er en kompakt PLS fra Wago. Denne PLS-en kjører på et Linux operativsystem og er modulær, ettersom den støtter mange ulike I/O tilleggsmoduler i Wago sin 750 serie. Flere av disse modulene blir gått gjennom i de neste delkapitlene.

I det integrerte kortet til høyre på PLS-en finner man strømforsyningen til kontrolleren. Denne er todelt, hvor de to øverste klemmene er til 24 VDC strømforsyning til drift av PLS- systemet. De 6 nederste klemmene er til feltsiden, som skal forsyne de eksterne I/O-kortene.

**Figur 2.1:** PFC200 (750-8207) [29]

### 2.3.2 Strømforsyning (787 - 1212)

PLS-en krever 24 VDC forsyningsspenning, og det er dermed nødvendig med en strømforsyning. Prototypen bruker en AC-DC omformer slik at PLS-en kan kobles til 230 VAC i stikkontakten. Forsyningen som er brukt er av type 787-1212 fra Wago. Denne gir ut 24 VDC (maks 60 W) som gjør den godt egnet til å forsyne PLS-ens I/O moduler, som også krever 24 VDC. I en permanent løsning vil det være mer aktuelt med andre alternativer for strømforsyning. Ideelt vil PLS-en få strøm fra batteribanken i APS, med en DC-DC omformer.

**Figur 2.2:** 24 VDC strømforsyning [29]

### 2.3.3 DI - Digital Inngangsmodul (750 - 430)

Denne 8 kanals modulen blir brukt til digitale inngangssignaler, i form av 24 VDC spenning. Når en av inngangene blir påtrykt 24 VDC spenning vil dette sette inngangen høy i PLS-en. 0 V potensialet har den internt ifra strømforsyningskortet som er integrert i PLS-en. I dette prosjektet blir denne brukt til ulike følere som digital oljetrykkssensor, motortemperatur og dieselnivå og signaler fra nødstoppknapp, dynamo og startsignal.

**Figur 2.3:** Digital inngangsmodul [29]

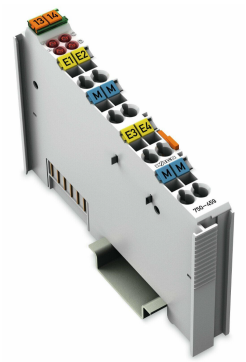
### 2.3.4 DO - Digital Utgangsmodul (750 - 530)

Til å styre enheter har gruppen brukt en digital utgangsmodul. Når utgangen blir satt høy i programmet inne i PLS-en, vil utgangen få et potensiale på 24 VDC sett fra strømforsyningskort. Siden det er begrenset med strøm dette kortet/PLS-en kan levere, og det ikke alltid er 24 VDC utstyr som skal styres, er utgangene på kortet brukt til styrestrøm for aktivering av 24 VDC releer, som aktiveres ved høy utgang. Enheter som skal styres med denne modulen er komponenter i en motor, slik som gløder, startmotor, driftsmagnet og en lastkontaktor.

**Figur 2.4:** Digital utgangsmodul [29]

### 2.3.5 AI - Analog Inngangsmodul (750 - 459)

Denne analoge inngangsmodulen består av 4 kanaler for analog avlesning. Den måler inngangsspenning fra 0-10 VDC, og detekterer i tillegg om inngangsspenningen er utenfor godkjent måleområdet. I prosjektet kan denne brukes med måleomformere til å måle batterispenninger, og ved bruk av analoge følere til måling av drivstoffnivå, temperatur i kjølevæske og mer.



**Figur 2.5:** Analog inngangsmodul [29]

### 2.3.6 RTD - Motstandsmåler (750 - 463)

Motstandsmåleren er en analog I/O modul som måler motstand/resistans. Denne modulen er spesiallaget for tilkobling til et PT1000 element, slik at man kan måle temperaturer. Modulen har mulighet for 4 samtidige målinger, men har kun 2 målekretser. Dette betyr at om man for eksempel kun skal koble til et PT1000 element, må man slutte målekretsen ved hjelp av en motstand eller kortslutning i området $0 - 5k\Omega$ i motstående kanal.

**Figur 2.6:** Motstandsmåler [29]

### 2.3.7 Energimåler (750 - 494)

Energimåleren kan måle spenninger og frekvens på et 3 fase strømnett, samt måle strømmer på hver av fasene. I de fleste tilfeller må det brukes en strømtransformator for strømmålingene, ettersom denne modulen ikke tåler mer enn maks 1 ampere. Modulen har svært mange innstillinger som kan nås via e!COCKPIT, og Wago tilbyr biblioteker for å enkelt lese av signaler. Denne modulen blir i hovedsak brukt til måling av frekvens og spenning fra generator.

**Figur 2.7:** Energimåler [29]

### 2.3.8 Endemodul (750 - 600)

Endemodulen er et obligatorisk kort for alle modulbaserte PLS-er i Wago sin 750 serie. Den plasseres som siste modul i "rekken" på PLS-en og har som oppgave å avslutte den interne datakretsen, og sikre god dataflyt. Kortet ser visuelt helt likt ut som de overnevnte modulene/kortene, men har ingen tilkoblingsmuligheter eksternt.

### 2.3.9 HMI-skjerm - TP600 (762-4303)

Enheten gruppen har brukt til HMI-visualisering er en trykkfølsom skjerm fra Wago på 7 tommer, av modell 762-4303/8000-0002. Dette panelet er programmerbart og kan fungere fint for seg selv også uten en PLS. Den har et innebygd nettverkskort, og kan dermed brukes til visualisering direkte fra webserveren fra eksempelvis en PLS, som er gjort i dette prosjektet. Dette gjøres ved hjelp av en direkte ethernet-tilkobling fra panel til PLS.

**Figur 2.8:** TP600 [29]

### 2.3.10 Releer

Autopanelet som erstattes og elektronikken på motoren bruker 12 VDC. PLS-en som benyttes i dette prosjektet driftes av 24 VDC. Prototypen trenger derfor mellomreleer med 12 VDC spoler på innganger og 24 VDC spoler på utganger for at oppkoblingen skal være mulig. Kontaktsettene i releene er potensialfrie som gjør at de egner seg godt som mellomreleer.

**Figur 2.9:** 24 VDC rele fra Finder [12]

# 3. Teori

For å få en bedre forståelse av hva denne rapporten handler om, er det nyttig å forstå noen essensielle konsepter. I de neste kapitlene blir det gjennomgått nødvendig teori for å forstå hvordan hovedoppgaven er bygd opp og for å forstå hvordan det ferdige systemet fungerer.

## 3.1 Programmerbar Logisk Styring (PLS)

En PLS, eller en programmerbar logisk styring, er en kompakt datamaskin som ofte er brukt i industrielle sammenhenger [27]. Ved hjelp av en programvare tilknyttet PLS-en, kan man utvikle et program som PLS-en kan kjøre. Dermed kan en generell PLS programmeres til å utføre mange slags oppgaver, alt etter hva behovet er. PLS-en kan ta inn verdier fra forskjellige målere eller sensorer, bruke denne informasjon innad i programmet som er lastet opp, og styre dens utganger deretter.

I motsetning til en vanlig datamaskin er PLS-er spesialdesignet for å operere i industri på en driftssikker måte. Ofte er de modulære slik at bruker kan velge de I/O-kortene som trengs til den spesifikke oppgaven den gjeldende PLS-en skal utføre. PLS-er har også muligheten til å kommunisere med andre enheter over en rekke forskjellige og kjente kommunikasjonsprotokoller, som for eksempel OPC UA eller Modbus [18].

En stor fordel med PLS-er er at de bruker programmeringsspråk basert på IEC 61131-3 standarden. Flere av språkene er grafiske, som gjør de relativt enkle å programmere i. Dette er spesielt med tanke på personer med lite erfaring innen tekstprogrammering. Det er mulig å blande sammen flere språk i samme PLS. Dette gjør det mulig å bruke det språket som passer best til oppgaven. Med unntak av strukturert tekst (ST) og instruksjonsliste (IL), kan vedlikeholdsarbeidere med lite erfaring innen programmering lese et program godt nok til å bruke det under feilsøking.

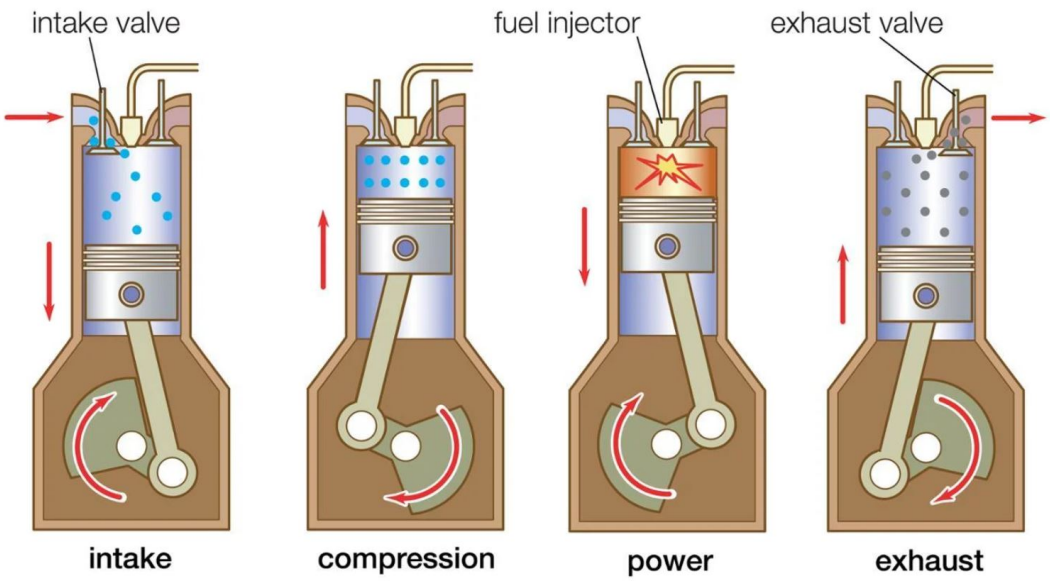
## 3.2 Autopanel

Et autopanel er et styringssystem med et integrert HMI-panel. Panelet er programmerbart, og kan brukes for autonom styring av diverse systemer. HMI, "Human Machine Interface", er grensesnittet for menneske-maskin interaksjon, altså overgangen fra menneske til maskin i form av et brukergrensesnitt. Her kan bruker overvåke verdier, endre på grenser, kvittere alarmer og lignende, altså generell overvåking og styring av en prosess. Brukergrensesnittet er koblet opp mot en kontroller som er blitt programmert til å styre maskinen gjennom lesing og skriving til mange forskjellige variabler. Styringen er innebygget i autopanelet, men noen variabler kan endres gjennom HMI-en for tilpassing til brukssituasjon.

## 3.3 Dieselaggregat

Et dieselaggregat er en sammensatt maskin som består av en dieselmotor og en generator [4]. Aggregatet gjør mekanisk energi fra omdreininger i motoren om til elektrisk energi i generatoren.

En dieselmotor fungerer ved selvantenning av diesel, i motsetning til en bensinmotor som benytter tennplugger. Ved oppstart av en dieselmotor brukes glødeplugger som varmer opp luften for at dieselen lettere skal kunne selvantennes. Motoren tar inn luft i sylinderne, som blir komprimert av et stempel som er koblet på en motoraksling. I det stempelet når toppunktet blir dieselen sprøytet inn og blandes med den varme komprimerte luften. Den høye temperaturen i blandingen fører til selvantenning, og dermed en eksplosjon. Denne eksplosjonen fører til en trykkøkning som presser stempelet nedover og får akslingen til å gå rundt [5].




**Figur 3.1:** Virkemåte til en dieselmotor [8].

Akslingen fra dieselmotoren er tilkoblet en generator som skal gjøre den mekaniske energien om til elektrisk energi. Denne omgjøringen av energi går ut på det elektromagnetiske induksjonsprinsippet. En generator består av to hoveddeler: en stator og en rotor. Statoren er som beskrevet av navnet fastkoblet, og består av flere spoler. Rotoren er koblet til motorakslingen, og består av en spole med en jernkjerne. Når spolen i rotoren blir påført spenning, funger denne som en elektromagnet og det skapes dermed et magnetfelt som går igjennom spolene i statoren [26].

Siden magnetfeltet gjennom spolene i statoren endrer seg i det akslingen roterer, blir det indusert en spenning i henhold til Faradays induksjonslov som sier at den induserte spenningen er lik antall spolevindinger ganger endring i magnetisk fluks:

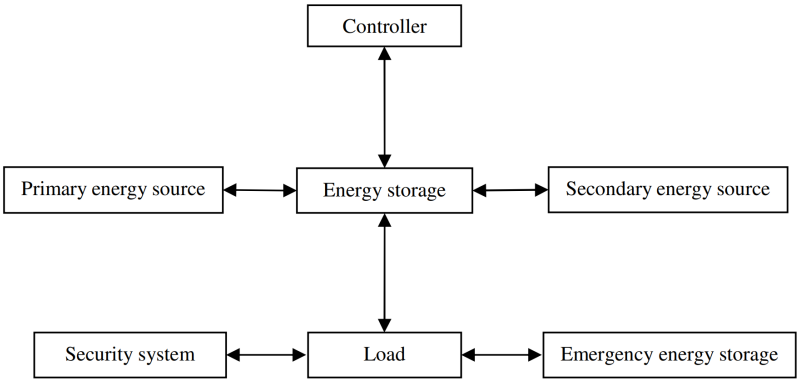
$$\varepsilon = -n\frac{d\Phi}{dt}$$

, hvor n er antall vindinger i spolen og $\Phi$ er den magnetiske fluksen [15]. For at utgangssignalet skal holdes mest mulig konstant på 230 VAC brukes en AVR ("Automatic Voltage Regulator"). Denne spenningsregulatoren holder spenningen jevn uavhengig av hvor mye strøm som brukes og om motor RPM varierer litt under belastning.

Frekvensen er derimot direkte forbundet som en skalar av motor-RPM. På APS-enheten, som testingen ble utført med, var forholdet 1:2 mellom motor og generator. Det vil si at 1500 omdreininger i motor gir 3000 RPM, som er lik 50 Hz ut fra generatoren. Det er derfor viktig at motoren holder turtallet på 1500 konstant for stabil frekvens. Hvis diesel- eller lufttilførselen til motoren blir begrenset, vil turtallet synke og det er da viktig å koble fra lasten på generatoren, så elektronisk utstyr ikke tar skade.

## 3.4 Autonom strømforsyning - APS

En autonom strømforsyningsenhet, APS ("Autonomous Power Supply") [7], er en enhet som automatisk forsyner en last. En APS består i hovedsak av en kontroller, energilagring (batteribank), primær- og sekundærenergikilde, og en last som skal forbruke den produserte energien. Primærenergikilden står generelt for minst 80% av produksjonen av det totale forbruket, men dette kommer selvfølgelig an på hvor god og stabil sekundærenergikilden er. I dette prosjektet kan man se på et dieselaggregat som hovedkilden, mens det oftest er solcellepanel, og i noen tilfeller en vindturbin som er sekundærkilden. Ved hjelp av et styresystem i en kontroller skal den kontrollere lading av batteribank, og koble inn og ut last når dette er nødvendig.




**Figur 3.2:** Blokkskjema av generelt APS system [7].

## 3.5 PLS-programmering

### 3.5.1 Programmeringsspråk

Med unntak av CFC er alle programmeringsspråkene i dette delkapittelet basert på, og i henhold til, IEC 61131-3 standard. CFC er ikke definert i IEC 61131-3 og er en utvidelse av standarden.

#### 3.5.1.1 Strukturert Tekst (ST)

Strukturert Tekst (ST) er et tekstbasert programmeringsspråk som kan ligne på programmeringsspråket C. Språket benytter seg av flere innebygde funksjoner som IF-setninger, FOR- og WHILE-løkker og flere andre kjente funksjoner som benyttes i andre programmeringsspråk. Denne typen programmering gjør det enklere å lage mer kompliserte programmer enn i de grafiske språkene, da man kan benytte de innebygde funksjonene til å gjøre en rekke oppgaver. Ulempen her er at det ofte er vanskeligere, og mer tidkrevende å forstå programmet, spesielt for personer uten mye programmeringskunnskap.

#### 3.5.1.2 Sekvensprogrammering (SFC)

"Sequencial Function Chart" (SFC) eller sekvensprogrammering er et grafisk programmeringsspråk som benyttes for et program som skal kjøres i en bestemt rekkefølge [9]. Programmene som lages med denne typen programmering er ofte laget for en sekvens som skal kjøres flere ganger ved bestemte krav eller hendelser, som for eksempel et start/stopp-signal. En stor fordel med SFC er at det er veldig enkelt å se for seg hendelsesforløpet i programkoden, da man kan se hvilket steg programmet befinner seg i, og hva som er neste steg. Ulempen er at det er vanskeligere å lage et komplisert program uten at strukturen ser rotete ut.

En stor fordel med SFC i e!COCKPIT er at man har mulighet til å definere ulike handlingsskript, som enkelt deklareres til å kjøre i de ulike stegene. Disse skriptene kan skrives i valgfritt språk (kap. 3.5.1). Man har mulighet til å velge om skriptet kun skal kjøres ved inngangen til et steg, ved utgangen av et steg, eller om det skal kjøre kontinuerlig så lenge det gitte steget er aktivt.

#### 3.5.1.3 Ladderdiagram (LD)

Ladderdiagram (LD) er et grafisk programmeringsspråk som er basert på elektriske koblingsskjemaer [9]. Dette er et av de mest brukte programmeringsspråkene. Det blir ansett som et enkelt språk for elektrikere og andre tekniske vedlikeholdspersonell å lære seg, nettopp fordi det ligner veldig på koblingsskjemaer. Ulempen med ladderprogrammering er at oppsettet av programmet er større, og at mer avanserte programmer fort blir rotete og ofte unødvendig kompliserte.

#### 3.5.1.4 Funksjonsblokkdiagram (FBD)

"Function Block Diagram" (FBD) eller funksjonsblokkdiagram er et grafisk programmeringsspråk som tar utgangspunkt i en funksjonsblokk med inn- og utganger. Blokken fungerer som en slags mikrokontroller, hvor det som skrives på inngangene brukes i et internt program og avgjør hva som skal skrives på utgangene. Funksjonsblokkdiagram kan inneholde flere funksjonsblokker som kan kobles sammen til et større program, [9].

I likhet med ladderdiagram er funksjonsblokkdiagram grafisk, og anses som et relativt enkelt programmeringsspråk å lære seg. Mens ladderdiagram ligner mer på styrestrømsskjemaer, likner funksjonsblokkdiagram mer på elektronikk-kretser.

#### 3.5.1.5 Continous Function Chart (CFC)

"Continous Function Chart" er et grafisk programmeringsspråk som strekker seg utenfor IEC 61131-3 - standarden. CFC likner veldig på et funksjonsblokkdiagram, men hovedforskjellen er rekkefølgen på hvordan programmet kjøres. FBD har en nettverksbasert rekkefølge på hvordan programmet kjøres, mens et CFC-program kjører i henhold til dataflyten i programmet [21].

CFC har samme fordeler som FBD. I tillegg er det lettere å lage større, mer komplekse strukturer, da man slipper å splitte opp i flere nettverk. Ulempen er at det er vanskeligere å vite dataflyten, da det mangler nettverksoppdeling.

### 3.5.2 Deklarering av variabler

Alle variabler må deklareres før de kan brukes i et program. En variabel deklareres som en spesifikk datatype slik at riktig mengde minne blir designert til variabelen og slik at kompilator kan kompilere skikkelig. Det er mulig å aksessere minne direkte ved hjelp av pekere (kap. 3.5.3), men dette blir fort uoversiktlig.

Større objekter som funksjonsblokker, strukturer ("structs") og instanser av klasser må også deklareres. Man sier da at de deklareres som instanser eller objekter, og ikke som variabler. Dette er fordi disse objektene ikke består av én enkel datatype, men flere og ofte forskjellige datatyper [17].

En funksjon (FC) i motsetning til en funksjonsblokk (FB) må ikke deklareres. En funksjon, i likhet med en metode (method), tar ikke vare på internt minne til neste programsyklus. Minne knyttet til funksjonen eller metoden frigis etter hver programsyklus, og kan dermed brukes på nytt hvor som helst i programmet.

#### 3.5.2.1 Datatyper

Siden e!COCKPIT er bygget på Codesys, støtter den alle datatypene som inngår i IEC 61131-3 [22]. De mest brukte variablene her er boolske, heltall ("integers") og desimaltall ("real"). Noen av disse datatypene vises i tabell 3.1.

| Datatype | Nedre Grense | Øvre Grense | Minne |
|---|---|---|---|
| BOOL | FALSE (0) | TRUE (1) | 4 Bit |
| INT | -32768 | 32767 | 16 bit |
| DINT | -2147483648 | 2147483647 | 32 bit |
| UINT | 0 | 65535 | 16 bit |
| USINT | 0 | 255 | 8 bit |
| REAL | -3.402823e+38 | 3.402823e+38 | 32 bit |
| TIME | 0 | 4294967295 | 32 bit |
| POINTER | 0x00000000 | 0xFFFFFFFF | 32 bit |

**Tabell 3.1:** Tabell over mye brukte datatyper i programmet.

#### 3.5.2.2 Remanente variabler

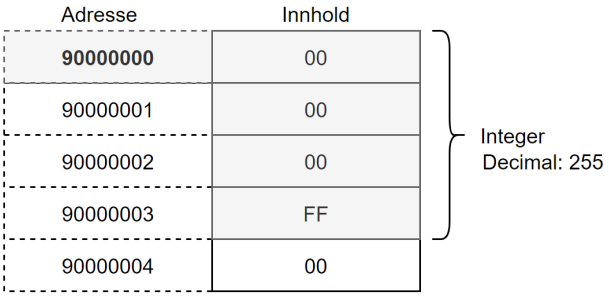
Remanente variabler [10] husker verdien sin ved strømbrudd. Disse variablene lagres ofte på eget sted i minne og har derfor en begrenset mengde minne. Det skilles mellom "RETAIN" og "RETAIN PERSISTENT". I dette prosjektet er det kun brukt "RETAIN PERSISTENT", og ikke "RETAIN". For en full oversikt over hvordan de remante variablene holder på verdien sin, se tabell 3.2.

| Online operasjon | VAR | VAR RETAIN | VAR RETAIN PERSISTENT |
|---|---|---|---|
| Reset «Varm» | | X | X |
| Reset «Kald» | | | X |
| Reset «Opprinnelse» | | | |
| Nedlastning | | | X |
| Online endring | X | X | X |

**Tabell 3.2:** Oversiktstabell for oppførselen til variabler etter online-operasjon. (X: verdien huskes) [10].

### 3.5.3 Pekere

En peker, eller pointer på engelsk, er en datatype som peker til adressen til en annen variabel. All data lagres i minne og har en korresponderende adresse. Figur 3.3 viser et eksempel på hvor man har en peker som peker til et heltall ("integer"). Verdien til pekeren vil være adressen til første byte i datatypen den peker til. I dette eksempelet er adressen 90000000. Når en pointer blir deklarert oppgir man hvilken datatype den skal peke til, i dette tilfellet blir den deklarert som en "POINTER TO INT". Den vet dermed hvor mange adresser fremover den må lese. Man kan dereferere pekeren for å lese innholdet, som vist på linje 6 i kode 3.1.




**Figur 3.3:** Adresse vs. innhold

Pekere gir direkte tilgang til minne og man kan returnere flere verdier, deriblant større datatyper som strukturer. Dette vil spare på minne, og pekere kan i tillegg redusere syklustiden til programmet. Man må også være forsiktig ved bruk av pointere. Hvis det prøves å dereferere en pointer som peker til NULL kan programmet i verste fall krasje. Det er derfor en god ide å sjekke at den ikke peker til NULL før man prøver å dereferere som vist på linje 5 i kode 3.1.

```
1 iInteger  : INT := 255;      // Deklarering Integer
2 piPointer : POINTER TO INT; // Deklarering Pointer
3
4 piPointer := ADR(iInteger); // Pek pointer -> adressen til iInteger
5 IF piPointer <> 0 THEN      // Sjekk at pointer ikke peker til NULL!
6     piPointer^ := 123;      // Dereferer pointer og skriver over innholdet.
7 END_IF                      // iInteger er naa 123
```

**Kode 3.1:** Pekere eksempelskode

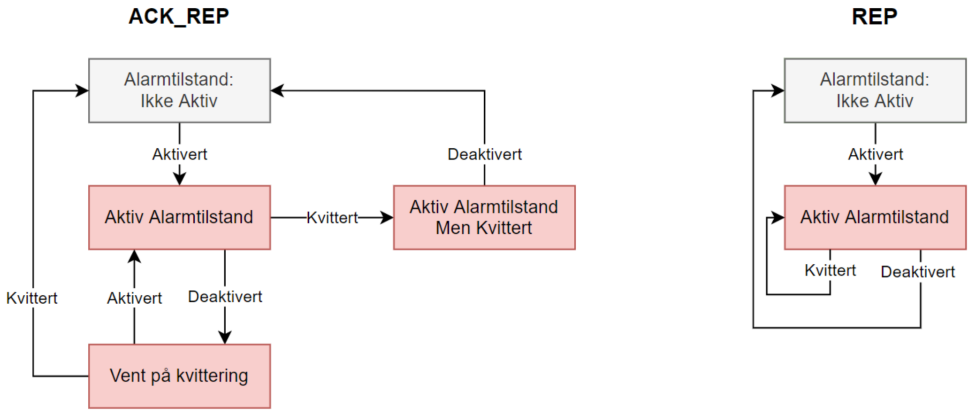
### 3.5.4 Alarmkonfigurator

Alarmkonfigurator ("Alarm Configurator") er en ferdig implementert modul i e!COCKPIT og Codesys som brukes for å sette opp et alarmsystem. Konfiguratoren er sammenkoblet med en alarmtabell som kan vise aktive alarmer i en visualisering (kap. 4.6.7.6). I selve konfiguratoren kan man sette opp alarmklasser og alarmgrupper som man selv kan modifisere til sitt bruk. Alarmene som settes opp i alarmgruppene blir lagret i "Alarm storage", som er en database hvor alarmene som oppstår lagres og som kan visualiseres i alarmhistorikk.

#### 3.5.4.1 Alarmklasser

I konfiguratoren settes det opp ulike alarmklasser som beskriver de ulike alarmtypene. Hver alarmklasse har en prioritetsverdi som bestemmer graden av alvorlighet, der én er den høyeste prioriteten. Alarmklassene kan kjøre forskjellige aksjonsskript som kjøres når en alarm av denne klassen går høy. Denne funksjonaliteten er brukt i prosjektet for å telle antall aktive alarmer. Her settes det også opp hvilken kvitteringsmetode alarmen skal følge, som beskriver måten en alarm blir deaktivert på. I dette prosjektet er det i hovedsak benyttet to ulike kvitteringsmetoder: ACK_REP og REP.

Alarmer deklarert som ACK_REP (kvitter og reparer) krever at den aktive alarmen først kvitteres, før den igjen sjekker om alarmen er stående. Om alarmen ikke er stående, altså er reparert, vil gjeldende alarmen bli deaktivert. Den integrerte modulen i e!COCKPIT har dette innebygd, men denne funksjonaliteten er i dette prosjektet implementert ved hjelp av en funksjonsblokk. Alarmer deklarert som REP (reparer) blir deaktivert ved at alarmsituasjonen forsvinner/repareres. Eksempel på dette kan være måling av drivstoffnivå hvor alarmen er aktiv når nivået er for lavt, men med en gang tanken fylles vil alarmen deaktiveres av seg selv.



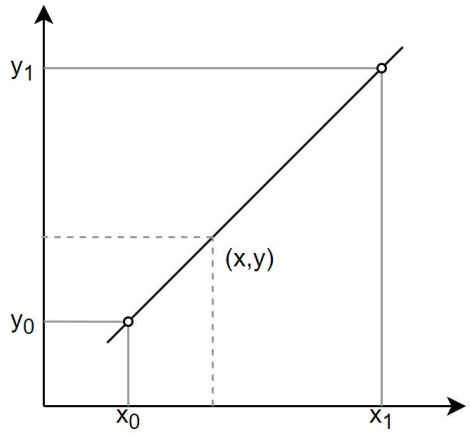

**Figur 3.4:** Flytskjema av virkemåte til de brukte kvitteringsmetodene.

#### 3.5.4.2 Alarmgrupper

Alarmgrupper kan ses på som en samling med alarmer som tilhører samme del av et system. Alarmene i hver gruppe kan være av forskjellige alarmklasser, og kan detekteres både analogt og digitalt. Alarmgruppene er i tillegg koblet sammen med en tekstliste med samme navn som alarmgruppen. Disse tekstlistene inneholder ID-ene til alarmene i alarmgruppen, samt alarmmeldingen som blir fremvist i alarmtabellen når alarmen med tilsvarende ID går høy.

## 3.6 Lineær interpolasjon

Lineær interpolasjon er en matematisk metode for å angi funksjonsverdier mellom to gitte punkt. Ved å anta at det er lineær sammenheng mellom de to gitte punktene, utledes en formel for å tilnærme funksjonsverdier i dette intervallet [30].




**Figur 3.5:** Lineær interpolasjon

Gitt to punkter $(x_0, y_0)$ og $(x_1, y_1)$ med kjente verdier. Stigningen i punktet $(x_0, y_0)$ tilsvarer stigningstallet $a_0$, og stigningen i punktet $(x_1, y_1)$ tilsvarer stigningstallet $a_1$. Siden punktene ligger på samme linje i intervallet $x \in [x_0, x_1]$, vil de to stigningstallene være lik hverandre. Denne sammenhengen brukes for å utlede formelen for lineær interpolasjon:

$$a_0 = a_1 \Longrightarrow \frac{y - y_0}{x - x_0} = \frac{y_1 - y}{x_1 - x}$$

Ved å løse likningen for y, der x blir en fri variabel, finner man at formelen for lineær interpolasjon tilsvarer funksjonen

$$y = y_0 + (x - x_0)\frac{y_1 - y}{x_1 - x}$$

der funksjonen gjelder i intervallet $x \in [x_0, x_1]$.

Dette kan direkte kobles til den mer kjente formelen

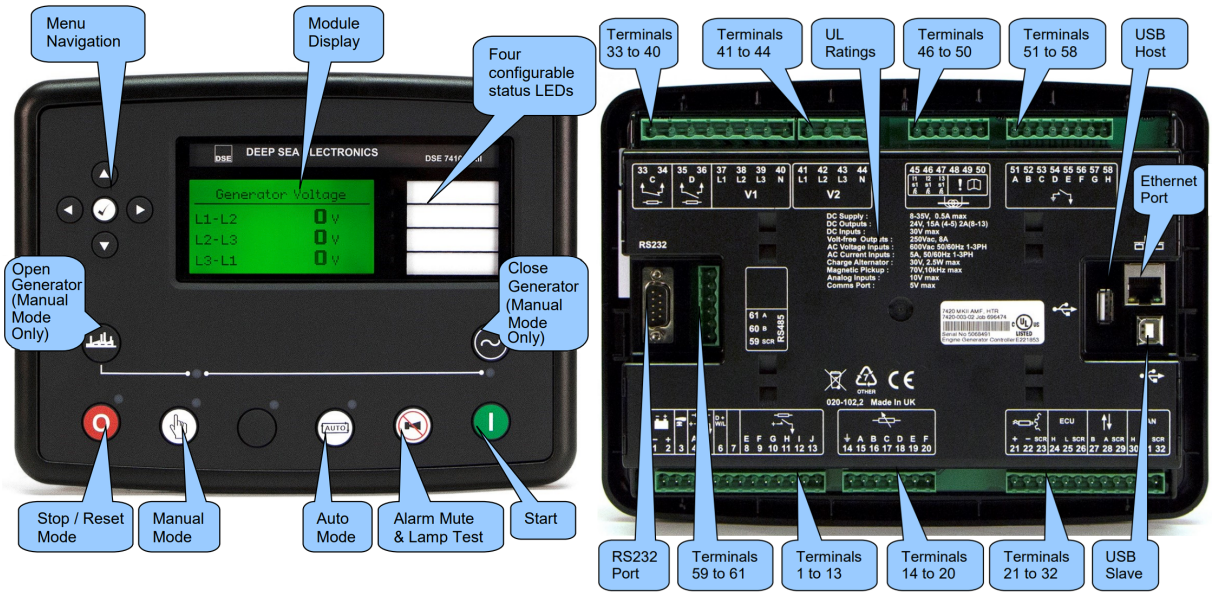
$$y = ax + b, \text{ der } \begin{cases} a = & \frac{y_1 - y}{x_1 - x} \\ b = & y_0 - \frac{y_1 - y}{x_1 - x} x_0 \end{cases} \tag{3.1}$$

Metoden er brukt for å finne en lineær funksjon for glødetid ut ifra glødetiden ved to temperaturer. Dermed kan glødetiden regnes ut fra omgivelsestemperaturen, og man slipper unødig lang glødetid på varme dager.

# 4. Systemstruktur

## 4.1 Autopanel som skal byttes ut

Oppdragsgiver har benyttet autopanel fra industriselskapet Deep Sea Electronics (DSE). I APS-anlegget hvor styresystemet skal erstattes, er det installert et autopanel "7410 MKII" fra DSE. Dette er laget med automatisk start og manuell modus for styring av dieselaggregatet. Forsiden styrer aggregatet via knapper og baksiden består av alle tilkoblingene, som vist i figur 4.1. DSE-panelet har et innebygget alarmsystem og bytter på å visualisere flere verdier. På forsiden er det fem knapper for navigering i menyer, fire LED-lys for alarm og noen knapper for manuell og automatisk modus. Ved tilkobling til PC, over den innebygde webserveren eller direkte med USB, er det tilgang til å endre ganske mye innstillinger.

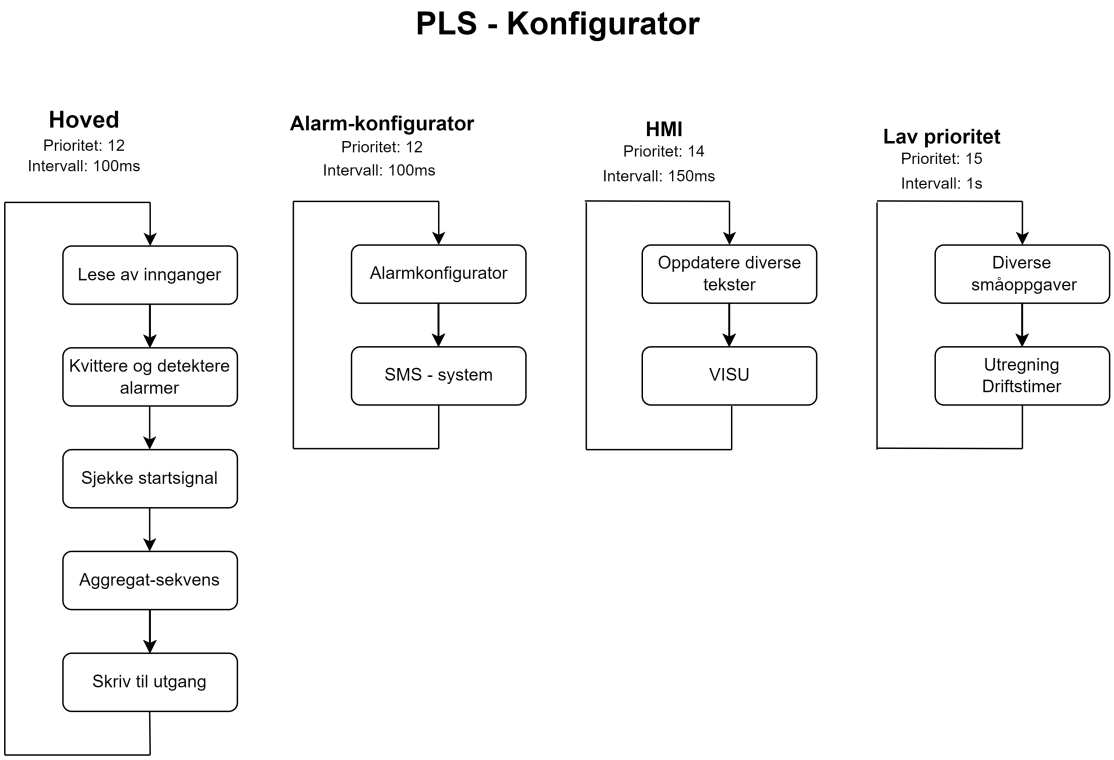



**Figur 4.1:** Autopanelet som skal erstattes [25].

## 4.2 Generell oppbygging

### 4.2.1 Oppgavebehandling i PLS

Programmene som er laget i PLS-en må defineres i en "task"/tråd for at de skal kjøres. I det fullstendige programmet er det laget fire tråder med ulik prioritet og tidsintervall. Grunnen til dette er at ikke alle oppgavene som er laget er like viktige, og fordi ulike koder bør kjøres i forkant/etterkant av andre kodebiter. Som vist i figur 4.2, er tråden kalt "Hoved" den med lavest prioritetsnummer, som tilsvarer høyest prioritet i programkjøringen. Dette betyr at om det skjer en konflikt i programmet, så vil denne bli prioritert **før** andre tråder med et høyere prioritetsnummer.




**Figur 4.2:** "Task Configuration" - 4 ulike tråder i PLS, kjøres med bestemte intervall.

Hovedtråden starter med å lese innganger før den bruker disse verdiene til å detektere alarmer. Deretter sjekker den om kriteriene for at det interne kjøreflagget er høyt før den går videre til aggregatsekvensen. I sekvensen styres det hvilke utganger på PLS-en som skal være høye og lave.

Tråden "Alarm-konfigurator" er knyttet opp mot det interne forhåndsprogrammerte alarmsystemet til e!COCKPIT. I denne tråden kjøres også programmene for SMS systemet, som tar hånd om inngående og utgående SMS (kap. 4.5). Deretter kommer tråden for HMI, altså visualisering og overvåking (kap. 4.6). Det kjøres først diverse programsnutter som endrer tekster og lignende som brukes i HMI-en, før hovedprogrammet "VISU" kjører den faktiske visualiseringen. Til slutt har er det tråden med lavest prioritet, den kjøres med en god del høyere tidsintervall enn de andre trådene. Denne tråden kjører diverse ikke-kritiske småoppgaver, og inneholder også et program som kontinuerlig oppdaterer driftstimer til aggregatet.

Kommunikasjon mellom PLS og HMI-panelet vil skje automatisk så lenge de begge har strømtilførsel, og de fysisk er tilkoblet hverandre med ethernetkabel. I HMI-tråden kjøres det i prinsippet en webserver [2] som kan nås på PLS-ens IP-adresse. Panelet er konfigurert slik at når det skrues på vil det automatisk prøve å koble seg til denne adressen.

### 4.2.2 Programmering

#### 4.2.2.1 Variabelnavn

Det er benyttet spesifiserte prefiks for å deklarere variabler i e!COCKPIT. Dette gjør det lettere å se datatypen til hver enkel variabel, og er med på å holde bedre struktur i programmet. Denne type navngiving kalles ungarsk notasjon [1]. Noen av prefiksene som denne notasjonen inneholder er vist i tabell 4.1.

| Datatype | Prefiks | Minne |
|---|---|---|
| BOOL | x | 4 bit |
| INT | i | 16 bit |
| DINT | di | 32 bit |
| UINT | ui | 16 bit |
| USINT | usi | 8 bit |
| REAL | r | 32 bit |
| POINTER | p | 32 bit |
| STRING | s | |
| TIME | d | 32 bit |
| ARRAY | a | |
| Globale variabler | g_ | |

**Tabell 4.1:** Tabell over datatyper og prefikser som er benyttet i programmet.

#### 4.2.2.2 Variabelliste

For å lage struktur i deklarering av variabler er det laget en rekke globale variabellister (GVL). Listene inneholder variabler som hører til ulike deler av programmet, beskrevet av navnet på variabellisten. Følgende globale variabellister er satt opp i programmet:

- **GVL**
  Inneholder variabler knyttet opp mot I/O, og variabler tilknyttet sekvensen.
- **GVL_alarm**
  Inneholder variabler knyttet opp mot alarmsystemet. Her deklareres alle alarmer som er implementert i programmet.
- **GVL_param**
  Inneholder parametere og grenseverdier knyttet opp mot alarmdeteksjon og aggregatstyring. De fleste variablene i denne listen er justerbare i HMI-en, og er deklarert som "RETAIN PERSISTENT" slik at verdier huskes ved strømbrudd.
- **GVL_SMS**
  Inneholder variabler knyttet opp mot SMS-kommunikasjonen, deriblant deklarering av telefonkatalog-array og sms-nivå på alarmer. Deler av denne er deklarert som "PERSISTENT" slik at verdier huskes ved strømbrudd.
- **GVL_colors**
  Variabelliste tilhørende HMI-en. Variablene i denne listen benyttes for fargetema i HMI-en.
- **GVL_persist**
  Remanente variabler definert andre steder i programmet må også defineres i denne listen. Med e!Cockpit er det mulig å raskt oppdatere denne listen, semi-automatisk.

### 4.2.3 Inn- og utgangssignaler

#### Lese av innganger

Basert på om sensorene montert på anlegget er digitale eller analoge, hentes verdiene fra henholdsvis AI- eller DI-kortene montert på PLS-en. Dette kan endres i innstillinger i HMI-en (kap. 4.6.7.11), avhengig av tilgjengelige sensorer. I kode 4.1 ser man at når analog sensor er valgt, vil variabelen for dieselnivå regnes ut i en funksjonsblokk (Scale_AI, kap. 4.2.4.4) basert på kalibreringen til sensor. Om sensor derimot er digital vil dieselnivået være invers av inngang (se linje 10 i kode 4.1), da slike sensorer oftest er montert som normalt lukket (NC). Som vist i koden er denne sensoren i dette tilfelle tilkoblet inngang 6 på PLS-ens digitale inngangskort.

```
1  // FUEL LEVEL
2  // Use either Analog signal or Digital Signal, based on one bool
3  IF GVL_param.xAI_fuel_level_sensor THEN
4   wDiesel_level_raw := g_piaAI_selector[GVL_param.iAI_index_fuel_level]^;
5     g_rDiesel_level := Scale_AI(rMeasured_value := WORD_TO_REAL(SHR(wDiesel_level_raw,3)),
6                rZero := GVL_param.rFuel_zero, // Can be set in HMI (Advanced Settings)
7                rURL  := GVL_param.rFuel_URL, // Can be set in HMI (Advanced Settings)
8                rX1   := 4096);
9  ELSE
10    g_xDiesel_level := NOT IoConfig_Globals_Mapping.xDI6;  // NC
11 END_IF
```

**Kode 4.1:** Eksempel på innhenting av verdier fra inngangskort

For innhenting av spenning- og frekvensverdier fra generator blir det brukt en egen modul på PLS-en (kap. 2.3.7). I e!COCKPIT finnes det en integrert funksjonsblokk for denne modulen, som gjør at man enkelt kan hente ut de måleverdiene man har bruk for. Blokken regner ut gjennomsnittsverdiene for spenning og frekvens fra de siste fem sekundene og gjør dem tilgjengelig for videre bruk. Hvis strømmåling er tilkoblet og aktivert i HMI-en, vil funksjonsblokkene også hente nåværende strøm, og ulike effektverdier. Det er i tillegg lagt til en skaleringsfaktor som oppskalerer strømmålingene, da den er nedskalert i strømtransformatoren for å kunne måles i PLS-en.

#### Skrive til utgang

Skriving til utganger er relativt enkelt. De boolske variablene som styrer utgangene blir styrt av sekvensen, men blir igjen overstyrt av nødstoppsignalet. Når utgangen blir høy, settes 24 VDC på, og releet koblet til blir aktivert. Hvordan dette gjøres vises i kode 4.2.

```
1  //Digital outputs - Write to output_modules
2  IF g_xEmergencystop_button THEN     //IF emergency stop, set all outputs LOW/FALSE
3    IoConfig_Globals_Mapping.xDO1 := FALSE;
4    IoConfig_Globals_Mapping.xDO2 := FALSE;
5    IoConfig_Globals_Mapping.xDO3 := FALSE;
6    IoConfig_Globals_Mapping.xDO4 := FALSE;
7  ELSE
8    IoConfig_Globals_Mapping.xDO1 := g_xLoad_contactor;   // Turns on load contactor
9    IoConfig_Globals_Mapping.xDO2 := g_xStarting_engine;  // Turns on start motor
10   IoConfig_Globals_Mapping.xDO3 := g_xMagnet;           // Fuel magnet
11   IoConfig_Globals_Mapping.xDO4 := g_xGlow;             // Glowingbefore start-attempt
12 END_IF
```

**Kode 4.2:** Skriv til utgangskort

### 4.2.4 Diverse programlogikk

#### 4.2.4.1 Driftstimer

For å vite når vedlikehold og lignende bør utføres på aggregatet er det viktig å loggføre antall timer det har vært i drift. Dette gjøres ved hjelp av et enkelt program som øker en variabel ved et fast intervall, som i vårt tilfelle. Det vil si at når denne økende variabelen har nådd 180 (20 sekunder x 180 = 1 time), så økes driftstimetelleren med én, og deretter nullstilles 20 sekunds-telleren. Dette vil bli gjort kontinuerlig, så lenge motoren er i drift. Det er også laget en teller for vedlikehold som fungerer på samme måte, bare at denne kan nullstilles etter utført vedlikehold. I tillegg blir denne brukt til å gi advarsel når telleren overskrider et gitt serviceintervall, som ofte er satt til 500 timer. Dette intervallet blir oftest oppgitt av motorleverandør, og kan stilles inn på vinduet "Motor Innstillinger" i brukergrensesnitt 4.35.

#### 4.2.4.2 Glødetid

Siden ideell glødetid er avhengig av motor, type glødeplugg og omgivelsestemperatur, er det lagt inn gode muligheter for å endre denne tiden i brukergrensesnittet (kap. 4.6.7.4. Hvis måling av omgivelsestemperatur er tilgjengelig, blir glødetiden regnet ut som en lineær funksjon av temperaturen ved bruk av interpolering (kap. 3.6) mellom to punkter. Denne funksjonen blir regnet ut ved at man oppgir ønsket glødetid for - 20 ℃ og 20 ℃. Om man derimot ikke har måling av temperatur, så fastsetter man en konstant glødetid direkte i brukergrensesnittet. Det er i tillegg satt grenser på glødetiden, om det eventuelt skulle være en feil på temperatursensor. Maks grense er satt til 45 sekunder og minimum på 2 sekunder.

#### 4.2.4.3 Temperaturoverstyring

Av og til kan det være ønskelig å starte og kjøre anlegget selv om temperaturalarm for kjølevæske er aktiv. Dette er mulig enten via SMS-systemet eller direkte via HMI. Man må da spesifisere hvor mange minutter alarmen skal overstyres. I det overstyringen aktiveres vil det kobles inn en forsinket utkobling, som er aktiv i overstyringstiden. Kjøreflagget (kap. 4.3.1) vil da ikke la seg deaktivere av temperaturalarmen i dette tidsrommet. Det vil fortsatt meldes om alarm, men alarmaksjonen fra denne alarmen vil være frakoblet. Oppdragsgiver sine APS-er står noen ganger i veldig krevende miljø, hvor eksempelvis snø og is kan tette for luftsirkulasjon og dermed hindre tilstrekkelig kjøling av motoren. Det kan da være nyttig å overstyre alarmen, men siden motoren kan ta skade kreves det innlogging som administrator, eller å ha SMS tilgang til anlegget.

#### 4.2.4.4 Valg og skalering av analoge innganger

Skalering for analoge innganger på kort 750-459 kan justeres i brukergrensesnittet under "Avanserte innstillinger" (kap. 4.49). Ved å gjøre skalering tilgjengelig i brukergrensesnittet kan nye følere med annen skalering monteres uten at programmet må gjøres om. Skaleringen foretas av en egendefinert funksjon kalt "Scale_AI", som vist på linje 2-5 i kode 4.3, og benytter følgende formel:

$$y_{skalert} = y \cdot (URL - zero)/x_1 + zero \tag{4.1}$$

der $y$ er den originale måleverdien, $x_1$ er oppløsningen på måleverdien, $URL$ ("Upper Range Limit") er øvre målegrense, og $zero$ er nullpunktet til målingen, eller nedre målegrense. Verdiene $URL$ og $zero$ er justerbare i brukergrensesnitt.

```
1  wOil_Raw_AI     := g_piaAI_selector[GVL_param.iAI_index_Oil]^;
2  g_rOil_pressure:= Scale_AI(rMeasured_value := WORD_TO_REAL(SHR(wOil_Raw_AI, 3)),
3                             rZero  := GVL_param.rOilpresure_zero, // Can be set in HMI
4                             rURL   := GVL_param.rOilpresure_span,  // Can be set in HMI
5                             rX1    := 4096);
```

**Kode 4.3:** Valg og skalering av analoge innganger

Valg av analoge innganger på modul 750-459 kan velges i brukergrensesnittet under "Avanserte innstillinger" side 3 (kap. 4.47). Her er det brukt en liste med pekere som vist i kode 4.4. Listen inneholder adressen til alle analoge innganger knyttet til kort 750-459. Første og siste indeks i listen peker til en variabel som er satt til null, og er brukt for deaktivering av innganger.

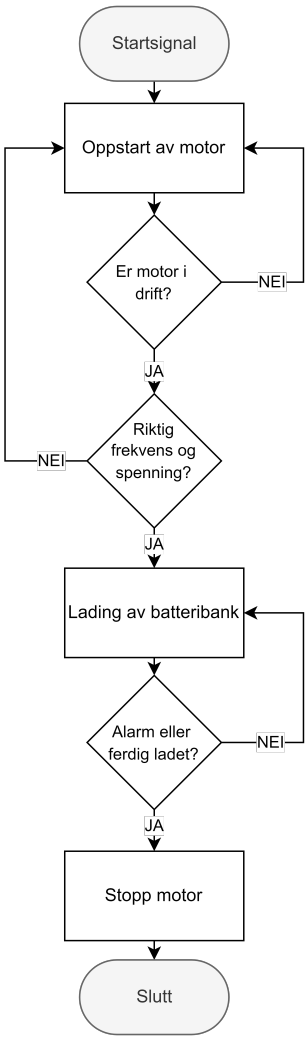
På linje én i kode 4.3 er et eksempel på hvordan programmet velger riktig inngang. Her er det "iAI_index_Oil" som er justerbar på brukergrensesnitt, og kan ha verdiene null til fire. Alle analoge innganger er tilknyttet en tilsvarende justerbar variabel.

```
1 g_iEmpty_element : INT := 0;
2 g_piaAI_selector : ARRAY[0..5] OF POINTER TO WORD := [ADR(g_iEmpty_element),
3                          ADR(IoConfig_Globals_Mapping.wAI1),
4                          ADR(IoConfig_Globals_Mapping.wAI2),
5                          ADR(IoConfig_Globals_Mapping.wAI3),
6                          ADR(IoConfig_Globals_Mapping.wAI4),
7                          ADR(g_iEmpty_element)]; // pointers needs to be initialized.
```

**Kode 4.4:** Valg av analoge innganger

## 4.3 Aggregat-sekvens

For å styre motoren og innkobling av last til generatoren er det brukt sekvensiell programmering i SFC (kap. 3.5.1.2). Dette er gjort for å få et oversiktlig program med tydelige steg for steg. SFC-programmet bruker aksjonsskript skrevet i strukturert tekst i steg hvor større biter med program må utføres. Dette er med på å splitte opp programmet og gjør det lettere å lese. Stegene er laget med logiske navn, og har spesifikke overgangskriterier som må oppfylles for å gå videre til neste steg. Figur 4.3 viser et forenklet flytskjema av logikken til styringen av aggregatet. Se vedlegg D for et mer detaljert sekvensskjema.




**Figur 4.3:** Aggregat flytskjema - Oppstart av motor, lading av batteri og stopp av motor. Se vedlegg D for et mer detaljert sekvensdiagram for aggregatet.

Sekvensen har et oppstartssteg der den venter på kjøresignal for å starte aggregatet. De påfølgende stegene blir brukt til oppstart av motoren med gløding, driftsmagnet og kjøring av elektrisk startmotor. Deretter skal det bekreftes ved hjelp av spesifikke sensorer, samt måling av frekvens og spenning fra generator at motoren har startet. Hvis motoren derimot ikke har startet, gjennomføres det et nytt startforsøk. Etter at motoren har startet får den gå en liten stund på tomgang for å komme, før energimåleren sjekker generatoren sin frekvens og spenning opp mot definerte grenseverdier.

I sekvensen blir det brukt mange forskjellige parametere, eksempelvis grense for godkjent generatorspenning. Disse er justerbare fra HMI-panelet (kap. 4.6.7.4).

Dersom alt er i orden når spenning og frekvens sjekkes, vil lasten til generatoren kobles inn og aggregatet begynner opplading av batteriene i APS-enheten. Så lenge aggregatet har signal om å lade og ingen kritiske alarmer er aktive, vil aggregatet lade batteriene. Under drift overvåkes systemet for å gi alarmer med advarsler og kritiske alarmer som fører til stopp av motoren.

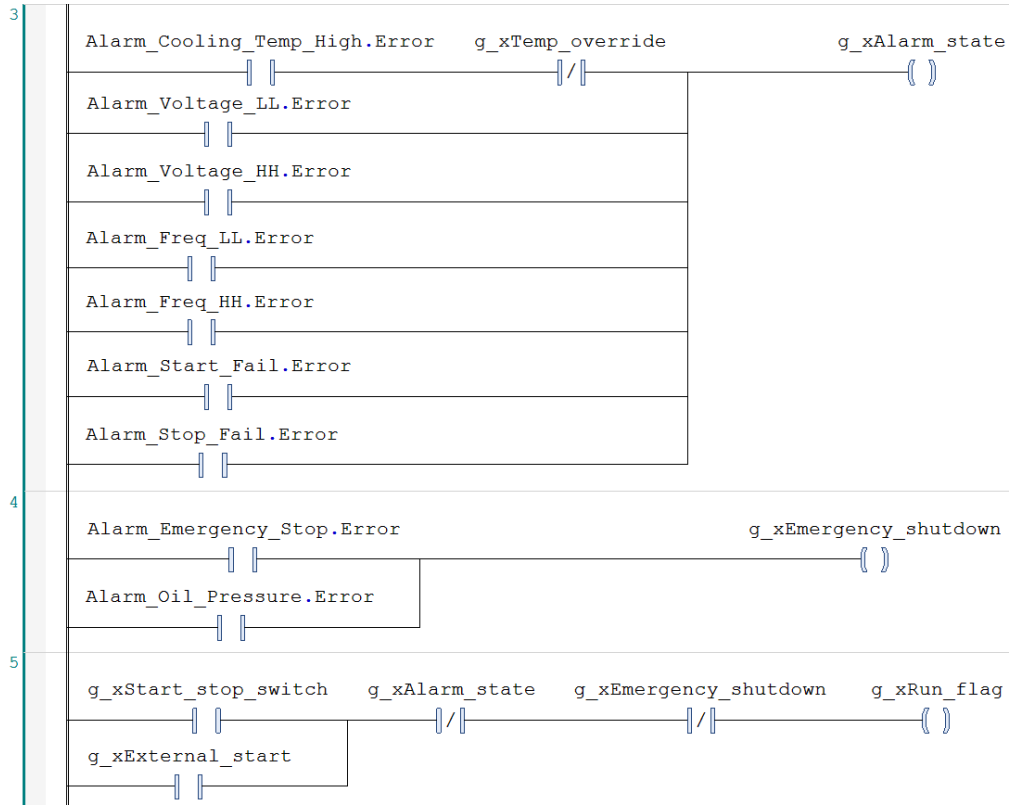
### 4.3.1 Kjøreflagg

Styringen av kjøreflagget (figur 4.4), til bruk i sekvensen for start og stopp av aggregatet, er programmert i ladderdiagram. Dette gjør det enkelt for vedlikeholdspersonell å overvåke programmet visuelt, spesielt under feilsøking. Logikken her er enkel og oversiktlig, med kun globale boolske variabler med både normalt åpne eller lukkede kontakter.

Alle alarmene her skal koble ut lastkontaktoren til generatoren umiddelbart og deretter stoppe motoren. Alarmene blir her delt i to grupper avhengig av hvilken type stopp som er ønskelig. Den første gruppen er de alarmene som setter variabelen "Alarm state" høy, og er de alarmene som fører til en normal stopp av motor. Først blir lastkontaktoren koblet ut, og deretter kjører motoren uten last en periode før man stenger av helt.

Den andre gruppen omhandler de mest kritiske alarmene som setter variabelen "Emergency shutdown" høy. Denne variabelen blir satt høy om oljetrykket går lavt en kort periode under normal drift, eller ved aktivering av nødstoppknappen. Når denne gruppen er aktiv, vil motoren stoppes umiddelbart ved at både lastkontaktor og driftsmagnet frakobles.

Alarmdeteksjon står beskrevet i større detalj i kapittel 4.4.6.




**Figur 4.4:** Ladderdiagram for kjøreflagg med alarmstatus og startsignal

Startsignalet for å starte aggregatet kommer fra en omformer som er koblet opp mot batteribanken, solcellelading, aggregatet og utgang til strømforbruk. Dette er et signal som ligger høyt så lenge batteribanken trenger lading eller forbruket er veldig høyt, ellers ligger signalet lavt. Det er i tillegg montert en bryter med egen strømtilførsel ved berøringsskjermen for å kunne overstyre oppstart, slik at aggregatet kjører kontinuerlig. Disse signalene kommer inn i parallell på DI1 og tilsvarer variabelen "g_xStart_stopp_switch" i figur 4.4. Det er også en variabel "g_xExternal_start" som er ubrukt, men ligger klar for bruk i skytjeneste eller lignende. Startsignalene er koblet i parallell og aktiverer kjøreflagget som er forriglet av alarmene slik at det ikke kan aktiveres ved aktiv alarm.

### 4.3.2 Oppstart motor

Som nevnt i kapittel 3.3 er det nødvendig å gløde motoren under oppstart. Når kjøreflagget går høyt, vil sekvensen først gå inn i glødesteget hvor glødesignalet fra PLS-en settes høyt. Steget er ferdig to sekunder før glødetiden er over.

Neste steg i sekvensen er å sette driftsmagneten høy slik at det åpnes opp for drivstoff før oppstart. I dette steget som varer i to sekunder er også glødingen aktiv før sekvensen går til neste steg, kjøring av startmotor. I stegene for gløding og magnet kjøres det handlingsskript for å kontrollere at oljetrykket er lavt. Om oljetrykket er høyt antas det at motoren er i gang, og det er dermed viktig at sekvensen ikke får gå videre til å kjøre startmotoren. For å unngå dette er det lagt til en forrigling med at motoren skal stoppes og oljetrykket skal være lavt før en ny start.

Starteren kjører fram til det blir høyt oljetrykk eller at frekvensen i generatoren er høy nok. Det er brukt to overgangsbetingelser med "OR"/eller for å sikre at startmotoren ikke kjører unødvendig lenge, og man unngår dermed slitasje. Hvis motoren ikke starter innen sekundene angitt av maks startertid, vil den forsøke et nytt startforsøk. Når oljetrykket eller frekvensen er høy nok i to etterfølgende kjøretidssykluser vil sekvensen gå videre til neste steg. Signalet må være høyt i to sykluser for å eliminere eventuell støy fra sensorene så ikke starteren stopper for tidlig ved en feil.

### 4.3.3 Nytt startforsøk

Det er ikke alltid at motoren greier å starte på første forsøk. Det er derfor lagt inn egne steg for å kontrollere ny oppstart av motoren, og telle antall forsøk. Et nytt startforsøk er definert som hver gang glødingen startes. Når motoren ikke starter innen startertidsgrensen, sjekkes det antall startforsøk og årsaken til nytt startforsøk lagres for visning i HMI-en og melding på SMS. Sekvensen går dermed til stopp av motor for å forsikre seg om at alt er slått av før en ny start. Deretter startes glødingen på nytt og nåværende antall startforsøk blir inkrementert. Når grensen for antall mislykkede startforsøk nås vil det gå en alarm som trenger å kvitteres før motoren kan starte på nytt. Denne grensen er satt til fem som standard, men er justerbar.

Hver gang kjøreflagget går lavt, blir antall startforsøk tilbakestilt. Antall startforsøk fra de siste 20 oppstartene blir loggført i en tabell. Denne tabellen kan være nyttig å se på hvis det har oppstått en alarm. Årsaken til et nytt startforsøk blir også lagret hver gang alarmen "start feilet" blir aktivert for visualisering i alarmtabellen. Årsaken til de 10 siste startfeilene er tilgjengelig.

### 4.3.4 Innkobling av lastkontaktor

For å kunne starte lading av batteribanken må det bekreftes at motoren har startet, og at generatoren har riktig frekvens og spenning. Når dette er bekreftet, vil sekvensen gå videre til et steg for å vente på at motoren skal varme seg opp. Denne ventetiden er justerbar i HMI-en.

Når ventetiden er over sjekker sekvensen en boolsk variabel som sier om frekvens og spenning er innenfor minimum- og maksimum-grensene, som standard er satt til 43-57 HZ og 205-255 V. Disse grensene er også justerbare. Dersom variabelen er "usann" vil motoren stoppes og iverksette et nytt startforsøk. Om variabelen er sann, går sekvensen direkte videre til å aktivere lastkontaktor, som i praksis betyr lading av batteribank. Aggregatet er nå i full drift og her kommer det til å bli stående helt frem til startsignalet blir lavt eller ved at en alarm deaktiverer kjøreflagget.

Når aggregatet er i full drift med lasten innkoblet overvåkes nå frekvens og spenning kontinuerlig. Den kontinuerlige overvåkingen gjør at man kan koble direkte til utstyr, og samtidig være trygg på at utstyret ikke blir skadet ved feil verdier på spenning og frekvens. Det er derfor viktig at lastkontaktoren kobles ut umiddelbart ved slike feilsituasjoner.

### 4.3.5 Stopp av motor

I en normal driftssituasjon uten aktive alarmer vil omformeren sette startsignalet lavt når det ikke lenger kreves lading. Lasten vil da kobles ut, og motoren får gå på tomgang i et visst antall sekunder før magneten kobles ut og motoren stopper. Om kjøreflagget går høyt når motoren kjøles ned på tomgang, vil den umiddelbart hoppe tilbake i sekvensen for sjekk av spenning og frekvens, før den kobler inn igjen lastkontaktor.

Hvis noe som trekker mye strøm fra strømforsyningsanlegget blir koblet ut i en kort periode, og deretter koblet inn igjen, kan det skje at kjøreflagget går lavt en liten stund. Om dette skulle skje så slipper motoren å stoppe og starte unødvendig, om flagget er borte i mindre enn angitt nedkjølingstid. I tilfellet hvor alarmen for nødstopp eller lavt oljetrykk blir aktiv så vil sekvensen hoppe over nedkjølings-steget og dermed stoppe motoren umiddelbart for å hindre skade på motor eller annet materiell.

Den siste overgangsbetingelsen i sekvensen skal forsikre at motoren faktisk har stoppet før sekvensen går tilbake til oppstartssteget. Betingelsen er at oljetrykket har blitt lavt,

noe som normalt skjer ganske raskt etter at drivstofftilførselen har blitt kuttet. I samråd med oppdragsgiver har det blitt laget en aksjon hvor man prøver en ny utkobling av driftsmagnet hvis oljetrykket fortsatt er høyt etter 30 sekunder. Grunnen til høyt oljetrykk kan komme fra at driftsmagneten ikke greier å tette for drivstoff, som fører til at motoren fortsetter å kjøre. Det kan også være en feil med signalet fra oljetrykksgiveren, som det er viktig å få oppdaget og rettet opp i.

Uavhengig av hva feilen er, så vil sekvensen prøve å koble inn og ut driftsmagneten gjentatte ganger, hvor den deretter sjekker om oljetrykket endres. Om oljetrykket i motor fortsatt er høyt vil det aktiveres en alarm som varsler at motoren ikke har klart å stoppe og at oljetrykket var høyt. Denne alarmen må kvitteres før aggregatet kan starte på nytt.

### 4.3.6 Nødstopp

Maskindirektivet stiller krav til implementering av nødstopp på maskiner. I dette tilfellet er det brukt stoppkategori 0: "Stopp ved umiddelbart å bryte krafttilførselen til drivinnretningene (dvs. et ukontrollert stopp)" [19]. Dette tilsvarer den strengeste stoppkategorien.

Utgangsreleene tilkoblet PLS-en kan brennes fast, eller det kan oppstå feil internt i programvaren. Det er derfor ikke nok å kun ha nødstopp som en inngang på PLS-en. Nødstopp er derfor i tillegg koblet som forsyning til utgangsreleer/kontaktorer. Denne løsningen sørger for at hverken driftsmagnet, gløding, starter eller lastkontaktor aktiveres, og at alt dermed stopper umiddelbart. For oversikt over oppkoblingen, se koblingsskjema i vedlegg B.

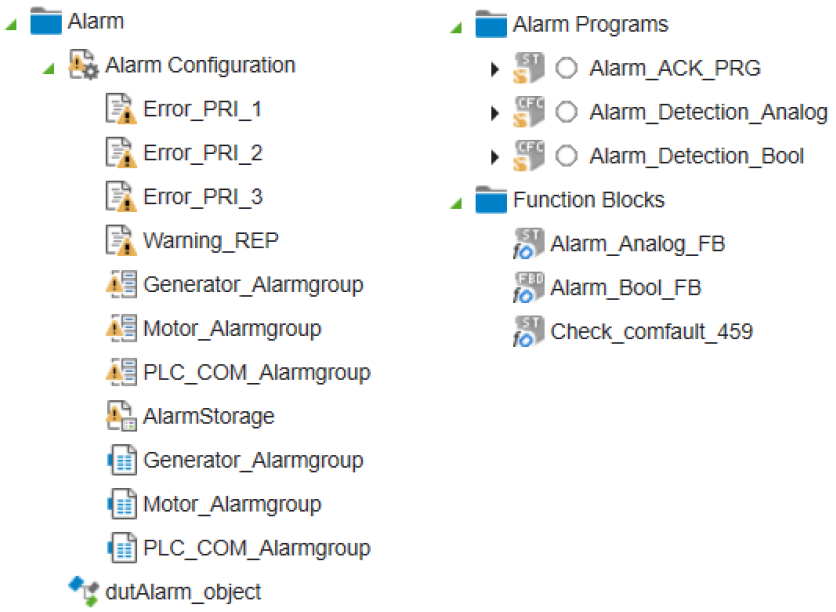
## 4.4 Alarmsystem

For å overvåke styresystemet og sørge for en sikker drift av dieselaggregatet, er det implementert et komplekst alarmsystem som består av flere sammenkoblede deler. De neste delkapitlene går inn på hva slags forskjellige deler alarmsystemet består av, og hvordan det fungerer.

### 4.4.1 Struktur

Det komplette alarmsystemet er bygget opp av en blanding av ferdigimplementerte funksjonaliteter i e!COCKPIT, og en rekke egenkomponerte programmer. Systemet består av programmer for å konfigurere alarmer, detektere disse alarmene og å deretter å sørge for at alarmene blir visualisert i brukergrensesnittet.

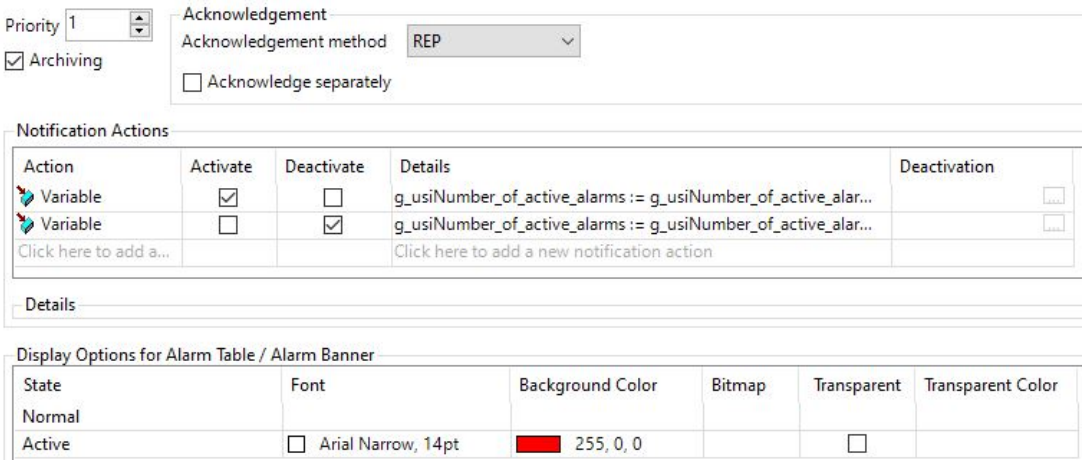
Innholdet i valgt alarmkonfigurasjon består av fire alarmklasser, tre alarmgrupper med tilhørende tekstlister og en alarmdatabase for lagring av alarmer. De egenkomponerte programmene består av et kvitteringsprogram, et analogt alarmdeteksjonsprogram og et digitalt alarmdeteksjonsprogram. Det er også implementert tre funksjonsblokker som blir brukt til alarmdeteksjon.




**Figur 4.5:** Alarmsystemstruktur

### 4.4.2 Alarmklasser

Det er laget fire alarmklasser for systemet: tre feil-alarmer (rød) med ulik prioritet (1-3) og én advarsel-alarm (oransje) med lavest prioritet (10). Alle fire klassene blir lagret i Alarm Storage som gjør det mulig å visualisere tidligere alarmer i historikken i brukergrensesnittet. Aksjonsfunksjonaliteten i alarmklassene brukes til å telle antall aktive alarmer, som igjen er koblet til alarmlampen i HMI-programmet. Et eksempel på hvordan alarmklassene er satt opp er vist i figur 4.6.




**Figur 4.6:** Alarmklasseoppsettet for Error_PRI_1 (Feilalarm med høyest prioritet).

### 4.4.3 Alarmgrupper

Det er laget tre forskjellige alarmgrupper: En for motoren, en for generatoren og en for selve PLS-en. Motor-alarmgruppen inneholder alle alarmer som er koblet til drift og start/stopp av motoren, inkludert hoved- og startbatteri. Generator-alarmgruppen inneholder høy/lav-alarmer for spenningen og frekvens som generatoren produserer. PLS-alarmgruppen tar for seg alarmer knyttet til kommunikasjon inn- og ut av PLS-en, inkludert feil i SMS-systemet. Alle alarmene som er lagt inn i konfiguratoren i dette prosjektet observeres som digitale bits (kap. 4.4.6). Et eksempel på hvordan alarmgruppene er satt opp er vist i figur 4.7

| ID | Observation Type | Details | Class | Message | Higher Prio. Alarm | |
|---|---|---|---|---|---|---|
| 100 | Digital | GVL_alarm.Alarm_Freq_LL.Error = TRUE | Error_PRI_2 | [100] Generator frekvens er for lav. | | |
| 101 | Digital | GVL_alarm.Alarm_Freq_L.Error = TRUE | Warning_REP | [101] Generator frekvens er lav | 100 | GVL_alarm.Alarm_Freq_LL.Error = TRUE |
| 102 | Digital | GVL_alarm.Alarm_Freq_H.Error = TRUE | Warning_REP | [102] Generator frekvens er høy | 103 | GVL_alarm.Alarm_Freq_HH.Error = TRUE |
| 103 | Digital | GVL_alarm.Alarm_Freq_HH.Error = TRUE | Error_PRI_2 | [103] Generator frekvens er for høy | | |
| 104 | Digital | GVL_alarm.Alarm_Voltage_LL.Error = TRUE | Error_PRI_2 | [104] Generator spenning er for lav | | |
| 105 | Digital | GVL_alarm.Alarm_Voltage_L.Error = TRUE | Warning_REP | [105] Generator spenning er lav | 104 | GVL_alarm.Alarm_Voltage_LL.Error = TRUE |
| 106 | Digital | GVL_alarm.Alarm_Voltage_H.Error = TRUE | Warning_REP | [106] Generator spenning er høy | 107 | GVL_alarm.Alarm_Voltage_HH.Error = TRUE |
| 107 | Digital | GVL_alarm.Alarm_Voltage_HH.Error = TRUE | Error_PRI_2 | [107] Generator spenning er for høy | | |

**Figur 4.7:** Oppsettet av generator-alarmgruppen

### 4.4.4 Alarmobjekt

I programmet er alle alarmer deklarert som en instans av et egendefinert alarmobjekt. Hensikten bak dette objektet er å samle alle relevante variabler tilknyttet hver alarm i et og samme objekt, noe som skaper god struktur. Alarmobjektet inneholder en unik alarm-ID, en feilbit som definerer om alarmen er aktiv, en triggerbit som brukes for manuell testing av alarm via HMI (kap. 4.6.7.7), og en kvitteringsbit som brukes til å kvittere alarmer. I tillegg inneholder objekter to variabler knyttet opp mot SMS-systemet (kap. 4.5). Variabelen "piSMS_lvl" brukes av SMS-systemet til å filtrere hvilke kontakter som skal varsles av hvilke alarmer (kap. 4.6.7.9). Variabelen "psAlarm_msg" blir også kun brukt av SMS-systemet og er en peker/pointer som peker til alarmteksten til objektet. Denne teksten ligger i alarmkonfiguratoren, og pekeren blir initialisert i SMS-systemet.

```
TYPE dutAlarm_object :
STRUCT
  iID            : INT;
  piSMS_lvl      : POINTER TO INT;
  psAlarm_msg    : POINTER TO STRING;
  Error          : BOOL;
  Trigger_error  : BOOL;
  Acknowledged   : BOOL;
END_STRUCT
END_TYPE
```

**Kode 4.5:** Alarmobjekt

Når en alarm-instans blir deklarert blir flere av variablene, deriblant ID, initialisert til bestemte verdier. Et eksempel på hvordan dette gjøres er illustrert i kode 4.6.

```
// Generator
(* iID on alarmobject needs to match ID in Alarmgroup. Each objects ID is unique! *)
Alarm_Freq_LL : dutAlarm_object:=(iID:=100, piSMS_lvl:=ADR(g_iSMS_lvl_Alarm_Freq_LL));
Alarm_Freq_L  : dutAlarm_object:=(iID:=101, piSMS_lvl:=ADR(g_iSMS_lvl_Alarm_Freq_L));
Alarm_Freq_H  : dutAlarm_object:=(iID:=102, piSMS_lvl:=ADR(g_iSMS_lvl_Alarm_Freq_H));
Alarm_Freq_HH : dutAlarm_object:=(iID:=103, piSMS_lvl:=ADR(g_iSMS_lvl_Alarm_Freq_HH));

Alarm_Voltage_LL:dutAlarm_object:=(iID:=104,piSMS_lvl:=ADR(g_iSMS_lvl_Alarm_Voltage_LL));
Alarm_Voltage_L :dutAlarm_object:=(iID:=105,piSMS_lvl:=ADR(g_iSMS_lvl_Alarm_Voltage_L));
Alarm_Voltage_H :dutAlarm_object:=(iID:=106,piSMS_lvl:=ADR(g_iSMS_lvl_Alarm_Voltage_H));
Alarm_Voltage_HH:dutAlarm_object:=(iID:=107,piSMS_lvl:=ADR(g_iSMS_lvl_Alarm_Voltage_HH));
```

**Kode 4.6:** Deklarering av alarmer

Adressene til alle alarmobjekter er samlet i en tabell/array (se kode 4.7). Dette gjør programmet ryddigere der man må lete frem en alarm eller behandle flere alarmer i samme syklus. Denne listen blir blant annet brukt i kvitteringsprogrammet og SMS-systemet.

```
g_paAlarm_objects : ARRAY[1..29] OF POINTER TO dutAlarm_object := [
        ADR(Alarm_Freq_LL),             // #1
        ADR(Alarm_Freq_L),              // #2
        ADR(Alarm_Freq_H),              // #3
        . . .
        ADR(Alarm_com_coolant),         // #28
        ADR(Alarm_com_oilpressure)];    // #29
```

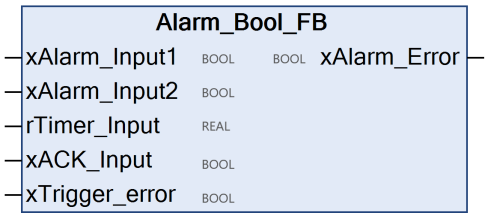
**Kode 4.7:** Array/tabell med pointers/pekere til alarmobjekter

### 4.4.5 Funksjonsblokker

For å detektere alarmer er det blitt laget to funksjonsblokker: En blokk for alarmdeteksjon av digitale verdier, og en for deteksjon av analoge verdier. I tillegg er det laget en egen funksjonsblokk for å detektere kommunikasjonsfeil. For mer detaljert informasjon og kildekode, se vedlegg F. Videre kommer en forklaring på hvordan disse blokkene fungerer.

#### 4.4.5.1 FB: Digital alarmdeteksjon

For å detektere alarm på digitale målesignaler er det laget en funksjonsblokk kalt "Alarm_Bool_FB". Funksjonsblokken har totalt fem innganger og en utgang, der utgangen tilsvarer selve alarmbiten for feil. De to første inngangene er ment for målesignalet og en tilleggsbetingelse, der begge inngangene må være høye for å kunne aktivere utgangen. Om man kun har målesignal som betingelse, skriver man "True" på inngang to. Funksjonsblokken har i tillegg en forsinkelse/timer-inngang, der det settes antall sekunder de to første inngangene i blokken må være høye for at utgangen på blokken skal settes høy.

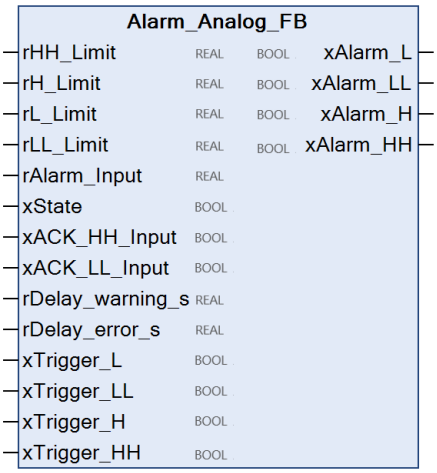



**Figur 4.8:** Funksjonsblokk for digital alarmdeteksjon.

For å kvittere alarmen på utgangen er det lagt til en kvitteringsinngang. Blokken kvitterer alarmen når denne inngangen er satt høy, hvis minimum en av de to inngangssignalene har gått lav. Dette tilsvarer at alarmen ikke er stående. Den siste inngangen er implementert for testing, der blokken setter utgangen høy hvis trigger-inngangen er satt høy. Denne inngangen kan bli brukt til testing av alarmaksjoner, og kan styres fra vindu: "Alarmtest" i brukergrensesnittet (kap. 4.6.7.7).

#### 4.4.5.2 FB: Analog alarmdeteksjon

For å detektere alarm på analoge målesignaler er det laget en mer komplisert funksjonsblokk kalt "Alarm_Analog_FB". Denne funksjonsblokken har totalt 14 innganger og 4 utganger, der de 4 utgangene er selve alarmbitene for advarsel (L og H) og feil (LL og HH). De 4 første inngangene er for å sette alarmgrensene for henholdsvis høy/lav-alarmene, deretter kommer inngangen for målesignalet. Det er også implementert en tilstands-inngang som fungerer som et kriterium for at utgangene skal kunne bli skrevet høy. Denne må settes "True" om blokken alltid skal være aktiv. For å kvittere feil-alarmene (LL og HH) på utgangen er det lagt til to kvitteringsinnganger som kvitterer når inngangene er satt høy, om det ikke er stående alarm.

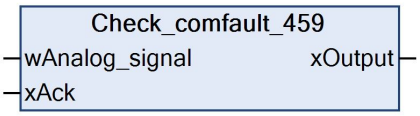



**Figur 4.9:** Funksjonsblokk for analog alarmdeteksjon

I tillegg er det implementert 2 innganger for delay/forsinkelse av både advarsel- og feilalarmene. Disse inngangene er laget slik at en L/H-alarm ikke vil bli aktiv før kriteriene har vært høye kontinuerlig i x-antall sekunder. Det samme gjelder for LL/HH-alarmene. De fire siste inngangene er implementert for testing, der blokken setter alarmutganger høye hvis den tilsvarende trigger-inngangen er satt høy.

#### 4.4.5.3 FB: Kommunikasjonsfeil

I sammenheng med å detektere kommunikasjonsfeil er det implementert en enkel funksjonsblokk. Denne funksjonsblokken har to innganger og én utgang, der utgangen tilsvarer selve alarmbiten for feil. Den første inngangen tar inn et dataord, hvor blokken sjekker de to første bitene i ordet (LSB). Om begge bitene er lik én i over to sekunder, vil utgangen settes høy. Den andre inngangen på blokken er for å kvittere feilen, hvor utgangen blir satt lav hvis kvitteringsinngangen er satt høy og alarmen ikke er stående.

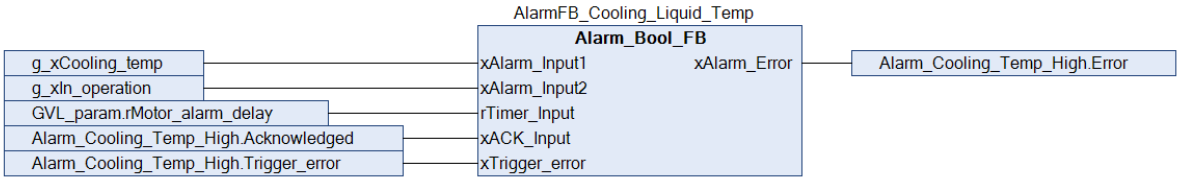



**Figur 4.10:** Funksjonsblokk for deteksjon av kommunikasjonsfeil

### 4.4.6 Alarmdeteksjon

#### 4.4.6.1 Digital alarmdeteksjon

Ved bruk av funksjonsblokken for detektering av digitale signaler, kan man eksempelvis se på overvåking av temperaturen i kjølevæsken. Dette er en kritisk alarm som vil føre til stopp av anlegg, da motor kan ta skade ved videre drift. Figur 4.11 viser at det er lagt inn to variabler på de to første inngangene, "g_xCooling_temp" og "g_xIn_operation". Disse tilsvarer kriteriene for at alarmen skal kunne aktiveres. Deretter er det lagt inn en parameter for forsinket innkobling av alarm, som er hentet fra variabellisten "GVL_param". Når de to første inngangene er sammenhengende aktive i mer enn den gitte tidsforsinkelsen vil alarmen bli satt høy. Kvitteringsinngangen må så aktiveres for å nullstille alarmen.

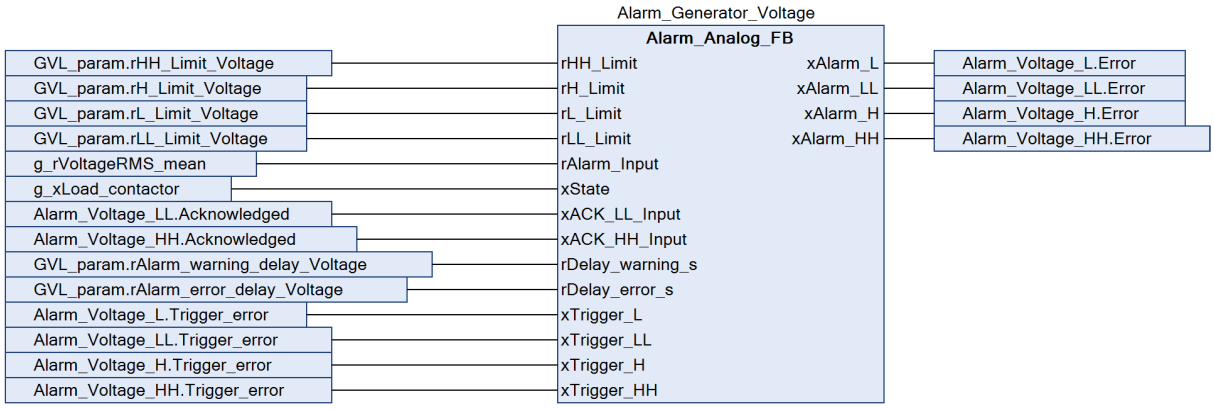



**Figur 4.11:** Eksempel på alarm deteksjon med boolske innganger.

Eksempel på andre signaler som blir brukt av denne funksjonsblokken er oljetrykk, nødstopp, dynamo og dieselnivå. Enkelte av disse signalene kan på ulike anlegg ha enten analog eller digital sensor, og man må da inn i brukergrensesnittet og velge hvilke type sensorer man har, og justere grensene deretter.

#### 4.4.6.2 Analog alarmdeteksjon

Ved bruk av funksjonsblokken for detektering av analoge verdier, kan man eksempelvis se på overvåking av generatorspenning. Blokken henter fire verdier for de ulike alarmgrensene i "GVL_param", i tillegg til alarmsignalet som i dette tilfellet er "g_xVoltageRMS_mean" (den målte spenningen fra generatoren). For at denne alarmen skal gå høy, må lastkontaktoren i anlegget være aktiv, som i PLS-en er lagret som "g_xLoad_contactor". Er kriteriene for alarm oppfylt, for eksempel at målt spenning ligger under lav-lav grensen i en periode lenger en forsinkelsestiden, vil gitte utganger gå høy. Figur 4.12 viser hvordan dette ser ut i programmet.




**Figur 4.12:** Eksempel på alarm deteksjon med grense-innganger.

Eksempel på andre signaler som blir overvåket ved bruk av denne blokken kan være dieselnivå, kjølevæsketemperatur, oljetrykk, spenning på start- og hovedbatteri og generatorfrekvens. Igjen er dette avhengig av hvilke sensorer man har til rådighet.

#### 4.4.6.3 Kommunikasjonsfeil

For detektering av kommunikasjonsfeil med de ulike sensorene bruker vi en egen funksjonsblokk, vist i kode 4.8. Om sensoren er aktiv, og man ønsker å motta alarmer på kommunikasjonsfeil, vil denne koden kjøre. Signalet som er hentet fra analog inngangskortet (kap. 2.3.5), inneholder diverse avviksbits som denne funksjonen bruker. Om de to minst signifikante bitene (altså bitene i dataordet som er mest til høyre) er høye, tyder dette at målesignalet er utenfor måleområdet, og nåværende måling derav ikke er gyldig. Det vil da bli aktivert en alarm.

```
// 0 ... 10V card (Product: 750 459)
// Only detect error if enabled in HMI
IF GVL_param.xAI_fuel_level_com_fault_check AND GVL_param.xAI_fuel_level_sensor THEN
    //Check for wirebreak diesel_lvl
    Check_comfault_1(wAnalog_signal := Get_Inputs.wDiesel_level_raw,
          xAck        := TRUE,
          xOutput     => Alarm_com_diesel_lvl.Error);
  IF Alarm_com_diesel_lvl.Trigger_error THEN
    Alarm_com_diesel_lvl.Error := TRUE;
  END_IF
END_IF
```

**Kode 4.8:** Deteksjon av kommunikasjonsfeil

#### 4.4.6.4 Strømbrudd

Detekteringen av strømbrudd er laget i en kort kodesnutt, vist i kode 4.9. Programmet tar utgangspunkt i to boolske variabler. Første gang PLS-en startes vil begge variablene settes høye. Den ene variabelen blir lagret ved strømbrudd, imens den andre forsvinner i takt med strømmen på PLS-en. Hvis det oppstår strømbrudd og strømmen kommer tilbake, vil man oppdage at den ene variabelen er lav, og man kan dermed fastslå at et

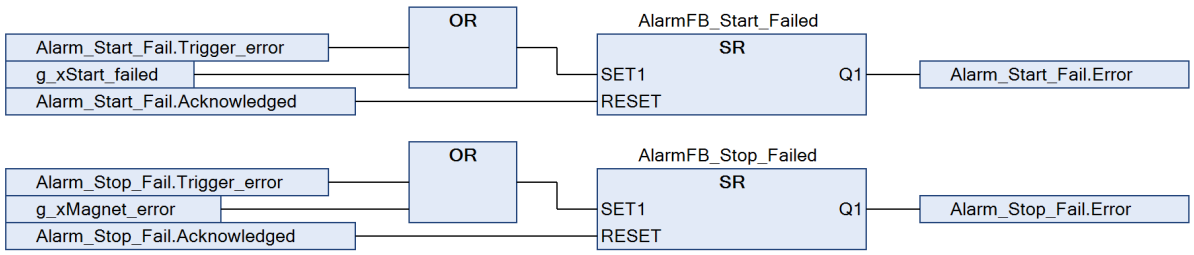
strømbrudd har inntruffet. Det vil da i koden aktiveres en alarm, og de to variablene blir igjen satt høye.

```
IF NOT xFirst_run_done THEN    //Code runs first run(not after power failures/warm reset)
  xFirst_run_done := TRUE;
  xInit_flag := TRUE;          //Sets init flag HIGH, resets when power OFF
ELSIF NOT xInit_flag THEN      //Code runs when Power had been away and comes back
  Alarm_power_loss.Error := TRUE;
  xInit_flag := TRUE;          //Sets init flag HIGH, resets when power OFF
END_IF
```

**Kode 4.9:** Deteksjon av strømbrudd

#### 4.4.6.5 Start- og stoppfeil

For å detektere feil i løpet av oppstart eller stopp av motoren, er det benyttet en enkel set/reset-blokk (set-dominans). Variablene som styrer alarmen til henholdsvis start/stopp blir satt direkte i handlingsskript i sekvensen. Alarmene kvitteres ved at kvitteringsbiten til start/stopp-alarmen er tilkoblet reset-inngangen på blokken.




**Figur 4.13:** Alarmdeteksjon av start- og stoppfeil

### 4.4.7 Alarmlampe

Øverst i toppmenyen på brukergrensesnittet vises en alarmlampe ved stående alarm. Lampens utseende og blinking avhenger av alvorlighetsgraden til alarmen og om alarmen er kvittert, som vist i henhold til figur 4.14.

| Alarm | Alarm Kvittert | Advarsel | Advarsel Kvittert | | |
|---|---|---|---|---|---|
| 1 | 0 | | | Blinker | |
| 0 | | 1 | 0 | Blinker | |
| 1 | 1 | | | Fast | |
| 0 | | 1 | 1 | Fast | |
| 0 | | 0 | | AV | |

**Figur 4.14:** Alarmlampe logikk. 1 = Sann, 0 = Usann, tom = likegyldig

### 4.4.8 Kvittering av alarmer

Etter en alarm har blitt aktiv, er det flere av disse som må kvitteres av personell. Dette er for å sikre at anlegget ikke skal fortsette som vanlig om en kritisk feil har oppstått. For å kvittere alarmene er det blitt laget et kvitteringsskript som kalles "Alarm_ACK_PRG" som kaller på tre egendefinerte aksjoner. Disse benytter funksjonalitetene knyttet til alarmkonfiguratoren. Kode 4.10 viser hovedprogrammet til kvitteringen.

```
// Ack all
IF xAck_all THEN
  ACK_all()       ;//Program that ack's all active alarms
  xAck_all := FALSE;
END_IF

// Ack selected
IF xAck_sel THEN
  ACK_sel();      //Program that ack's just the selected alarm
  xAck_sel := FALSE;
END_IF;

// Reset Ack-bit if necessarry and update alarmobject on topmenu (HMI)
```

**Kode 4.10:** Skript for å kvittere alarmer

#### 4.4.8.1 Kvitter valgt alarm

Denne aksjonen er brukt for å kvittere enkeltalarmer hver for seg. Alarmtabellen i brukergrensesnittet er knyttet opp mot to variabler kalt "AlarmGrp" og "AlarmID", som lagrer alarmgruppen og ID-en til valgt alarm i alarmtabellen i brukergrensesnittet. I det man kvitterer valgt alarm, vil dette skriptet sjekke hvilken alarmgruppe alarmen tilhører og deretter kvittere alarmen med ID tilsvarende "AlarmID".

```
//** ACK_sel() **
// Variable Alarmgrp and AlarmID are linked to the currently selected alarm on HMI.
IF AlarmGrp = 'Generator_Alarmgroup' THEN
  CASE AlarmID OF //Case to identify Alarm ID
    0: Alarm_Freq_LL.Acknowledged   :=TRUE;
    1: Alarm_Freq_L.Acknowledged    :=TRUE;
    2: Alarm_Freq_H.Acknowledged    :=TRUE;
    ...
  END_CASE
ELSIF AlarmGrp = 'Motor_Alarmgroup' THEN
  CASE AlarmID OF
    0: Alarm_Cooling_Temp_High.Acknowledged :=TRUE;
    1: Alarm_Diesel_L.Acknowledged          :=TRUE;
    2: Alarm_Diesel_LL.Acknowledged         :=TRUE;
    ...
  END_CASE
...
Alarmbanner_prg();  // Updates the state of the alarmbanner
```

**Kode 4.11:** Utdrag av kode som kvitterer valgt alarm

Aksjonen som kalles på nederst i kode 4.11 styrer blinking av alarmikonet i toppmenyen i brukergrensesnittet. Hvis det er en aktiv alarm som ikke er kvittert, vil ikonet blinke. Om alarmen blir kvittert, men fortsatt er stående, vil alarmikonet lyse konstant, uten blinking.

```
// Update blinker
g_xBlink_alarm_icon := FALSE;
FOR ii:=LOWER_BOUND(g_paAlarm_objects,1) TO UPPER_BOUND(g_paAlarm_objects,1) DO
  IF g_paAlarm_objects[ii] <> 0 THEN  // NULL-pointer check
    IF g_paAlarm_objects[ii]^.Error AND NOT g_paAlarm_objects[ii]^.Acknowledged THEN
      g_xBlink_alarm_icon := TRUE;  // Turns blinking back on if theres still active alarms in the system
    END_IF
  END_IF
END_FOR
```

**Kode 4.12:** Skript for styring av alarmikon i HMI

#### 4.4.8.2 Kvitter alle alarmer

Denne aksjonen er brukt for å kvittere alle alarmer. En "FOR-loop" iterer gjennom alle alarmene ved bruk av en peker til alarmobjektene. Når alarmene er kvittert, blir alarmikonet i HMI-en satt til å ikke lenger blinke.

```
//** ACK_all() **
// Iterates through all alarms and ack's
FOR i:=LOWER_BOUND(g_paAlarm_objects,1) TO UPPER_BOUND(g_paAlarm_objects,1) DO
  g_paAlarm_objects[i]^.Acknowledged := TRUE;
END_FOR
// The alarm icon should stop blinking after acking
g_xBlink_alarm_icon := FALSE;
```

**Kode 4.13:** Kvitterer alle alarmer

#### 4.4.8.3 Nullstill kvittering

Ved trigging av ny alarm må kvitteringsbittet gå lavt. Det er derfor laget et skript som tilbakestiller alarmobjektets kvitteringsbit når samme alarmobjekts feilbit går høy.

Denne aksjonen brukes for å tilbakestille kvitteringsbit, og i tillegg oppdatere alarmikonet. Koden itererer over alle alarmobjekter og oppdaterer kvitteringsbiten ved positiv flanke. Legg merke til at flanketrigging er implementert manuelt i strukturtekst. Dette er gjort på linje 3 og 4 i kode 4.14.

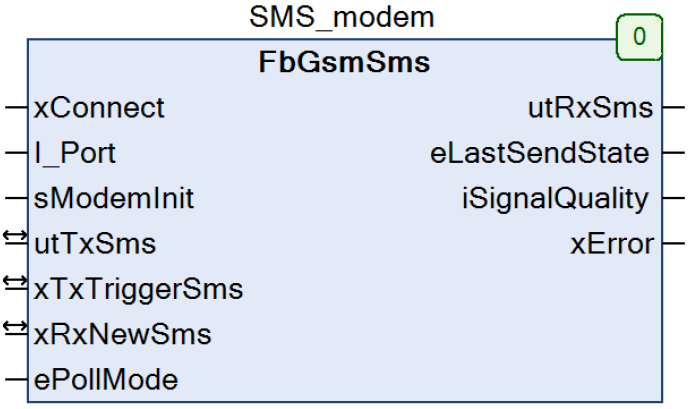
```
1  FOR i:=LOWER_BOUND(g_paAlarm_objects,1) TO UPPER_BOUND(g_paAlarm_objects,1) DO
2    // Check errors (alarms)
3    IF g_paAlarm_objects[i]^.Error <> xaPrev[i] THEN
4      xaPrev[i] := g_paAlarm_objects[i]^.Error;
5      IF xaPrev[i] THEN   // Rising edge
6        g_xBlink_alarm_icon := TRUE;  // Turn on blinking Alarmicon on HMI
7        g_paAlarm_objects[i]^.Acknowledged := FALSE;  // Reset ack
8      END_IF
9    END_IF
10 END_FOR
```

**Kode 4.14:** Nullstiller alle kvitteringer

## 4.5 SMS-system

Når en alarm blir aktiv er det viktig at relevant personell blir varslet om gjeldende alarmsituasjon. Anleggene Getek leverer er sjeldent bemannet, og lokal varsling på HMI-en eller via alarmlamper er derfor ikke nok. Varsling over SMS løser dette problemet.

Programvaren e!COCKPIT er utstyrt med et eget bibliotek for SMS-kommunikasjon: "WagoAppSerial_Sms". Dette prosjektet bruker blokken "FbGsmSms" fra dette biblioteket. Funksjonsblokken må ha tilgang til kontrollerens modem og vite mottakers telefonnummer, og hvilken beskjed som skal sendes. Når en ny SMS mottas av modemet går pinne "xRxNewSms" høy. Denne er av type "IN_OUT" og programmet på utsiden av blokken må sette denne lav igjen før en ny SMS kan mottas. Når en ny SMS skal sendes setter programmet på utsiden av blokken pinnen "xTxTriggerSms" høy. Funksjonsblokken vil resette pinnen tilbake til lav når SMS-en er blitt sendt. Begge sider, sending og mottak, er utstyrt med hver sin buffer slik at kvittering kan skje så raskt som mulig. Det blir dermed sikret at ingen meldinger blir overskrevet. Bufferen er programmert på utsiden av blokka og er en del av programvaren rundt.

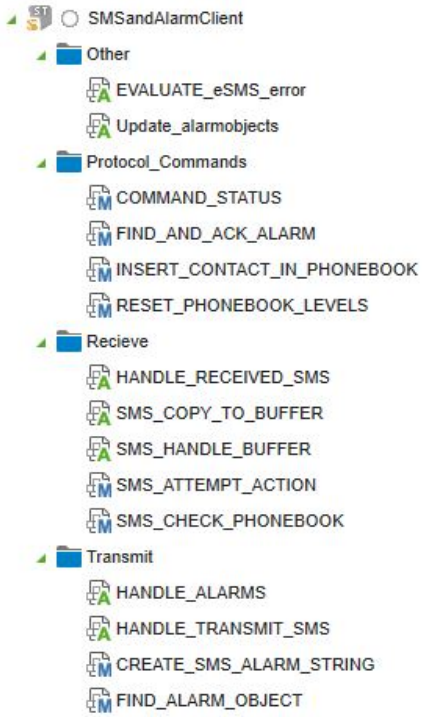



**Figur 4.15:** FB: Funksjonsblokk for SMS-modemet

SMS-systemet gjør det mulig å oppgi inntil 10 kontakter/telefonnummer som kan varsles til. Dette gjøres i "SMS kontakter" i HMI-en (kap. 4.6.7.8). Hver kontakt får angitt et nivå, som styrer hvilke varslinger man mottar. Man vil motta SMS-varslinger for alarmer med nivå mindre og lik sitt eget nivå, med unntak fra nivå 0. Alarmobjekter med nivå 0 har deaktivert SMS-varsling. Kontakter med nivå 0 har fortsatt mulighet til å sende kommandoer inn til systemet. Nivåene til hver alarm kan endres i HMI-en (kap. 4.6.7.9). De viktigste alarmene har nivå 1 og alle kontakter over nivå 0 vil motta disse. Uansett valgt SMS-nivå på en alarm, vil den varsles lokalt i HMI-en.

### 4.5.1 Struktur

SMS-systemet er programmert på en slik måte at mottakersiden og sendesiden til modemet ikke er direkte avhengig av hverandre. Hver enkelt del (motta, sende) kan aktiveres hver for seg i Innstillinger-vinduet i HMI (kap. 4.43). Filstrukturen reflekterer dette.



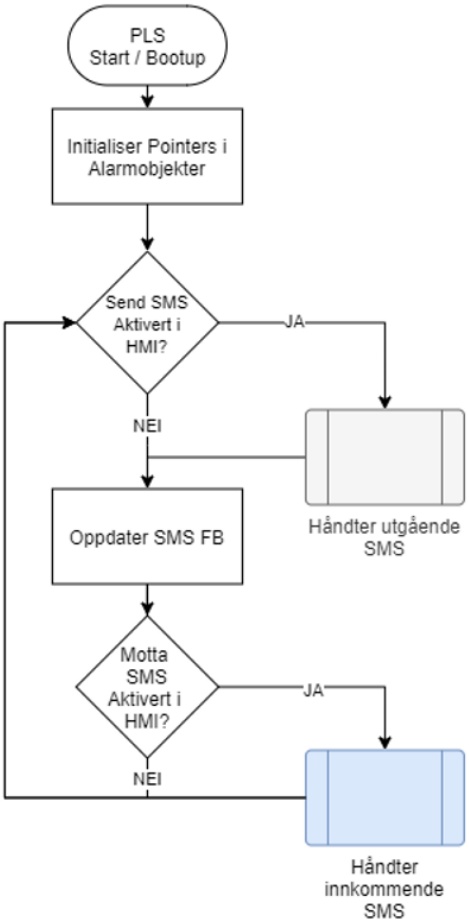

**Figur 4.16:** SMS filstruktur

Det er laget en egen mappe for protokollkommandoer. Selv om applikasjonen er noe kompleks er det relativt enkelt å legge til flere kommandoer. Filer i protokollmappa blir kallet fra fil: "SMS_ATTEMPT_ACTION". Ved utvidelse av SMS-systemet er det derfor ikke behov for å endre programmet andre steder enn denne fila og legge til en ny fil i mappen "Protocol_Commands".

I den siste mappen "Other" er det to filer. Filen "Update_alarmobjects" brukes for å initialisere pekere til alarmtekst i alle alarmobjekter. Som nevnt tidligere er det svært viktig at alle pekere er initialisert før de kalles på for å unngå programkrasj. Den andre filen i mappen er brukt for alarmdeteksjon av SMS-funksjonsblokken. Ved feil på SMS-blokken, eller hvis blokken kjører seg fast, vil programmet automatisk koble ut funksjonsblokken og oppdatere telefonnummer, før den kobler seg til igjen.

### 4.5.2 SMS: Hovedsekvens

Aller først i hovedsekvensen blir pekere til alarmtekster i alarmobjekter initialisert. Normalt sett initialiseres variabler direkte der de blir definert, men i dette tilfellet må tekstlister i alarmkonfiguratoren ha tid til å laste inn først. Det er derfor behov for å gjøre dette et annet sted i programmet. Siden det kun er SMS-systemet som bruker denne delen av alarmobjektet er det naturlig at dette blir gjort her. Resten av SMS-systemet vil ikke klare å kjøre før dette er blitt korrekt initialisert.



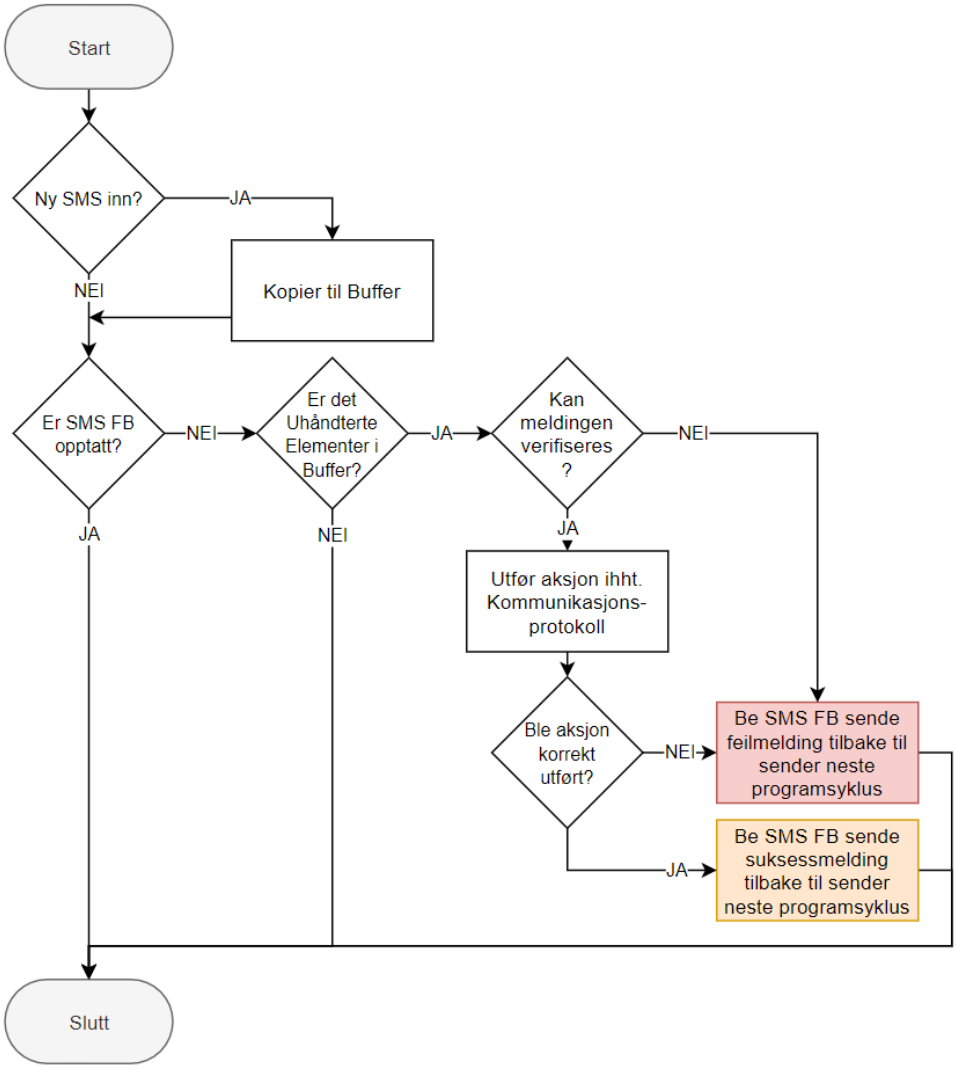

**Figur 4.17:** SMS Hovedsekvens

Delprogrammene "Håndter utgående SMS" og "Håndter innkommende SMS" er to separate deler, og kjøres kun om de er aktivert i innstillingene i HMI-en. Steget fra flytskjemaet 'Oppdater SMS FB' referer til funksjonsblokken for SMS styring fra figur 4.15, kap. 4.5.

Merk at alle flytskjema/figurer illustrert i denne delen av rapporten er noe forenklet. For en enda dypere forståelse av systemet må man lese seg opp i programkoden.

SMS-systemet er programmert på en slik måte at den ikke har en stor og uforutsigbar virkning på syklustiden til programmet. Når SMS-funksjonsblokken er opptatt eller den må vente på andre delprogrammer, vil den gå videre i hovedprogrammet og prøve videre neste syklus.

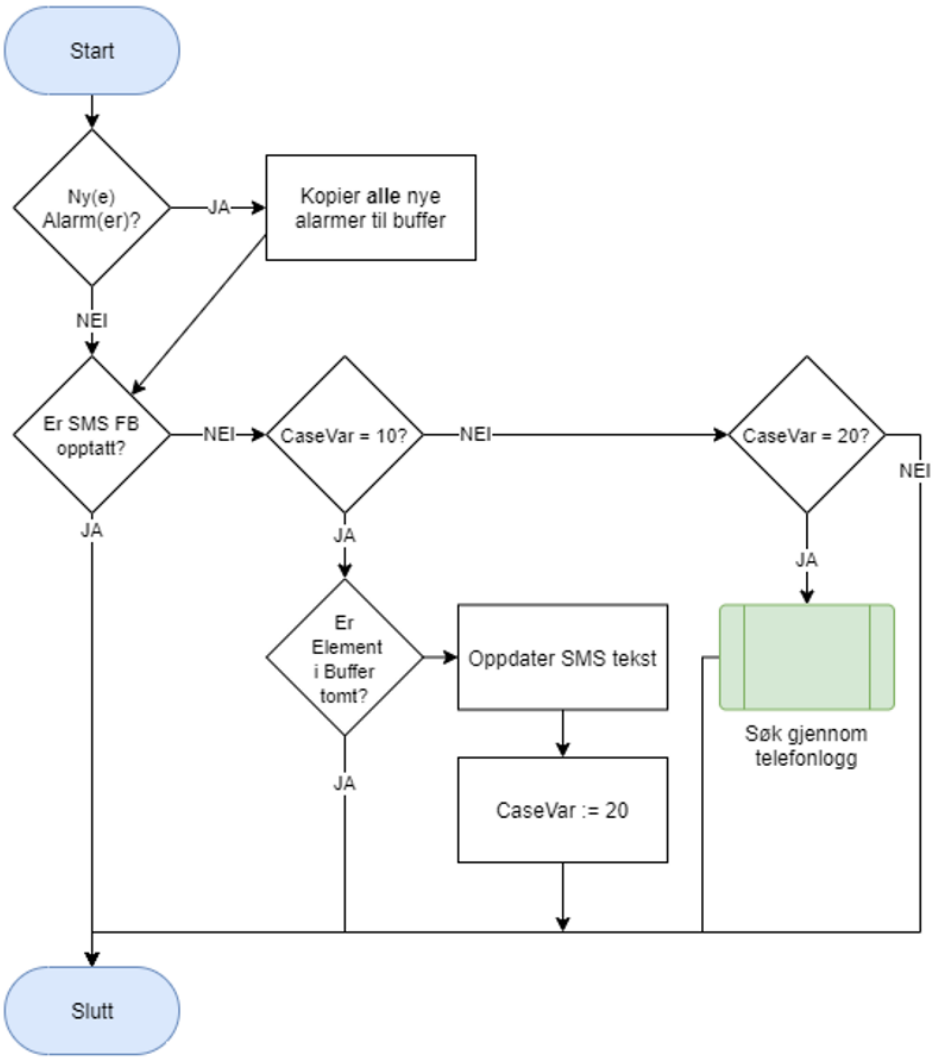
### 4.5.3 Håndtering av innkommende SMS




**Figur 4.18:** Håndtering av innkommende SMS

Denne delen av SMS-systemet håndterer innkommende SMS og forsøker deretter å utføre kommandoer i henhold til oppsatte protokoller (kap. 4.5.5). Hvis en kommando ikke blir gjenkjent vil det bli sendt en feilmelding tilbake til avsender. Innkommende SMS-er blir kopiert over i buffer så raskt som mulig slik at flagget kan kvitteres på funksjonsblokken og ny SMS kan mottas. SMS-funksjonsblokken kan være opptatt, enten av en tidligere kommando eller utsenderdelen av SMS-systemet. Kun om systemet ikke er opptatt, og det ligger uhåndterte elementer i buffer, vil systemet verifisere meldingen og utføre spesifikk aksjon.

### 4.5.4 Håndtering av utgående SMS




**Figur 4.19:** Håndtering av utgående SMS

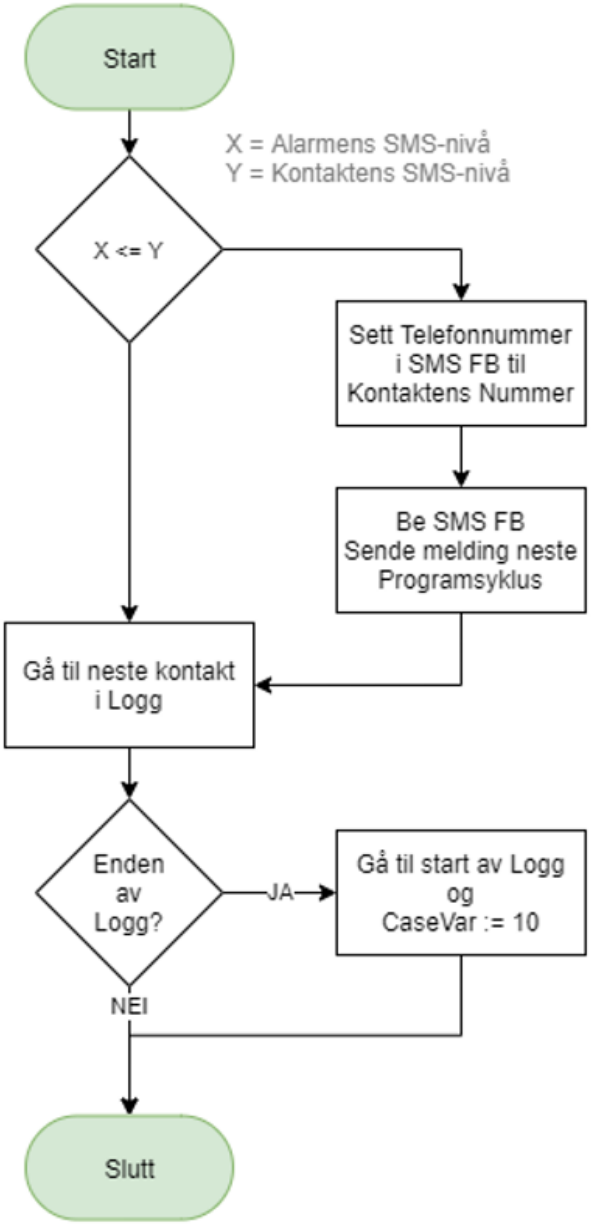
Denne delen av programmet håndterer utgående SMS, og gjelder kun for alarmmeldinger. Hver gang en alarm går høy vil den bli lagret i en buffer. Hvis flere alarmer går høy i samme programsyklus vil alle disse lagres i bufferen for behandling senere når funksjonsblokken blir ledig.

Når SMS-funksjonsblokken ikke er opptatt, vil den befinne seg i ett av to steg. I det første steget vil den lete etter tomme elementer i buffer. Når den har funnet et nytt element vil tekstmeldingen i SMS-funksjonsblokken oppdateres og den går over til det andre steget. Her blir det søkt igjennom en kontaktlogg, for utsending av SMS.

I brukergrensesnittet er det lagret en kontaktlogg som inneholder alle telefonnumrene som har tilgang til SMS-funksjonene i systemet. Det er kun kontakter oppført i denne kontaktloggen som kan motta SMS-varslinger (kap. 4.6.7.8). Av disse skal som nevnt kun kontakter med SMS-nivå større eller lik alarmobjektets SMS-nivå motta varsling. På denne måten kan man skrive seg opp i telefonloggen på HMI-en, og samtidig unngå å motta varslinger fra alle alarmer. SMS-nivået til hvert enkelt alarmobjekt kan bli endret på i HMI-en (kap. 4.6.7.9).

Merk at alarmobjekter med SMS-nivå 0 ikke vil varsles i det heletatt, da nivå 0 tilsvarer at den er deaktivert fra SMS-varsling.

Etter programmet har søkt gjennom hele loggen går den tilbake til steget hvor den igjen vil lete etter nye elementer i buffer.




**Figur 4.20:** Søk av telefonlogg

### 4.5.5 Kommunikasjonsprotokoll

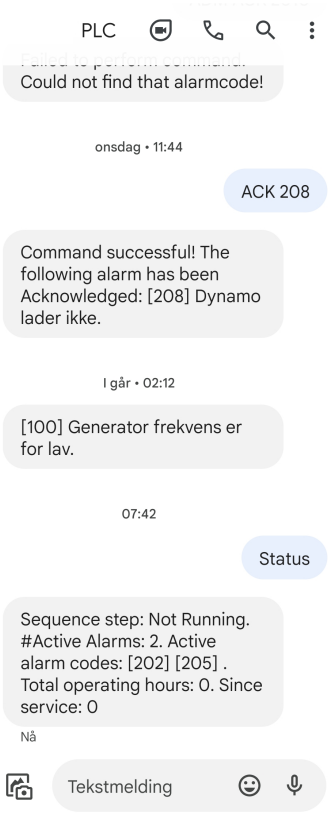
For å kommunisere med PLS-en over SMS ble det utviklet en kommunikasjonsprotokoll. Protokollen bruker mellomrom for å skille mellom ulike kommandoord, hvor det ikke skilles mellom små og store bokstaver. Når PLS-en får en melding må den verifiseres, som gjøres ved å sjekke om senderens telefonnummer står oppført i telefonloggen, eller om meldingen starter med et administratorprefiks. På denne måten har man en bakdør inn til systemet om man ikke allerede ligger i telefonloggen. Man slipper også å være fysisk ved systemet for å legge til en ny kontakt i telefonloggen, eller for å be om statusmelding fra anlegget. Et eksempel på bruk av administratorprefiks kan være som følger:

**{ADMINPREFIKS} STATUS**

Dette prefikset bør holdes hemmelig, da det åpner for at hvem som helst kan sende kommandoer til systemet. For full oversikt over tilgjengelige kommandoer se, vedlegg E: Kommunikasjonsprotokoll.

Det er klargjort i programmet at kommunikasjonsprotokollen tilbyr følgende kommandoer:

- **ACK {alarmkode}**
  Kvittering av individuelle alarmer. Programmet vil akseptere alle alarmkoder brukt i alarmsystemet, men kvittering av alarmer som er av type REP, har ingen effekt. Dette er alarmer som forsvinner av seg selv når feilen rettes opp (se vedlegg G). Eksempel, kvittering av nødstoppalarm: ACK 207
- **ACK ALL**
  Kvitterer alle aktive alarmer.
- **ORTA {minutter}**
  Deaktiverer temperaturalarm på kjølevæske i gitt antall minutter (kap. 4.2.4.3).
- **ADDTELE {tlf} {navn} {nivå}**
  Se neste side (figur 4.22).
- **RESETTELE**
  SMS-nivået på alle kontakter blir satt til 0 slik at de ikke vil motta SMS ved alarm.




**Figur 4.21:** SMS eksempel

- **STATUS**
  PLS-en vil returnere følgende informasjon: Sekvenssteg, antall aktive alarmer, liste med alarmkoder, antall driftstimer og timer siden service. Se figur 4.21.

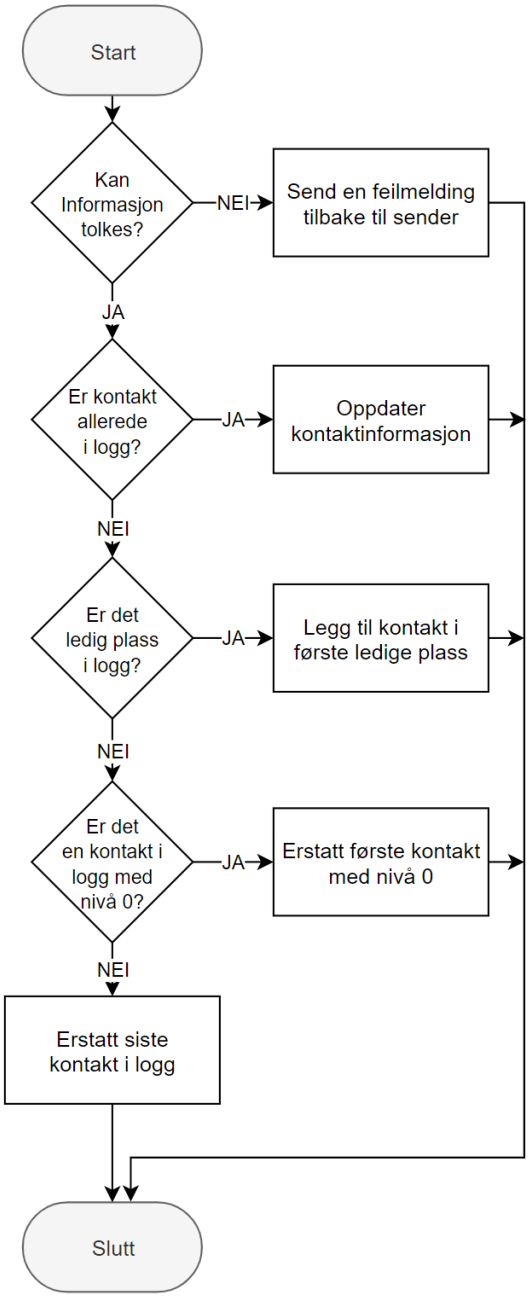
ADDTELE er den mest omfattende kommandoen man kan sende. Den legger til en ny kontakt i kontaktloggen med angitt navn og SMS-nivå.

Først vil den sjekke om informasjonen kan tolkes korrekt. Hvis telefonnummeret ikke har 8 siffer, vil den sende tilbake en feilmelding. Om angitt SMS-nivå ikke kan lagres som en USINT, vil programmet anta at vedkommende ønsker å bli varslet. Programmet setter derfor automatisk et høyt SMS-nivå (254), for å sikre at mottaker får alarmer på SMS. Her er det brukt "TRY/CATCH", hvor programmet prøver å utføre oppgaven og hopper ut hvis det oppstår en feil. Med denne løsningen unngår man at programmet krasjer, uavhengig av hvilken informasjon som kommer inn over SMS.

Hvis en kontakt allerede er lagt til i loggen vil kontaktens navn og SMS-nivå oppdateres, istedenfor å legge til ny kontakt. På denne måten unngår man at samme kontakt blir varslet flere ganger når ny alarm blir aktivert.

Videre, hvis det er ledig plass i loggen blir ny kontakt lagt til. Hvis det ikke er ledig plass i loggen vil den søke bakover gjennom loggen etter en kontakt med nivå 0, og overskrive denne. Hvis dette også feiler, vil siste kontakt i loggen overskrives.

Uansett hva som skjer skal sender motta en melding tilbake, enten i form av en konfirmasjonsmelding eller feilmelding. Se Vedlegg E for full oversikt.




**Figur 4.22:** Legge til ny kontakt i telefonlogg over SMS

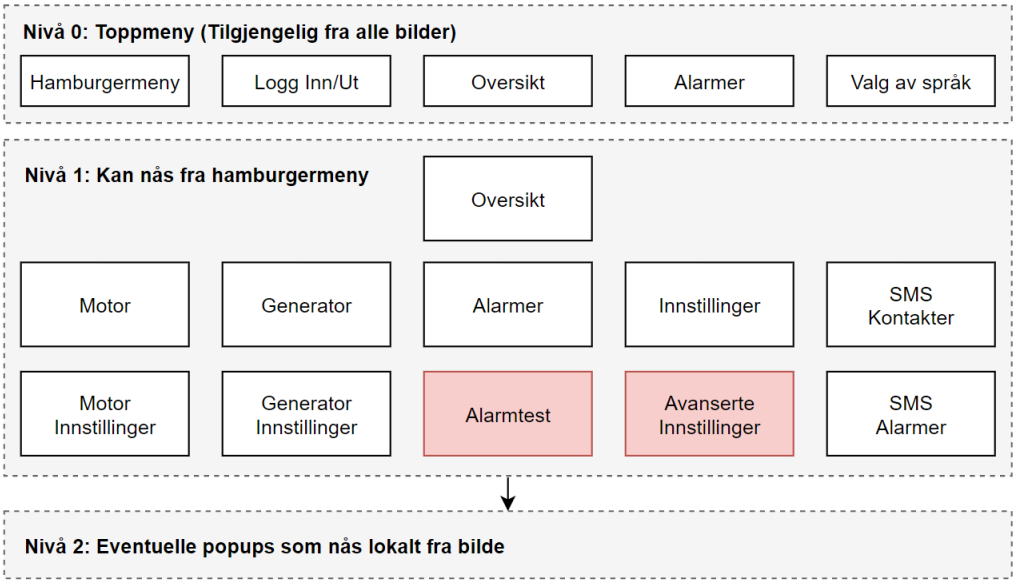
## 4.6 Brukergrensesnitt - HMI

Ved design av brukergrensesnitt er det forsøkt å følge standarden for "High Performance HMI" [13], spesielt ved fargevalg og minimalistisk design. Skjermen som brukergrensesnittet er designet for er på bare 7,0 tommer, altså ikke mye større enn dagens telefonskjermer. Panelet er en touchskjerm og det antas at bruker stort sett ikke har tastatur og mus tilgjengelig. For at brukergrensesnittet skal være mest mulig brukervennlig har gruppen derfor lagt stort fokus på å gjøre tekst og knapper store og oversiktlige. Dette innebærer blant annet at "hitbox" på enkelte brytere er noe større enn utseende gir uttrykk for, og talltastatur ("numpad") og tastatur ("keypad") 4.6.6 er forstørret til å dekke mest mulig av skjermen.

### 4.6.1 Hierarki

Brukergrensesnittets hierarki er designet til å være så oversiktlig som mulig, med færrest mulig nivåer. Når programmet først lastes opp på PLS-en blir man sendt til et oversiktsvindu. Når man ikke er logget inn har man leserettigheter til de fleste vinduer, bortsett fra vinduene markert med rødt i figur 4.23. Ved innlogging som administrator, som er det høyeste nivået, er det full tilgang på alt med lese- og skriverettigheter.

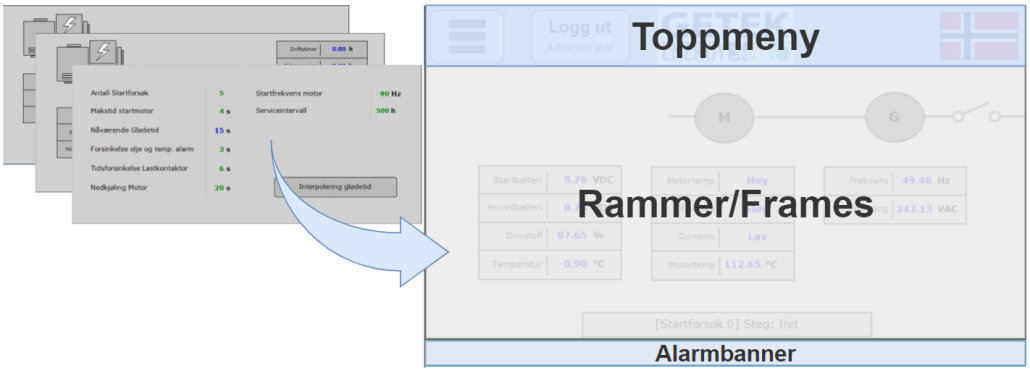
Brukergrensesnittet er delt opp i 3 nivåer. Toppmenyen (Nivå 0) er tilgjengelig i alle vinduer. Alle hovedvinduer (Nivå 1) er tilgjengelige gjennom hamburgermenyen i toppmenyen, og fra disse vinduene kan man åpne ulike pop-up-vinduer (Nivå 2), som kun er tilgjengelig på det lokale vinduet. I tillegg er det lagt til snarveier, slik at man kan trykke på alarmlampen for å bli sendt til alarmvinduet, eller Getek-logoen i toppmeny for å bli sendt til oversiktsvinduet. Merk at disse også er tilgjengelige i hamburgermenyen.




**Figur 4.23:** HMI-hierarki

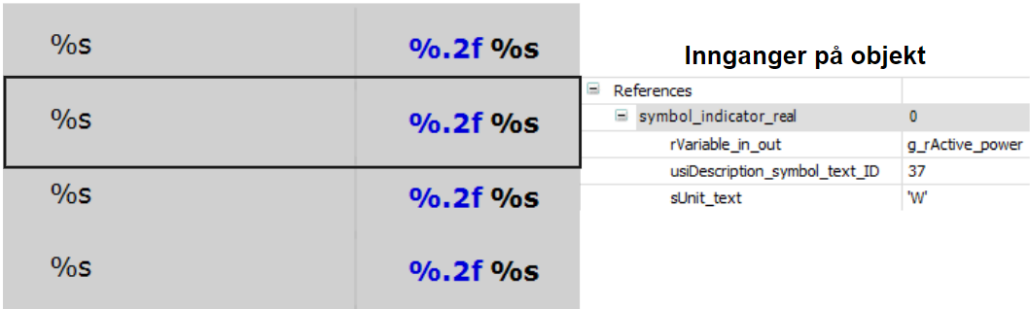
### 4.6.2 Modulært design

Brukergrensesnittet er programmert på en modulær måte. Hovedbildet er delt opp i tre deler, som illustrert på bilde 4.24. Alle bilder inneholder toppmeny og alarmbanner. Når bruker navigerer gjennom brukergrensesnittet er det kun midten av bildet som endres. Til dette er det brukt rammer, eller "Frames" på engelsk, som endrer visualiseringsbilde ved bestemte kommandoer.




**Figur 4.24:** Illustrasjon av konseptet rundt bruk av rammer ("Frames").

De fleste elementene i de ulike rammene er programmert som egne maler ("templates" på engelsk,) som vist på bilde 4.25. Dette gjør programmeringen effektiv, siden malene kan brukes flere steder. I tillegg blir det lettere å gjøre programendringer, da det bare er å endre på malen i stedet for hvert elementet.




**Figur 4.25:** Eksempel på bruk av maler ("templates").

### 4.6.3 Fargetema

Ved valg av fargetema har gruppen som nevnt tidligere tatt stor inspirasjon fra "High-Performance HMI". Tabellen vist i figur 4.26 viser hvilke farger som er brukt og hvilken funksjonalitet hver enkelt farge har. Spesielt viktig er det å merke seg at blå farge er brukt på variabler med leserettigheter, mens variabler med lese- og skriverettigheter har grønn farge.

Programmet initialiseres med et lyst fargetema, men det er også lagt til muligheten å endre dette til mørk modus i innstillingsvinduet i HMI (kap. 4.43). Alle fargevariabler er lagret i en global variabelliste kalt "GVL_color". På denne måten er det enkelt å endre fargetema eller legge til nytt tema ved behov. En svakhet med dette systemet er at alle klienter deler samme fargetema. Uansett har det liten betydning at fargetema er likt, og svært usannsynlig at mange klienter vil være aktive på HMI-en samtidig.

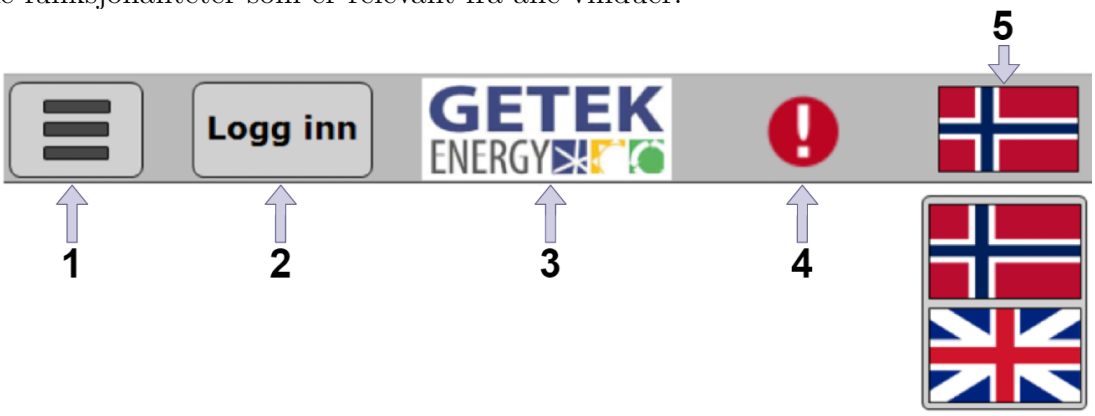
I tillegg er det brukt visustyle.xml-filer til å gi riktig fargetema, teksttyper og stil. Her er det to filer man kan velge mellom: "Light.visustyle.xml" og "Dark.visustyle.xml", ettersom hvilken man ønsker å bruke. Det anbefales å installere en av disse før man jobber på HMI-programmet i e!COCKPIT. Hvis ikke ser man ikke riktig farger på elementer før man går i simulering, eller laster opp koden til kontrolleren.

| Fargetema | | | | | | |
|---|---|---|---|---|---|---|
| R | G | B | HEX / DINT | Visning | GVL_color | Brukt til |
| Vanlig fargetema (Lys) | | | | | | |
| 0 | 0 | 0 | 16#FF000000 | | text_normal | Normal tekst |
| 0 | 0 | 215 | 16#FF0000D7 | | text_output | Tekst på verdier som ikke kan endres |
| 0 | 128 | 0 | 16#FF008000 | | text_input | Tekst på verdier som kan endres |
| 255 | 43 | 43 | 16#FFFF2B2B | | alarm_1 | Alarmfarge |
| 255 | 255 | 0 | 16#FFFFFF00 | | alarm_2 | Advarselfarge |
| 186 | 186 | 186 | 16#FFBABABA | | bg_1 | Bakgrunnsfarge, brukt på knapper og komponenter |
| 208 | 208 | 208 | 16#FFD0D0D0 | | bg_2 | Bakgrunnsfarge, lengst bak |
| 240 | 240 | 240 | 16#FFF0F0F0 | | component_on | Komponent er PÅ (eks: en motor) |
| 118 | 113 | 113 | 16#FF767171 | | component_off | Komponent er AV (eks: en motor) |
| 70 | 70 | 70 | 16#FF464646 | | symbol_1 | Brukt på enkelte symboler (lite brukt) |
| Mørk fargetema | | | | | | |
| 255 | 255 | 255 | 16#FFFFFFFF | | text_normal | Normal tekst |
| 0 | 172 | 255 | 16#FF00ACFF | | text_output | Tekst på verdier som ikke kan endres |
| 115 | 191 | 69 | 16#FF73BF45 | | text_input | Tekst på verdier som kan endres |
| 255 | 43 | 43 | 16#FFFF2B2B | | alarm_1 | Alarmfarge |
| 255 | 255 | 0 | 16#FFFFFF00 | | alarm_2 | Advarselfarge |
| 60 | 60 | 60 | 16#FF3C3C3C | | bg_1 | Bakgrunnsfarge, brukt på knapper og komponenter |
| 80 | 80 | 80 | 16#FF505050 | | bg_2 | Bakgrunnsfarge, lengst bak |
| 150 | 150 | 150 | 16#FF969696 | | component_on | Komponent er PÅ (eks: en motor) |
| 118 | 113 | 113 | 16#FF767171 | | component_off | Komponent er AV (eks: en motor) |
| 220 | 220 | 220 | 16#FFDCDCDC | | symbol_1 | Brukt på enkelte symboler (lite brukt) |
| Keypad / Numpads Bibiliotek | | | | | | |
| 35 | 40 | 41 | 16#FF232829 | | | Bakgrunn, lengst bak |
| 87 | 96 | 103 | 16#FF576067 | | | Knapper, øverste bånd etc. |
| 255 | 255 | 255 | 16#FFFFFFFF | | | Tekst farge (Hvit) |

**Figur 4.26:** Fargetema tabell, fra makrobasert Excel-ark

### 4.6.4 Toppmeny

Toppmenyen er tilgjengelig i alle vinduer i HMI-en, med unntak av popup-vinduer som tastatur. Disse unntakene er vinduer der det anses som viktigere å bruke meste mulig av bilde for å gjøre interaksjonen mest mulig brukervennlig. Toppmenyen inneholder en rekke funksjonaliteter som er relevant fra alle vinduer.




**Figur 4.27:** HMI, toppmeny

Innholdet i toppmenyen er som følger:

1. **Hamburgermeny**
   Åpner menysystem med alle tilgjengelige hovedbilder (kap. 4.6.5).

2. **Logg inn**
   Åpner logg-inn vindu (kap. 4.6.6), endres til "Logg ut" og viser aktiv bruker etter vellykket innlogging.

3. **Logo**
   I hovedsak bare reklame, men man kan klikke på logoen for å bli sendt til hovedbilde (kap. 4.6.7.1).

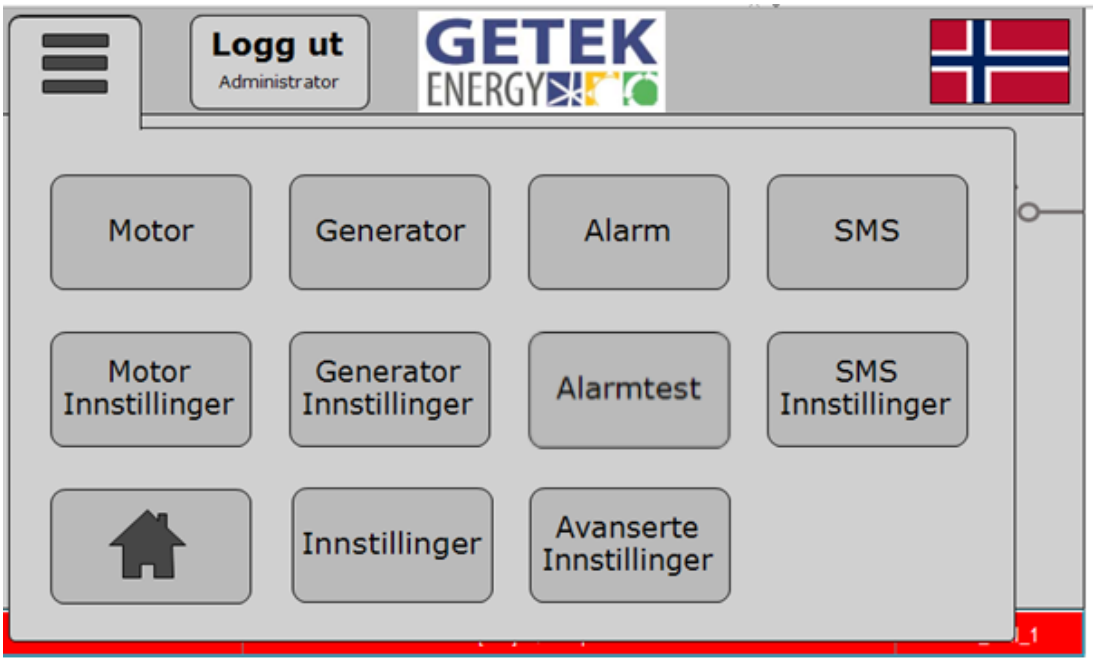
4. **Alarmlampe**
   Se delkapittel 4.4.7 for mer info om alarmlampe. Kan også trykkes på for å bli sendt til alarmvindu.

5. **Flagg**
   Åpner et lite popup-vindu for valg av språk. Foreløpig kun Norsk og Engelsk. Flere språk kan relativt enkelt legges til.

### 4.6.5 Hamburgermeny

"Hamburgermenyen" er et vindu hvor man har tilgang til alle hovedvinduene som brukergrensesnittet tilbyr. Det er dermed enkelt å hoppe fra et vindu til et annet. Hamburgermenyen er tilgjengelig fra alle vinduer, med unntak av enkelte lokale popup-vinduer. Knappene i menyen er store og er designet for bruk på en liten berøringsskjerm. Det er lagt av plass nede til høyre på menyen for eventuelle videre utvidelser av systemet.

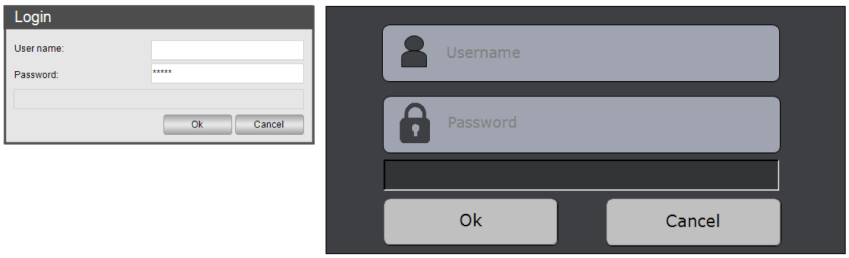
Hvert enkelt vindu er beskrevet senere i rapporten (kap. 4.6.7).



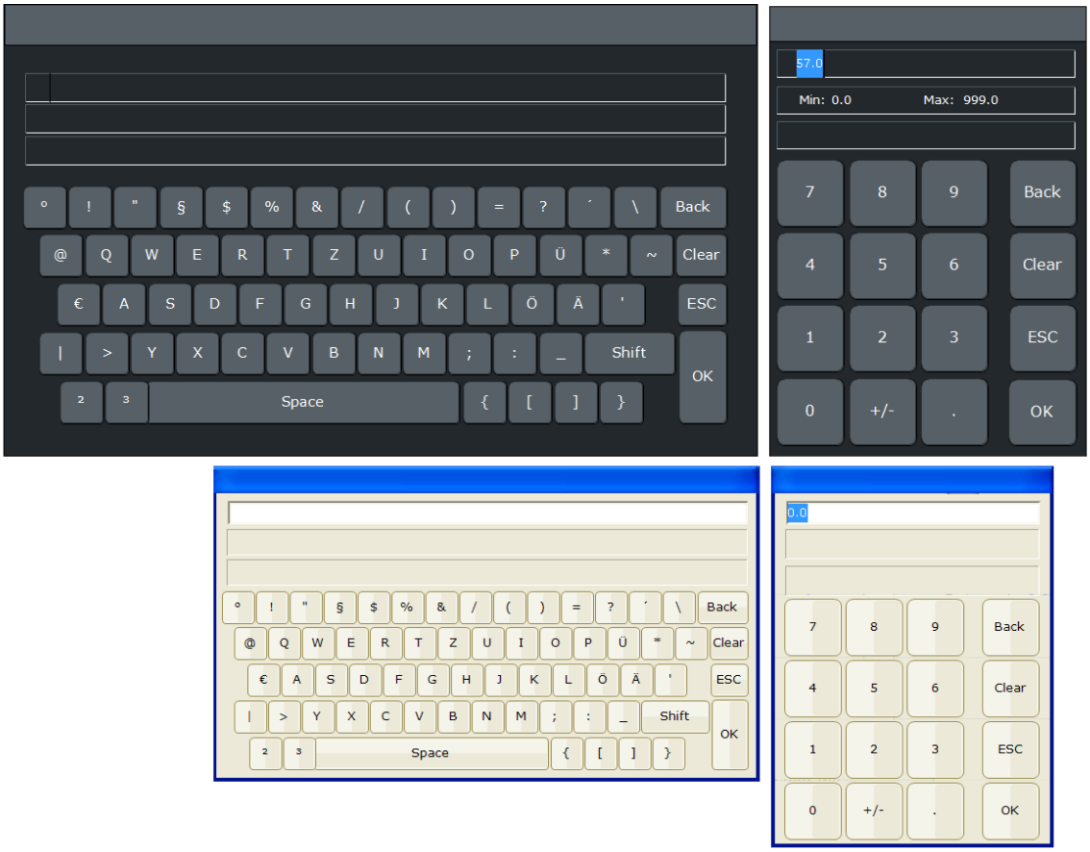

**Figur 4.28:** HMI menysystem, Hamburgermeny

### 4.6.6 Pop-up: Logg Inn, Tastatur og Talltastatur

En av tingene gruppen har gjort for å øke brukervennligheten på touchpanelet er å øke størrelsen på popup-vindu. Både det tidligere logg-inn-vinduet og tastaturene var i en liten skala i forhold til panelet som skulle brukes på APS-en, og det ble dermed bestemt å endre disse for vårt bruk. Her er det blitt tatt kopi av eksisterende biblioteker og endret størrelse, samt gått for et mer minimalistisk og flatt utseende.




**Figur 4.29:** Logg-inn-vindu original (V) vs. modifisert (H)



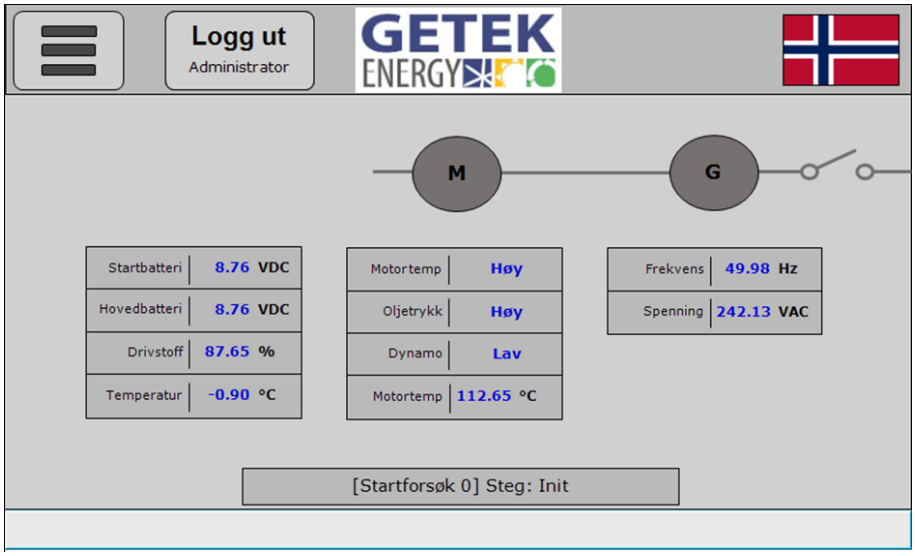

**Figur 4.30:** HMI tastatur og talltastatur, original (bunn) vs. modifisert (topp)

### 4.6.7 Vinduer

De neste delkapitlene viser hvilke vinduer som er implementert i brukergrensesnittet. For full oversikt over lese- og skriverettigheter på hvert vindu, se vedlegg H.

#### 4.6.7.1 Oversikt

Oversiktsvinduet kan nås enten ved å trykke på hjemknappen (bilde av hus) fra hamburgermenyen eller ved å trykke på logoen i toppmeny. Det er ingen skriverettigheter i dette vinduet. Motor, generator og brytersymbol (lastkontaktor) vil endre farge og utseende basert på om det er på eller av (se figur 4.26 for oversikt over fargetema). Motor- og generatorsymbolet er også mulig å trykke på, slik at man sendes direkte til de respektive vinduene.



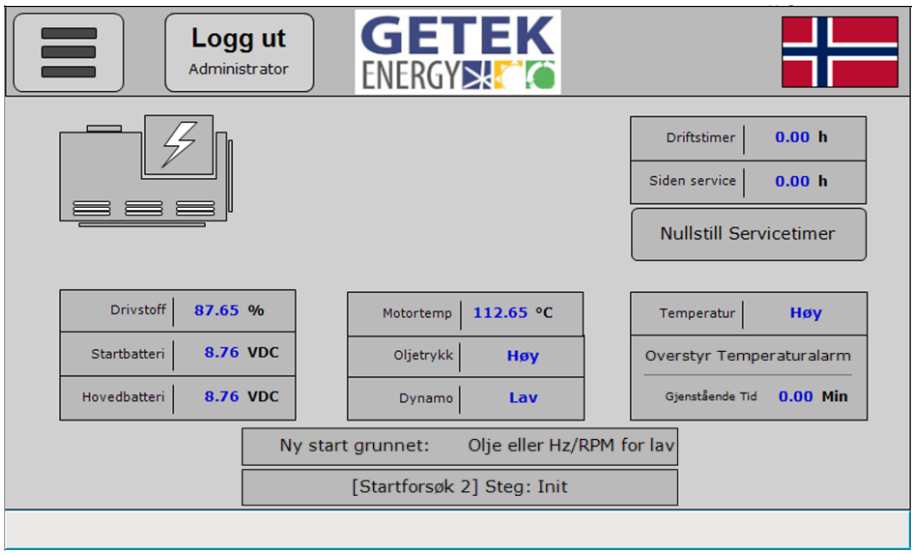

**Figur 4.31:** HMI-vindu: Oversikt

Vinduet fra figur 4.31 inneholder 3 kolonner med visualiserte verdier. Venstre kolonne viser noen praktiske verdier i forbindelse med aggregatet. Den midterste kolonnen viser verdier knyttet til motordelen av anlegget. Høyre kolonne viser verdier knyttet til generatordelen av anlegget. Merk at for å ha visualiseringen av disse verdiene må man ha aktivert måling av de respektive signalene, som gjøres i vinduet for "Avanserte Innstillinger" (kap. 4.6.7.11). Verdiene vist på denne og de neste figurene stemmer ikke helt, da man ikke var påkoblet fysisk rigg når bildene ble tatt.

På bunnen av vinduet er det en indikator som viser nåværende startforsøk og hvilket sekvenssteg sekvensen befinner seg i. Hvis det kreves flere enn ett startforsøk vil det komme fram en boks med årsak for nytt forsøk (eksempler på dette i figur 4.32 og 4.34). Mye av denne informasjon er også på motor- og generatorvindu med enda flere visualiserte variabler. Det er mye likhet i stil og oppsett mellom oversikt-, motor- og generatorvindu.

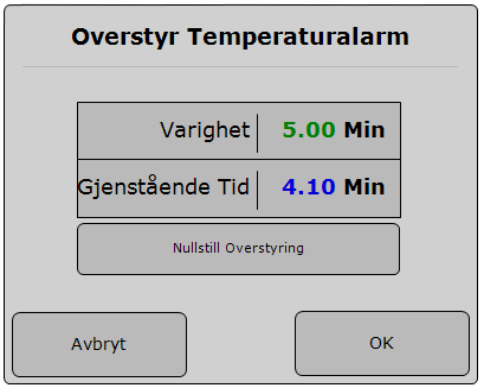
#### 4.6.7.2 Motor

Motorvindu viser de relevante verdiene for dieselmotoren. Oppe til høyre i vinduet er det visualisert totale driftstimer av motoren, timer siden siste service og en knapp for nullstilling av servicetimer. For å kunne nullstille servicetimene må man være innlogget som administrator. I tillegg er det gjort slik at hvis man trykker på knappen for nullstilling vil et popup-vindu dukke opp og spørre om man virkelig ønsker å utføre handlingen. Brukeren har her mulighet til å bekrefte eller avbryte nullstillingen.




**Figur 4.32:** HMI-vindu: Motor

Nederst til høyre i motorvindu har bruker mulighet til å overstyre temperaturalarmen for kjølevæske, som nevnt i kapittel 4.2.4.3. For å gjøre dette krever det at man er logget inn som administrator.

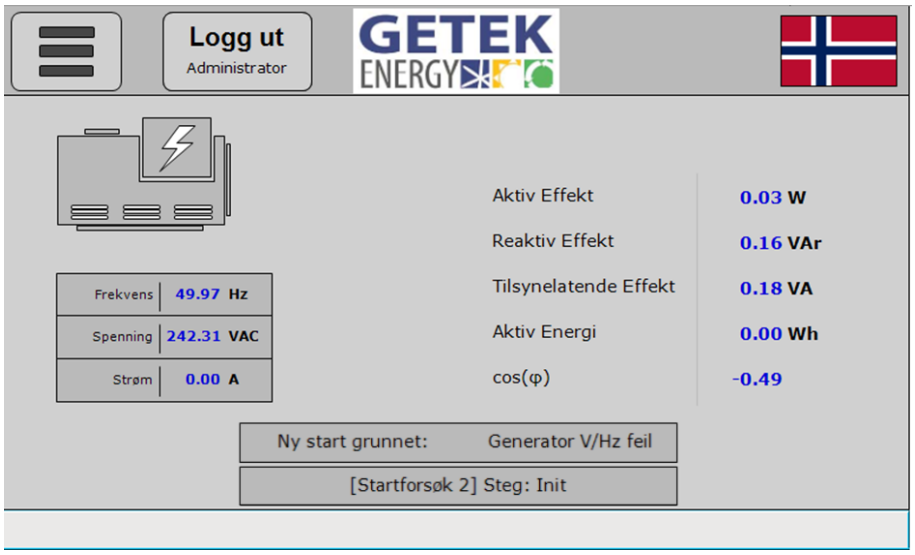



**Figur 4.33:** HMI-popup: Overstyr temperaturalarm

Hvis bruker trykker på boksen med "Overstyr Temperaturalarm" og "Gjenstående Tid", vil det komme fram et popup-vindu som inneholder varighet, gjenstående tid og nullstilling (hvis aktiv). Bruker kan her trykke "OK" eller "Nullstill Overstying" for å endre overstyringstiden, eller "Avbryt" for å lukke vindu uten endringer.

#### 4.6.7.3 Generator

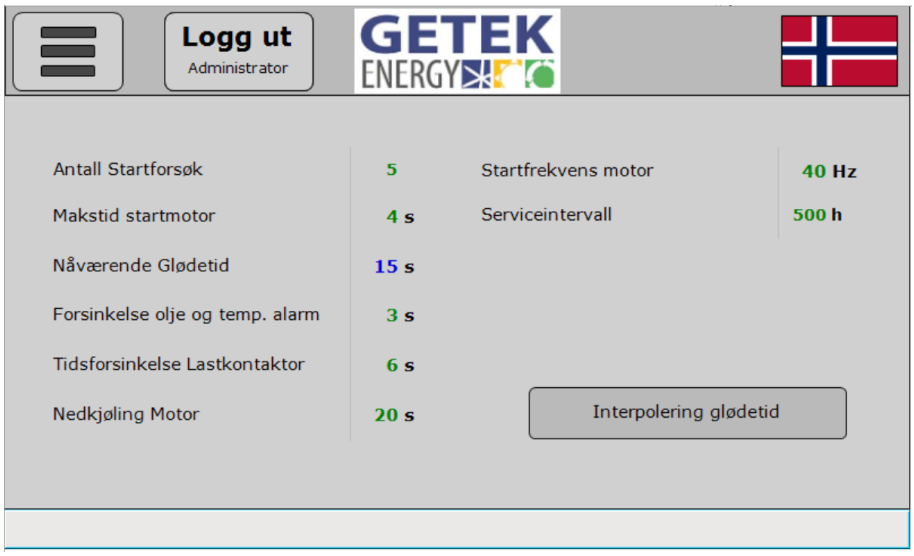
Generatorvinduet inneholder målingene av spenning og frekvens som produseres av generatoren. Ved å i tillegg ha strømmåling på PLS-en så blir også den aktive energien, effektfaktoren $cos(\phi)$, og den aktive-, reaktive- og tilsynelatende effekten visualisert. Det er kun frekvens og spenning som er absolutt nødvendig for styringen av dieselaggregatet, da dette er knyttet opp mot sekvensstyringen til aggregatet. All annen informasjon er informasjon som kan være verdt å vite, men ikke nødvendig. Om strømmåling ikke er tilgjengelig deaktiveres dette i vinduet for avanserte innstillinger (se kapittel 4.6.7.11).




**Figur 4.34:** HMI-vindu: Generator

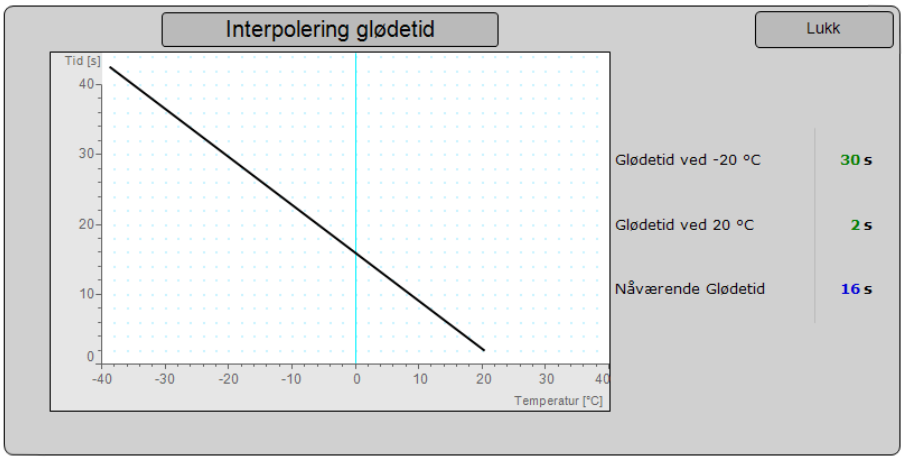
#### 4.6.7.4 Motor Innstillinger

For å lage programmet mest mulig fleksibelt for forskjellige typer motorer og variabelt vær, er mange av parameterne knyttet opp mot sekvensstyringen lagt til i et vindu kalt "Motor Innstillinger". Her kan man endre parameterne for å tilpasse sekvensen etter omgivelsene og motor. Dette inkluderer antall startforsøk før sekvens går i alarm, hvor lang ventetid sekvensen maksimalt har mellom diverse steg og antall timer for serviceintervall. Variabler som er justerbare på dette vindu er også merket i sekvensskjema (Se vedlegg D: Sekvensskjema, for mer informasjon).




**Figur 4.35:** HMI-vindu: Motor Innstillinger

Innstilling av glødetid ved hjelp av lineær interpolasjon (kap. 3.6) har et eget popup-vindu. Knappen for å åpne dette vindu er kun synlig når måling av omgivelsestemperaturen er slått på i avanserte innstillinger (kap. 4.6.7.11). Popup-vinduet har en grafisk visualisering av glødetider ved forskjellige temperaturer. Her angis glødetiden for −20 °C og +20 °C og nåværende glødetid ut ifra temperatur vises.

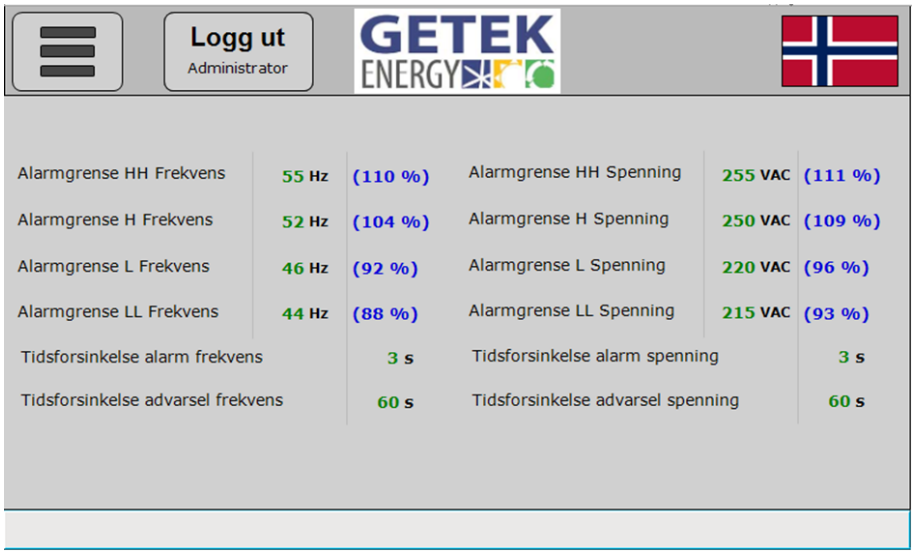



**Figur 4.36:** HMI-popup: Interpolering glødetid

#### 4.6.7.5 Generator Innstillinger

Målingene av frekvens og spenning fra generatoren sammenliknes med bestemte grenseverdier for å sørge for at de holder seg innenfor et akseptabelt område. Disse grenseverdiene kan endres på i "Generator Innstillinger"-vinduet, hvor man skriver inn grenseverdien i henholdsvis Hertz eller Volt. Ved siden av grenseverdien blir det regnet ut og visualisert grenseverdiens prosentandel av nominellverdien på henholdsvis 50 Hertz og 230 Volt.

I det målingen går utenfor grenseverdiene vil det etter en bestemt tid slå ut en advarsel eller alarm, avhengig av hvor kritisk denne overtredelsen er. Denne tidsforsinkelsen er justerbar nederst i vinduet. Forsinkelsen for alarm gjelder for grensene HH eller LL, og advarsel gjelder for grensene H og L.




**Figur 4.37:** HMI-vindu: Generator Innstillinger

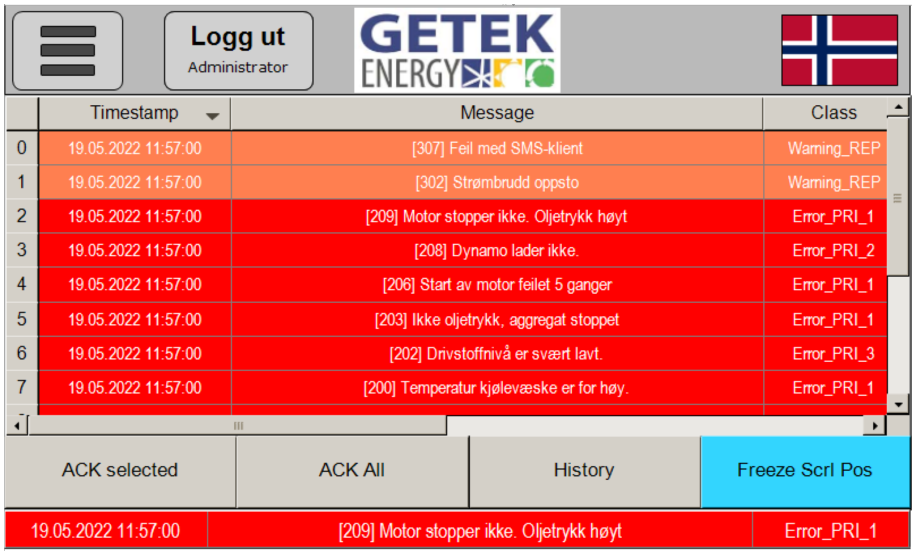
#### 4.6.7.6 Alarm

Alarmvinduet består av en tabell som visualiserer aktive alarmer, og denne er knyttet opp mot den innebygde alarmkonfiguratoren (se kapittel 3.5.4). Tabellen består av 3 kolonner:

- **Timestamp**
  Inneholder tidspunktet alarmen gikk høy.
- **Message**
  Inneholder selve alarmmeldingen, som gir en beskrivende forklaring på problemet.
- **Class**
  Inneholder hvilken alarmklasse den aktive alarmen er av.

Under alarmtabellen er det lagt til 4 knapper, som har følgende funksjoner:

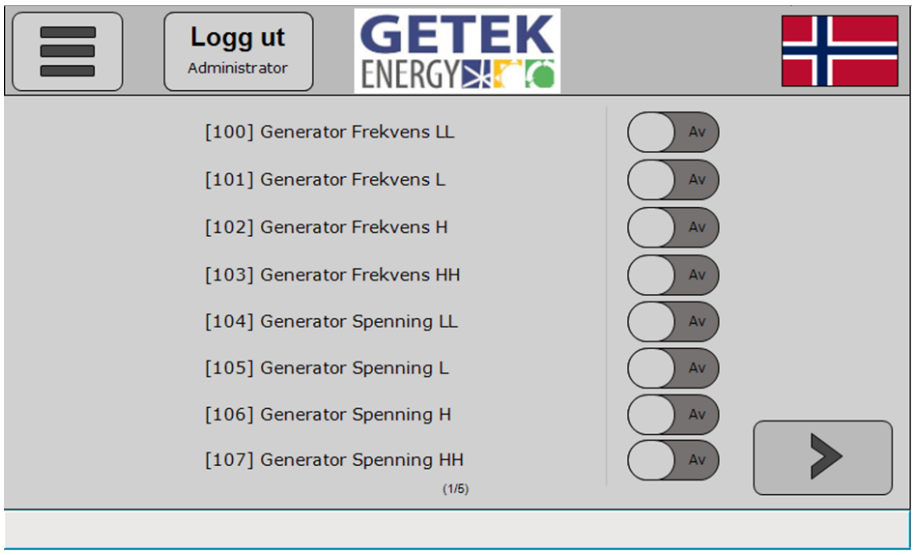
- **ACK selected**
  Brukes for å kvittere enkeltalarmer ved at man først trykker på en alarm i tabellen, og deretter trykker på kvitteringsknappen.
- **ACK All**
  Brukes for å kvittere alle aktive alarmer. Her trenger man ikke å klikke på noen alarmer først.
- **History**
  Åpner alarmhistorikken som viser tidligere aktive alarmer.
- **Freeze Scrl Pos**
  "Fryser" posisjonen i tabellen. Innkommende alarmer vil ikke bla siden nedover.




**Figur 4.38:** HMI.vindu: Alarm

#### 4.6.7.7 Alarmtest

For å teste alarmer er det laget et vindu bestående av 5 sider, der man kan aktivere ønskede alarmer. Sidene består av alarmkode, beskrivende alarmnavn, samt en av/på-knapp som er koblet til alarmdeteksjonsblokkene, som er forklart i kapittel 4.4.6. Nederst på vinduene er det lagt til pil(er) som man kan bruke til å navigere mellom de forskjellige sidene i alarmtest. På den siste siden er det i tillegg en knapp som aktiverer alle alarmene samtidig.




**Figur 4.39:** HMI-vindu: Alarmtest

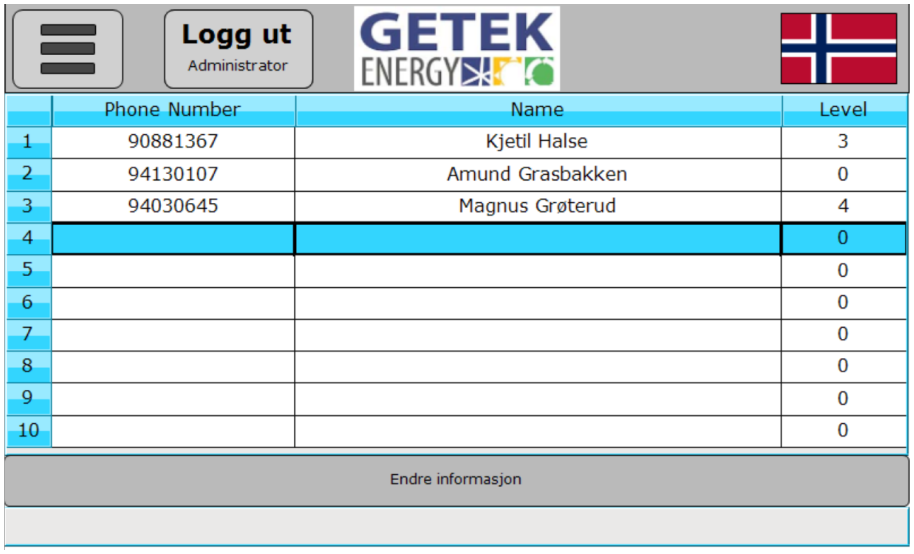
#### 4.6.7.8 SMS Kontakter

Dette vinduet består av en kontaktlogg i form av en tabell. Som nevnt i kapittel 4.5.4 legges det her til kontakter som skal motta alarmvarslinger over SMS og/eller trenger tilgang til å sende kommandoer inn til systemet. Tabellen består av 3 kolonner:

- **Phone number**
  Inneholder telefonnummeret til kontakten. Nummeret brukes ikke bare til å sende alarmer, men også for å verifisere innkommende meldinger, da man ikke skal ha tilgang om man ikke er oppført i denne loggen.

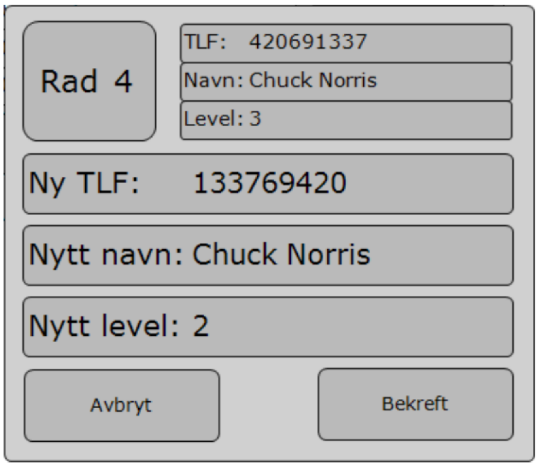
- **Name**
  Inneholder navnet på kontakten. Brukes kun visuelt på brukergrensesnitt og har ingen annen hensikt.

- **Level**
  Inneholder nivået på kontakten. Som nevnt tidligere brukes dette til å begrene type alarmmeldinger kontakten skal motta.




**Figur 4.40:** HMI-vindu: SMS

Ved å trykke på en rad i tabellen popper det opp en knapp nederst på bilde, merket "Endre informasjon". Trykker man på denne åpnes det et vindu der man kan endre innholdet i valgt rad. Dette vinduet viser tidligere informasjon, gir valget om å skrive inn ny informasjon og for hvilken rad i kontaktloggen det gjelder. For at informasjonen som er endret på skal lagres, må man trykke på en bekreftelsesknapp. Det er alltid mulighet til å trykke avbryt, så gammel informasjon forblir lagret.



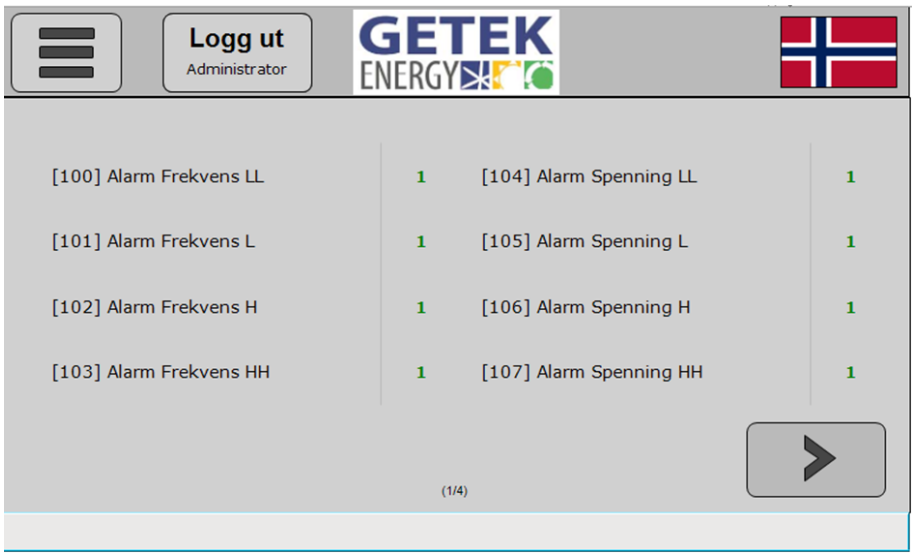

**Figur 4.41:** HMI-popup: Endring i SMS-kontaktlogg

#### 4.6.7.9 SMS Innstillinger

I dette vinduet stilles SMS-nivået til alarmobjektene. Vinduet består av 4 sider og hvert eneste alarmobjekt er oppført på disse sidene. Kontakter oppført i kontaktlogg på brukergrensesnitt vil som nevnt tidligere motta alarmmeldinger bare hvis

$$\textbf{Alarmobjektets SMS-nivå} \leq \textbf{Kontaktens SMS-nivå}$$

Alle SMS-nivåer knyttet til alarmene kan endres på i dette vinduet, om man for eksempel vil luke ut relativt uviktige alarmvarsler.




**Figur 4.42:** HMI-vindu: SMS innstillinger

#### 4.6.7.10 Innstillinger

Det "vanlige" innstillinger-vinduet inneholder innstillinger som ikke er knyttet opp mot generator- og motordelen av systemet, og består av 3 knapper. Den første knappen skifter til ett mørk fargetema for brukergrensesnittet (fig. 4.44). Den andre knappen er for å skru av og på mottakende kommandoer over SMS. Den siste knappen er for å skru av og på alarmvarslinger over SMS, beskrevet i kapittel 4.5.4.

**Figur 4.43:** HMI-vindu: Innstillinger

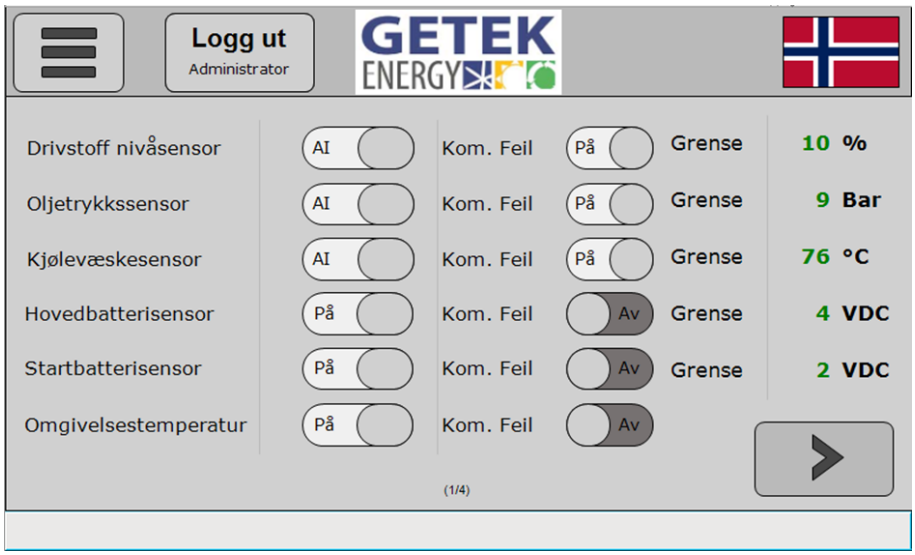
**Figur 4.44:** Eksempel på mørk modus

#### 4.6.7.11 Avanserte Innstillinger

Avanserte innstillinger er et vindu med 4 sider, som krever at bruker er logget inn som administrator. Det er ønskelig at programmet gruppen lager fungerer på flest mulig typer anlegg, uten å måtte omprogrammeres. På oppdragsgivers ulike anlegg kan det variere hvilke signaler som er analoge og digitale. I tillegg er det flere signaler som ikke er nødvendige for å kunne kjøre sekvensstyringen. Gruppen har derfor til størst mulig grad gjort ting valgfritt ut ifra hva som er mest logisk. Avanserte innstillinger er ment for teknisk personell ved installasjon av nytt anlegg eller utstyr, og skal i hovedsak ikke endres på etter vellykket installasjon.

**Avanserte Innstillinger 1**

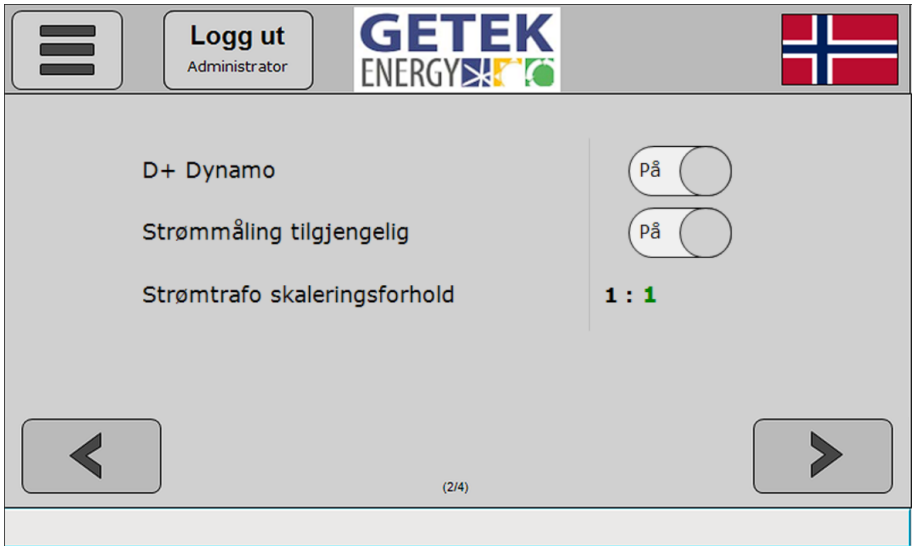
På den første siden velges det hvilke signaler som bruker analoge eller digitale innganger. Man kan fra denne siden også velge å slå av/på måling av hovedbatteri, startbatteri og omgivelsestemperatur, om disse ikke skulle være tilgjengelige på gjeldende anlegg. For alle analoge signaler kan man i tillegg velge å skru på deteksjon av kommunikasjonsfeil hvis ønskelig. Til høyre i vinduet settes alarmgrenser for hver av målingene. Disse grensene gjelder for lav-lav (LL) eller høy-høy (HH) alarmer. Lav (L) og høy (H) alarmer kalkuleres ut ifra verdiene på disse og kan ikke justeres direkte.




**Figur 4.45:** HMI-vindu: Avanserte innstillinger side 1

**Avanserte Innstillinger 2**

På den andre siden har man igjen tre valg. Man huker av om D+ på dynamo er tilgjengelig (denne er brukt for visualisering og alarmdeteksjon) og om strømmåling er tilgjengelig. Hvis strømmåling er tilgjengelig, kan man også justere skaleringen på gjeldende strøm-transformator. Ingen av disse valgene har en alarmaksjon som går ut over sekvensen og er ikke en nødvendighet for å kjøre anlegget, kun for ekstra overvåking.



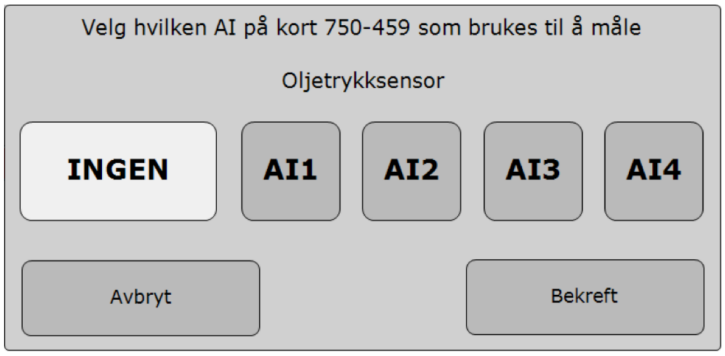

**Figur 4.46:** HMI-vindu: Avanserte innstillinger side 2

### Avanserte Innstillinger 3

På den tredje siden velges det hvilke innganger de analoge sensorene er koblet til på analogt-inngangskortet 750-459. Analogkortet har bare mulighet for tilkobling av fire sensorer. Etter samtaler med oppdragsgiver kan man anta sannsynligheten for et anlegg som har alle fem analoge sensorer er svært liten, og dette blir dermed sett på som en god løsning. Dette vinduet gjør at man kan bruke samme program uavhengig av hvilken kombinasjon av sensorer man har koblet til anlegget, samtidig som man slipper behovet for to analogkort av type 750-459 tilkoblet PLS-en. Valget av innganger vist på bilde 4.47 er initiale verdier, og er de samme som beskrevet i Vedlegg C: IO-liste. Mer spesifikt om hvordan programkoden er utført står i kapittel 4.2.4.4.

**Figur 4.47:** HMI-vindu: Avanserte innstillinger side 3

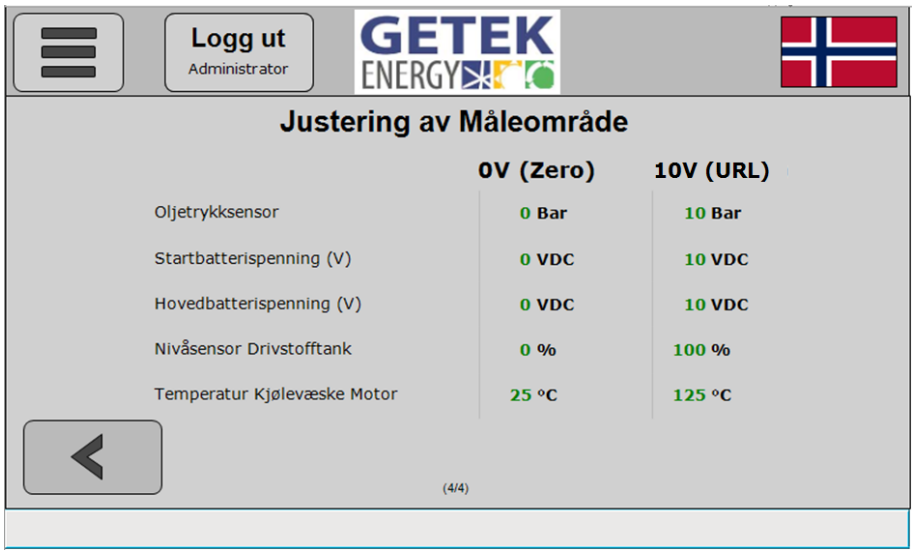
Når bruker trykker på et av elementene på vinduet dukker det opp et pop-up-vindu (figur 4.48). Bruker kan velge en inngang (AI 1 - 4), og trykke "Bekreft" for å lagre valget. Valgt inngang vil nå bli brukt for innhenting av data fra gjeldende sensor. Man kan også trykke "Avbryt" om man angrer seg, og gammel informasjon vil ikke bli overskrevet.




**Figur 4.48:** HMI-popup: Valg av analog input

### Avanserte Innstillinger 4

På den fjerde siden kan bruker justere måleområdet på følere knyttet til analogt inngangskort 750-459. Dette er de samme følerne som blir valgt på side 3 i avanserte innstillinger. Ved at dette er justerbart, kan følere på anlegget byttes til en annen føler med annet måleområde uten at programmet må skrives om. Merk at målesignalverdiens område er fast 0 til 10 V, og kan ikke endres på grunn av inngangskortet som er brukt.




**Figur 4.49:** HMI-vindu: Avanserte innstillinger side 4

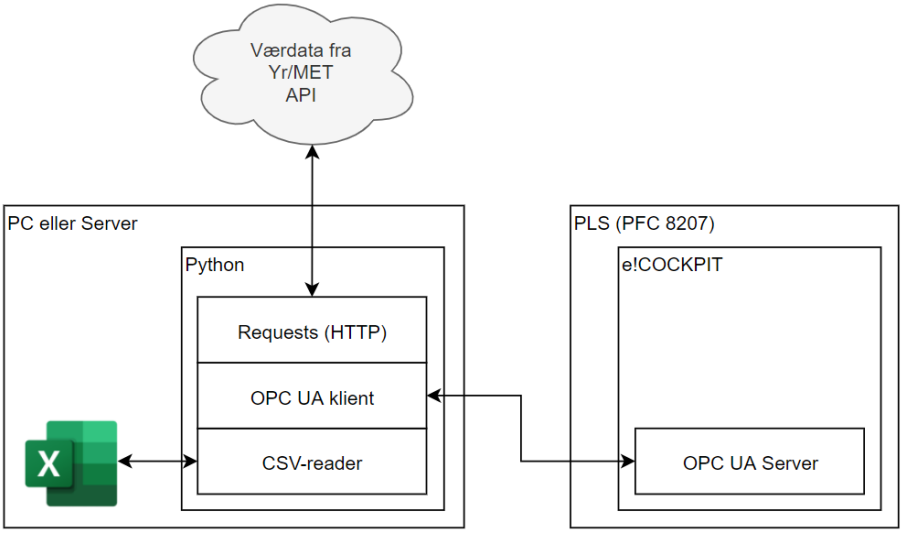
## 4.7 Batterilading vha. værprognose

Som bonus på oppgaven er det utviklet et system for værbasert lading av batteribanken. Det ble besluttet at værprognosene ikke trenger å oppdateres mer enn en gang i døgnet for hvert APS-anlegg. Det ble derfor utviklet en værprognoseapplikasjon i Python som kjøres eksternt på en datamaskin på kontoret, eller på en vilkårlig server. Denne applikasjonen leser av relevant informasjon fra en CSV-fil og henter værdata gjennom API fra internettsiden MET.no. CSV-filen kan åpnes i Microsoft Excel og inneholder nødvendig informasjon om APS-anleggene man skal sende til. Her kan det legges inn data til nærmest ubegrenset antall enheter i CSV-filen.

Hver rad i CSV-filen representerer ett APS-anlegg. Det kalkuleres hvor mye sol solcellepanelet vil motta i prosent, og hvor mye vind vindturbinen vil motta i prosent. Ved hjelp av dette, sammen med tidsperiode og nominelle effektverdier, blir det estimert hvor mye energi batteribanken vil motta fra disse kildene.

Kalkulasjon av sol er basert på skydekke. Programmet tar hensyn til variasjoner over døgnet. Dette følger en klokkekurve hvor 100% er ved klokka 12:00, og 5% ved minste punkt, altså midnatt. Programmet tar ikke hensyn til variasjoner over årstider. Kalkulasjon av vind er basert på vindstyrke og data relatert til montert vindturbin. Denne informasjonen må legges inn i CSV-filen. Det er her antatt et lineært forhold mellom vindturbinens effekt og vindstyrken, som er en grov forenkling.

Kalkulerte verdier av sol og vind blir sendt til mottagende PLS over kommunikasjonsprotokollen OPC UA. Denne protokollen må på forhånd aktiveres i PLS-ens WBM.




**Figur 4.50:** Værprognose: systemarkitektur

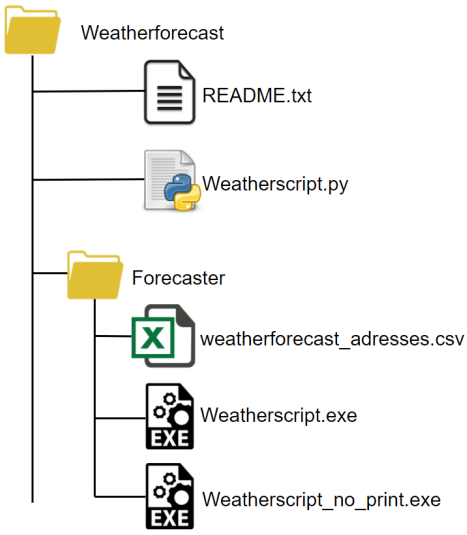
### 4.7.1 Mappestruktur

Mappestrukturen er relativt enkel. "README.txt" inneholder relevant informasjon til bruker. "Weatherscript.py" inneholder hele python-applikasjonen. Bruker trenger ikke røre dette skriptet, men det ligger ved slik at programmerere med erfaring kan vedlikeholde eller oppdatere skriptet. For å konvertere skriptet til exe-fil er det brukt "auto-py-to-exe" [28].

I mappen "Forecaster" ligger alt bruker trenger tilgang til. I "weatherforecast_adresses.csv" må bruker legge inn korrekt data. Det er viktig at denne filen ligger i samme mappe som exe-filene. Filen inneholder:

1. IP-adresse til mottakende PLS.
2. Koordinater i form av breddegrad, lengdegrad og høyde over havet.
3. Tidsvindu for værprognosen i antall timer fra nåtid.
4. Vindturbinens nominelle hastighet.
5. Vindturbinens innkoblingshastighet.

Merk at batterikapasitet og nominell effektverdi for solcellepanel og vindturbin stilles i brukergrensesnitt/HMI på PLS-en.




**Figur 4.51:** Værprognose: filstruktur

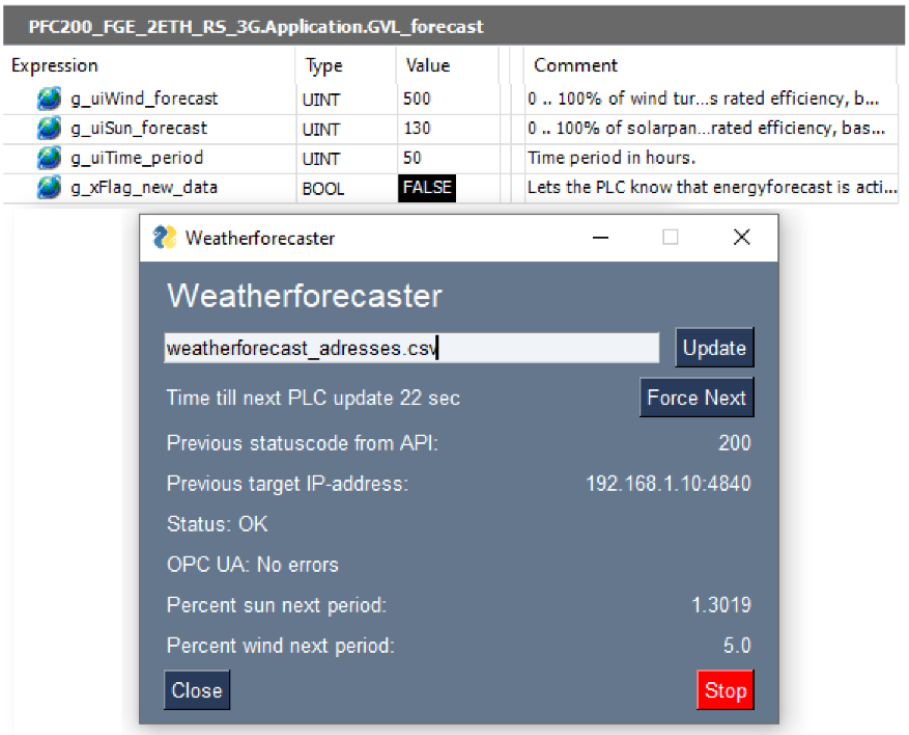
Filene "Weatherscript.exe" eller "Weatherscript_no_print.exe" åpner Python-applikasjonen. Den siste av disse åpner også Windows ledetekst, som kan være til hjelp under feilsøking. Disse to filene er "Weatherscript.py" konvertert til exe-fil. På denne måten trenger ikke bruker laste ned Python med tilhørende avhengigheter.

På figur 4.52 ser man et eksempel på brukergrensesnittet og tilhørende globale variabler på PLS-en. Prosent sol og vind er ganget med 100 før de blir sendt til PLS-en, for større oppløsning.

### 4.7.2 Pythonapplikasjonens GUI

Python-applikasjonen har sitt eget grafisk brukergrensesnitt (GUI) for å øke brukervennligheten. Til dette er det brukt biblioteket "pySimpleGUI" som er et svært enkelt bibliotek for utvikling av enkle brukergrensesnitt [24].

Øverst på brukergrensesnittet har bruker mulighet til å velge en annen CSV-fil. Ved oppstart vil filnavnet alltid settes til "weatherforecast_adresses.csv". Neste linje viser hvor lang tid til neste kalkulasjon og overføring av data. Bruker kan velge å trykke "Force Next" for å hoppe over ventetiden. Normalt vil applikasjonen vente 30 sekunder mellom hver APS-anlegg, eller rad i CSV-filen. Når applikasjonen har kjørt alle rader i CSV-filen, vil den vente et døgn før den leser samme CSV-fil på nytt.



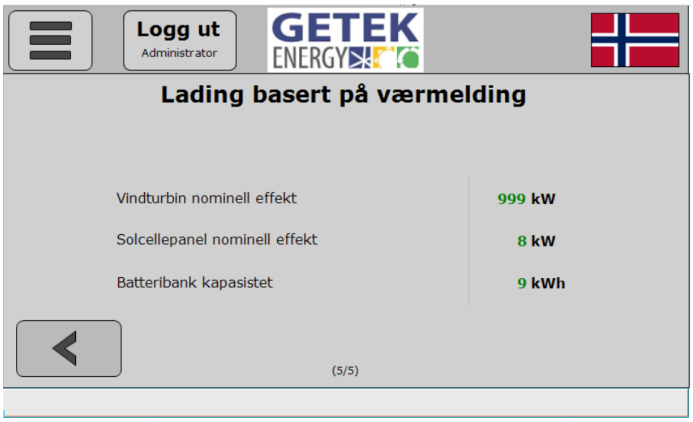

**Figur 4.52:** Værprognose: GUI med eksempel

Bruker kan velge å starte/stoppe applikasjonen ved hjelp av en knapp nede til høyre. Man kan også lukke applikasjonen, og tilhørende Windows ledetekst hvis åpen, ved å trykke "Close" nede til venstre. Ellers på bilde vises en del informasjon brukt under feilsøking, og for å overvåke om det forekommer kommunikasjonsfeil enten mellom applikasjon og API, eller applikasjon og OPC UA.

### 4.7.3 Værprognose i PLS-en

Denne delen av systemet kan dessverre ikke ferdigstilles da det ikke er tilgjengelig pålitelig måling av batterinivået i prosent, det resterende av denne delen er dermed kun en prototype. Av den grunn er værprognose-utvidelsen lagret i en egen e!COCKPIT-feil "Autopanel_V2.0.0", mens opprinnelig versjon uten værprognose er "Autopanel_V1.0.0".

PLS-en vil motta hvor mye sol og vind er forventet, i prosent. Disse verdiene er ganget opp med 100 og PLS-en må nedskalere disse verdiene. Den tar også inn tidsperiode i antall timer fra nåtid. I PLS brukergrensesnittet, på vindu "Avanserte Innstillinger" side 5 (Fig. 4.53), må de resterende dataene settes inn for lading med værprognose.




**Figur 4.53:** HMI-vindu: Værprognose

Programmet estimerer hvor mye energi batteribanken vil motta i neste tidsperiode ved hjelp av følgende formel:

$$E_{est} = VT_{nom} \cdot V_{fc} \cdot \Delta t + SP_{nom} \cdot S_{fc} \cdot \Delta t \tag{4.2}$$

- $E_{est}$ Estimert innkommende energi [kWh]
- $VT_{nom}$ Vindturbins nominelle effekt [kW]
- $SP_{nom}$ Solcellepanelets nominelle effekt [kW]
- $V_{fc}$ Utsettelse for vind [%]
- $S_{fc}$ Utsettelse for sol [%]
- $\Delta t$ Tidsperiode [h]

Videre kalkuleres estimert innkommende lading av batteriet for neste periode i prosent av total batterikapasitet.

$$Estimert\ lading[\%] = 100 \cdot \frac{E_{est}[kWh]}{Batteribankens\ kapasitet[kWh]} \tag{4.3}$$

Til slutt sammenlignes resultatet med prosessverdien, altså batterinivået i prosent, slik at PLS-en kan avgjøre om den skal akseptere forespørsel om lading fra eksternt startsignal. Det er her viktig å forhindre unødvendig hyppig start og stopp av aggregatet. Dette tas hånd om i programmet.

# 5. Diskusjon

Prosjektet har i hovedsak vært en software-basert oppgave, men det endte også opp med en del oppkobling på fysisk PLS. Mesteparten av arbeidstiden er brukt til programmering og simulering lokalt på PC-er, da den fysiske testingen på APS-en ikke kom før mot slutten av prosjektperioden. Siden både programvaren og programmering av brukergrensesnitt var nokså ukjent for hele gruppen, ble det brukt mye tid i starten på å lære seg de nye programmene.

I prosjektløpet har det vært flere utskiftninger i kontaktpersoner tilknyttet oppdragsgiver. Tidlig i prosjektet ble det informert at Tor Erik Næbb fra Wago skulle bytte jobb, og at Jim Roger Kvello skulle ta over hans stilling som gruppens kontaktperson i Wago. Senere kom det en mer overraskende endring hos Getek da Arnt Lysø gikk over til en ny jobb. Heldigvis har Helge Engebø fra Getek vært med hele veien og ble etter hvert hovedkontakten til gruppen. Utskiftningene har likevel ikke påvirket prosjektets progresjon i stor grad.

Hoveddelene av prosjektet ble konstruert hver for seg, og det har derfor vært essensielt å sørge for at de kunne bli slått sammen til slutt. Målinger fra aggregatet lest av i et skript, sendt videre til alarmdeteksjon i et annet skript og eventuelle aksjoner blir gjennomført i et tredje skript. Rekkefølgen på programkjøringen var derfor essensiell å ta hensyn til for at programmet skulle fungere som planlagt.

## Sekvens

Oppstartsekvensen som er implementert for dieselmotoren baserer seg på måling av oljetrykk, generatorspenning og generatorfrekvens. Med tanke på dette dukket det opp et problem med oppkoblingen av energimåleren som ble brukt i starten av prosjektet. Komponenten skulle drives av spenningen den målte, som i vårt tilfelle ikke ville fungere, da spenning og frekvens vil komme og gå hver gang aggregatet startes og stoppes. Det tok også flere sekunder under oppstart før denne energimåleren var klar til sending av målinger. Før dette ble oppdaget var det allerede lagt ned mye arbeid for å få kommunikasjonen gjennom Modbus til å fungere.

Da problemet med feil energimåler ble oppdaget startet leting etter en erstatning. Heldigvis hadde Support hos Wago gode råd, og etter leting i diverse datablad ble det funnet en god erstatning. Løsningen var å bytte ut den gamle Modbus RTU [18] baserte måleren med en modul i Wago sin 750 serie, tilpasset vår PLS. Dette gjorde innhenting av data en god del lettere, men det måtte istedenfor implementeres skalering av strøm, da den nye energimåleren krever bruk av strømtransformator. Denne måleren har en grense på maksimum én ampere over tid. Om dette ikke er tilstrekkelig har Wago også en tilsvarende måler som tåler fem ampere. DSE-autopanelet og sikringene på originalt anlegg er beregnet for strømmåling opp til fem ampere. Grunnet tidsbegrensninger og

leveringstid ble det én ampere varianten for testing, men ellers er fem ampere varianten å anbefale.

## SMS

Det ble informert om at de autonome strømforsyningene til Getek ofte befinner seg på krevende eller ugunstige steder. Det er dermed kostbart å gjennomføre unødvendige reiser til anleggene, ettersom det ofte kreves helikoptertransport. Ulike plasseringer av APS-anleggene fører også til at tilkobling til internett ikke er en selvfølge. SMS-kommunikasjon til systemet er derfor en nødvendighet. Fra Getek stiltes det krav til alarmvarsling og kvittering gjennom SMS, men det ble i tillegg implementert en rekke andre kommandoer for å minimere nødvendigheten av å være til stede fysisk. Etter fremvisning av SMS-konseptet for oppdragsgiver ble det konkludert med at denne løsningen var veldig brukervennlig.

For SMS-funksjonene dukket det opp et problem ved bruk av funksjonsblokken "WagoAppSerial_Sms.FbSimpleSms_8207" som ble brukt i starten av prosjektet. Hvis det oppstår en feil i blokken, for eksempel at et nummer som ikke eksisterer ble skrevet inn i i kontaktloggen, ville hele denne blokken henge seg opp. Man hadde da ikke mulighet til å sende nye kommandoer til blokken og SMS-systemet ville ikke fungere før PLS-en ble restartet.

Løsningen på dette problemet var å ta i bruk en annen, og mer generell funksjonsblokk for SMS, "WagoAppSerial_Sms.FbGsmSms". Denne blokken var mer kompleks, og tilbød flere funksjonaliteter. Den viktigste funksjonaliteten her var mulighetene for å kunne avbryte pågående oppgave i blokken, for så å hoppe videre til neste telefonnummer.

## Brukergrensesnitt

Det gamle autopanelet besto av et relativt enkelt brukerpanel med fysiske knapper. I dette prosjektet ble det tatt i bruk en 7.0" TP600-skjerm, og det var dermed mulighet for å utvide funksjonaliteten og legge til flere visuelle elementer. Ved bruk av en touchskjerm ble det enklere å lage et brukergrensesnitt som både er lett å bruke, og som viser frem den viktigste informasjonen. Skjermen er ganske liten, og ga tidlig i prosjektet en begrensning på mengden innhold man burde ha på et vindu, da lesbarheten og ryddigheten fort forsvinner jo mer man har på samme vindu. Likevel har den tilgjengelige plassen blitt godt utnyttet med både informativ tekst og dynamiske figurer, som ikke var tilgjengelig på det gamle DSE-panelet i det hele tatt.

Selv om man er tilkoblet til PC på DSE-panelet, er det mye begrensninger med denne løsning med tanke på å tilpasse informasjon i den innebygde webserveren. Dette er derimot enklere ved bruk av en PLS. Ved bruk av ulike typer motorer og følere, kreves det også ulik konfigurasjon i PLS-en. Det ble derfor bestemt å gjøre systemet mer allsidig ved å gjøre de fleste innstillingene valgfrie og redigerbare i brukergrensesnittet. En slik løsning unngår at bruker må inn i programkode for å kunne endre på innstillinger/parametere.

## Alarmsystemet

I alarmsystemet ble det i starten fokusert hovedsakelig på den innebygde alarmkonfiguratoren. Gjennom prosjektet ble det innsett at denne ikke inneholdt alle funksjonaliteten som var ønsket, og flere av de interne variablene var ikke mulig å hente i kode. Det ble dermed laget egne programmer som tok i bruk variabler knyttet til alarmkonfiguratoren, slik at man kunne manipulere disse som man selv ønsket.

Kvittering av alarmer var det største problemet som dukket opp i alarmsystemet. Den innebygde kvitteringen som alarmkonfiguratoren tilbydde, tilbakestilte kun den visuelle alarmvarslingen i alarmtabellen. Alarmbitene som ble satt i alarmdeteksjonen aktiverte disse varslingene i alarmtabellen. Disse alarmbitene var også direkte knyttet opp mot alarmaksjonene, slik at det var nødvendig at disse bitene forble høye til de hadde blitt kvittert. Løsningen ved å kun benytte seg av kvitteringsmetoden i alarmkonfiguratoren var derfor ikke tilstrekkelig.

For å sørge for at alarmbitene holdt seg høye, ble det implementert en egen kvitteringsmetode. Her ble det benyttet en egen kvitteringsbit innebygd i alarmobjektet. Den biten ble brukt som en inngang i funksjonsblokkene for å resette alarmbiten, og selve kvitteringen ble programmert i strukturert tekst. Med denne løsningen ble det unngått at systemet kunne starte før alarmene ble kvittert.

Det var også en utfordring å koble SMS-systemet sammen med innebygd alarmsystem. Det var relativt enkelt å hente forrige alarm som gikk høy og sende den over SMS. Problemet oppsto hvis flere alarmer slo inn samtidig. Ved å gå mer bort fra det innebygde alarmsystemet, og ved bruk av buffere i SMS-systemet, endte vi opp med et system der alle alarmer blir varslet selv når de går høy under samme programsyklus.

## Værprognose

Dette systemet er ikke ferdig utviklet og ikke testet på fysisk anlegg. Det er to årsaker til dette.

1. Det kreves pålitelig måling av batterinivået. Per nå er bare batterispenningen tilgjengelig.
2. Perfeksjonering av hovedproduktet skulle prioriteres fremfor dette, som definert i problemstilling.

Applikasjonen bygd i Python og kommunikasjon opp imot PLS-en er ferdig implementert og testet. Det som mangler i systemet er måling av batterinivået, ferdigstilling av program i e!Cockpit, og til slutt testing på et fysisk anlegg.

Siden startsignalet er eksternt og gruppen ikke har god nok oversikt over hvilken logikk som ligger bak dette signalet, har gruppen valgt å la værprognosen overkjøre startsignalet. Det vil si at hvis værprognosesystemet er aktivt i brukergrensesnittet på PLS-en, så må både eksternt startsignal og signal fra værprognosen være høy. Hvis en av disse er lav vil ikke lading forekomme. På denne måten unngår man lading av batteribanken når det er forventet mye energi-input i neste tidsperiode, slik at minst mulig energi blir tapt.

Det er også selvsagt en del usikkerhet i en værprognose fra meteorologisk institutt og alle har opplevd at værmeldingen ikke stemmer overens med det faktiske været. Det er derfor lagt inn en overstyring ved lavt batterinivå, slik at dieselaggregatet likevel får kjøre for å sikre tilgang på strøm.

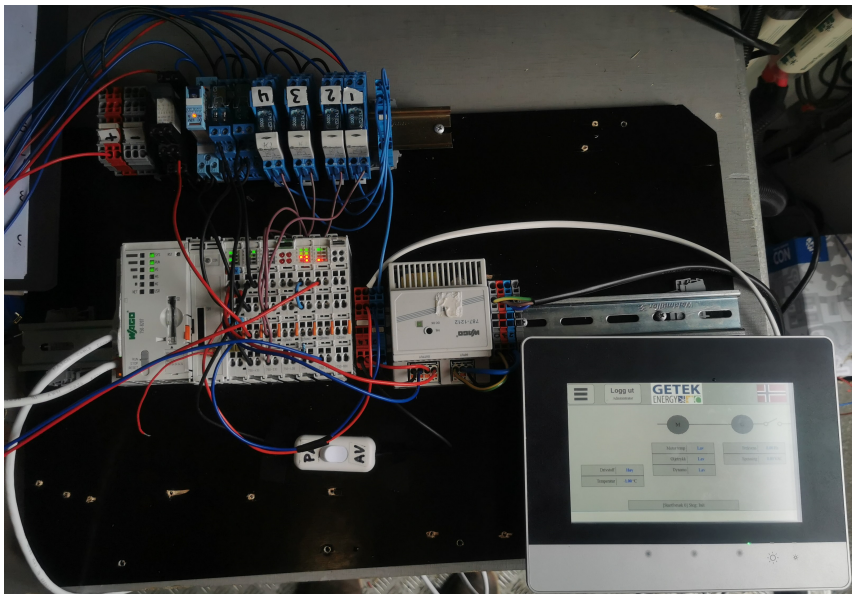
Siden programmet ikke tar hensyn til årstider, anbefaler gruppen å bruke et glidende gjennomsnitt av tidligere værprognoser, og korrigere dette med reelle data. Dette kan implementeres på PLS-en sine side.

Hovedmålet med værprognosen er å unngå at batteriet står fullt over lengre tid mens solcellepanel eller vindturbin prøver å mate energi. Samtidig er det viktig å unngå at aggregatet slår inn og ut hyppig, som vil føre til uønskede kostnader i form av slitasje av utstyr. Det er også viktig at værprognosen ikke fører til at batteriet tømmes helt. Gruppen mener de har oppfylt disse betingelsene, men av grunner nevnt tidligere er det ikke fullt testet enda.

## Fysisk testing

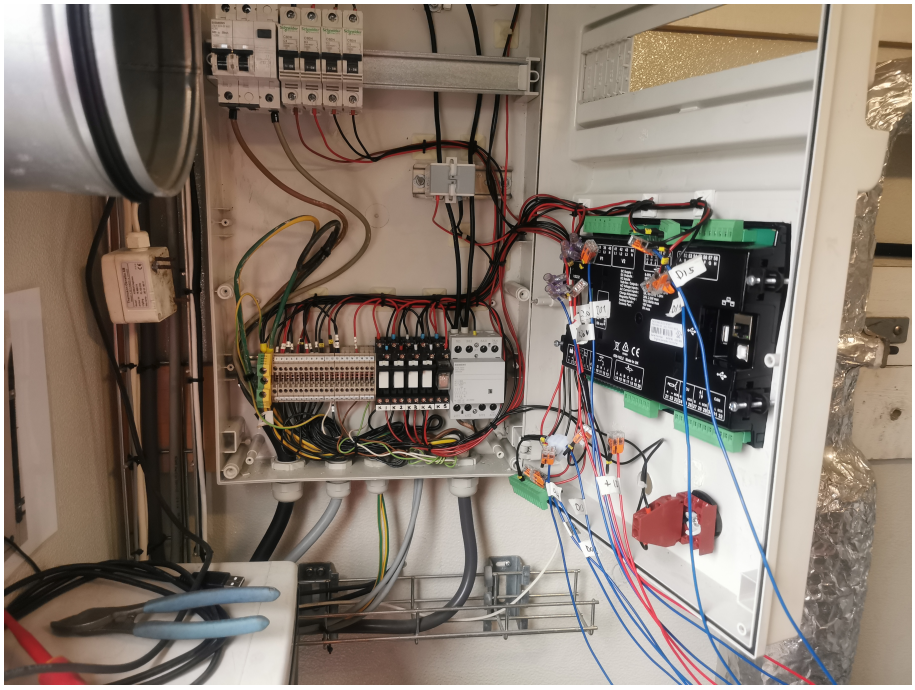
Mot slutten av bachelorperioden ble det tilgang til fysisk testing på en av oppdragsgiver sine APS-enheter. Det ble lagt en del arbeid i å sette seg inn i koblingsskjemaet for det eksisterende DSE-panelet, da det var nødvendig å koble om alle inn- og utganger fra skapet, til vår PLS. Ved bruk av releer ble oppkoblingen mulig, da DSE-panelet og utgangssignalene herfra ble drevet av 12 VDC, kontra PLS-en sin driftsspenning på 24 VDC. Under testingen ble PLS-en driftet av strømforsyning med 230 VAC fra omformeren i APS-anlegget. Merk: I et ferdig oppkoblet anlegget vil man drifte PLS-en fra batteribanken, med en DC til DC omformer for 24 VDC mellom, for å sikre jevn spenning. For oversikt over koblingen mellom PLS-en og APS-en, se vedlagt styrestrømskjema (Vedlegg B).

**Figur 5.1:** APS-anlegg for fysisk testing




**Figur 5.2:** Provisorisk oppkobling av PLS for fysisk test. Se vedlegg A for maskinvare på DIN-skinne.

Takket være mye test og feilsøking i forkant av fysisk oppkobling, ble den første testen på anlegget vellykket. Sekvensen fungerte som tenkt med en vellykket oppstart, drift og stopp av motoren. Alt som ble implementert i HMI-en og SMS fungerte også som det skulle. Det var veldig nyttig med en fysisk test for å se at ting fungerte som det skulle, etter mye simulering og testing ved hjelp av trykknapper og LED-lys som inn- og utganger. Etter første test ble det likevel gjennomført noen utbedringer i sekvensen, deriblant små tidsforsinkelser og mer bruk av oljetrykksmålingen under oppstart av motoren. Det måtte også justeres noen tidsforsinkelser for alarmer, blant annet for alarm [209] "Motor stopper ikke. Oljetrykk høyt".



**Figur 5.3:** Provisorisk oppkobling av PLS i skap for fysisk test.

Oljetrykksensoren på motoren, som var digital, ga høyt trykk umiddelbart motoren hadde startet, men ved en stopp gikk det omtrent 10 sekunder før den leste lavt oljetrykk. Dette gjorde at høyt oljetrykk ble lagt til som overgangsbetingelse i sekvensen for å stoppe kjøring av startmotoren under oppstart. Fra tidligere erfaring med større motorer ble det tatt utgangspunkt i en større tidsforsinkelse før oljetrykket ble angitt som høyt ved start og mindre forsinkelse før det ble erklært lavt etter stopp. Slike detaljer er det vanskelig å vite om uten fysisk testing.

Oppdragsgiver og veileder fikk en fysisk demonstrasjon av anlegget med gjennomgang av brukergrensesnitt, alarmsystem, SMS-system og aggregat-sekvens. Oppdragsgiver var særdeles fornøyd, spesielt med brukergrensesnittet og det integrerte SMS-systemet. Det kom tilbakemeldinger på at brukervennlighet, funksjonalitet og fleksibilitet ble løst veldig bra.

# 6. Administrativt

I forprosjektet ble det satt opp en grundig prosjektstyring i form av arbeidspakker, Gantdiagram, S-diagram og timeføring. Med unntak av arbeidspakkene, som er skrevet i Microsoft Word, er alt gjort i Microsoft Excel. Regnearket er designet på en dynamisk måte slik at arkene fungerer på tvers av hverandre. På denne måten slipper man å fylle inn samme data flere steder. I tillegg har gruppen brukt et annet Excel-ark internt for å holde oversikt på viktig informasjon i løpet av prosjektet. Dette innebærer informasjon som versjonslogg av programfiler, gjøremålslister, loggføring av spørsmål og svar, alarmoversikter, SMS-system og HMI-standarder.

Gantdiagrammet er en oversikt over arbeidsfordelingen i hele bachelorperioden, der arbeidspakkene spenner over bestemte uker. For noen av arbeidspakkene er det også satt opp spesifikke milepæler som skal ha blitt gjennomført til et bestemt tidspunkt. Ved å følge dette oppsettet er det sørget for at tidsplanen er blitt opprettholdt, og man derav unngår å komme i store tidsklemmer. Gantdiagrammet finner man i vedlegg I.3.

Gjennom prosjektet har gruppen oppdatert veileder og oppdragsgiver med regelmessige statusrapporter i form av PDF-dokumenter og PowerPoint-presentasjoner. Det er blitt gjennomført flere møter der det er blitt tatt opp nåværende progresjon, og spørsmål for å best mulig forme systemet etter oppdragsgivers krav. Veileder har også kommet med tilbakemeldinger under møtene, som også har blitt tatt i betraktning. Under møtene er det også tatt opp eventuelle avvik, og hvilke deler av prosjektet som er blitt satt i fokus for hver periode.

Alle relevante filer for bacheloroppgaven ligger i en prosjektadministrasjonsperm (PA). Denne blir levert som et vedlegg når prosjektet er ferdig. Dette er en digital mappestruktur med forprosjekt, arbeidspakker, avtaler, dokumenter fra møter, timeoversikt, framdriftsplan, brukermanual, dokumentasjon for anlegget, rapport og presentasjon med mer.

## 6.1 Timeoversikt

Gruppen har ført inn timer for hver uke i prosjektet inn i et Excel-ark for å illustrere progresjonen av oppgaven. Under timeføringen er det lagt til diagrammer som viser timer per uke, og diagrammer som viser totalt akkumulerte timer for hver student. Gruppen har dermed kontinuerlig fulgt med på hvor mye arbeid som er gjort hver uke.

For bacheloroppgaven er det satt et forventet timebruk på 550 timer per student fra NTNU, en totalsum på 2200 timer for hele gruppen. I starten av prosjektet gikk bacheloroppgaven parallelt med et annet fag, og de akkumulerte timene ble dermed noe mindre enn forventet. Dette er spesielt synlig i uke 10 og 11, da dette var under eksamenstiden. I tillegg ble det også gjort mindre arbeid i påskeferien i uke 15. Likevel ble timene tatt

igjen mot slutten av prosjektet, da det ble brukt mye tid på testing og rapportskriving både på kveldstid og i helgene.

De oppførte timene er visualisert i ett S-diagram som strekker seg over hele bachelorperioden. Med unntak av perioden rundt eksamen og påskeferie, har S-diagrammet gått i en relativt lineær kurve og ikke som en S. Dette kan anses som positivt, da en S-formet graf tyder på en ujevn progresjon med en svært hektisk periode i midten. S-diagrammet finner man i vedlegg I.4.

## 6.2 Økonomi

Den nye løsningen for et autopanel som er implementert i prosjektet krever også en annerledes økonomisk løsning. Løsningen som er implementert i dette prosjektet består av en rekke komponenter i forskjellige prisklasser, men da produktet tilsvarer kun en prototype så er det sannsynlig at prisen kan bli noe annerledes. Det er verdt å nevne at de fleste komponentene som er brukt i prosjektet kun er de som oppdragsgiver har hatt tilgjengelig, og at økonomi dermed ikke har vært et hovedfokus for gruppen.

Tabell 6.1 er en oversikt over den økonomiske løsningen i prosjektet. Merk at innholdet her er basert på Wago sine listepriser. Nederste rad gir et grovt overslag over pris med rabatt, basert på dialog med Wago.

| **Komponent** | **Element nr.** | **Antall** | **Pris [kr]** |
|---|---|---|---|
| Kontroller | 750-8207* | 1 | 11 345,00* |
| 4-kanals Analog Inngangskort (Resistansmåling) | 750-463 | 1 | 2 003,00 |
| 4-kanals Analog Inngangskort (0-10 V) | 750-459 | 1 | 1 741,00 |
| 8-kanals Digital Utgangskort (24 V) | 750-530 | 1 | 647,30 |
| 8-kanals Digital Inngangskort (24 V) | 750-430 | 1 | 573,60 |
| Energimåler | 750-494 | 1 | 3 046,25 |
| Endeblokk | 750-600 | 1 | 149,30 |
| Strømforsyning | 787-1212 | 1 | 904,70 |
| Touch Panel 600; (7,0"); Visu panel | 0762-4303/8000-0002 | 1 | 13 730,00 |
| **Sum** | | | **34 140,15** |
| **Sum, med antatt rabatt** | | | **26 354.21** |

**Tabell 6.1:** Tabell over den økonomiske løsningen benyttet i prosjektet. *Er erstattet med en nyere modell

Tabell 6.2 er en oversikt over den økonomiske løsningen som skal byttes ut. Prisen er omregnet fra GBP til NOK med valutakurs 11.98 fra 13.05.2022. Merk at dette er den rabatterte prisen som oppdragsgiver har betalt for autopanelet.

| **Komponent** | **Element nr.** | **Antall** | **Pris [kr]** |
|---|---|---|---|
| DSE MKII Autopanel | 7410-03 | 1 | 5350,93 |
| **Sum** | | | **5350,93** |

**Tabell 6.2:** Tabell over den økonomiske løsningen benyttet for dagens løsning.

DSE-panelet som oppdragsgiver bruker i dag er én enkel enhet. Oppdragsgiver betaler med dagens kurs 5350,93 kr (£444.96) for et DSE-panel, som er vesentlig billigere enn vår løsning. Prisen inkluderer ikke et eventuelt mobilmodem eller andre komponenter som Getek må ha i tillegg for å nærme seg funksjonalitetene dette prosjektet byr på. Igjen så har den økonomiske løsningen på prosjektet ikke vært gruppen sitt valg, og derfor heller ikke en prioritet på løsningen av oppgaven. Det må også tas i betraktning at denne kostnaden antageligvis blir nærmest neglisjerbar i forhold til prisen av et komplett APS-anlegg.

# 7. Konklusjon - Erfaring

Det er blitt utviklet et styresystem med brukergrensesnitt for et dieselaggregat ved hjelp av en PLS. Produktet består av mange deler som har blitt flettet sammen til et komplett system med mange funksjonaliteter. Det er brukt mye tid på å finne gode og sikre løsninger sammen med mye testing for å oppnå et pålitelig system.

Styringen starter med avlesninger fra følere som blir brukt til sekvensen, sjekking av alarmer og visualisering i HMI-en. Sekvensen styrer start og stopp av motoren og innkobling av last fra generatoren til lading av batteribank. Aggregat-sekvensen er kompleks og tar hensyn til mulige problemer under både start og stopp, og utfører alarmaksjoner ut ifra et spesialdesignet alarmsystem. Aggregatet er overvåket med et alarmsystem som skal beskytte aggregatet og elektronisk utstyr som er tilkoblet. Alarmsystemet skal også gi advarsler og informere i brukergrensesnittet og over SMS.

SMS-kommunikasjon utvider mulighetene og funksjonaliteten til styresystemet betraktelig. Ofte er oppdragiver sine APS-anlegg på utilgjengelige lokasjoner, uten personell i umiddelbar nærhet. Det vil da ofte bli kostbart å reise til anlegg. Opptil ti telefonnumre kan få alarmvarslinger når problemer oppstår, for å kunne starte feilsøking fortest mulig. Brukere kan benytte en rekke kommandoer for å kvittere alarmer, sjekke status og endre på kontaktlisten med mer. Dette utføres ved å sende SMS-er, med en brukervennlig kommunikasjonsprotokoll.

Brukergrensesnittet er stilrent og intuitivt med mange innstillingsmuligheter. Det er blitt satt fokus på brukervennlighet og fleksibilitet, med en stil i henhold til standard for "High Performance HMI" [13]. Uten innlogging kan aggregatet overvåkes, alarmer kan kvitteres og SMS varsling kan endres. Ved innlogging som administrator får brukeren tilgang på en rekke innstillinger for å tilpasse systemet. Alle alarmene kan også testes for å se at varsling og aksjoner fungerer som det skal. I avanserte innstillinger er det mulig å endre mange sensorinnstillinger; om de er tilgjengelige, digitale eller analoge, og hvilket måleområde de har.

Produktet er testet på fysisk anlegg og de ulike funksjonalitetene fungerte optimalt. Bachelorgruppen fikk skryt av oppdragsgiver for brukervennlighet, funksjonalitet og fleksibilitet. Prosjektgruppen har lykkes i å lage et system som fungerer etter de definerte kravene fra oppdragsgiver, inkludert funksjonaliteter utover kravene. Det konkluderes med at styringssystemet egner seg godt til bruk på en autonom strømforsyning.

## 7.1 Videreutvikling

### 7.1.1 Koble utstyr i APS-anlegget til PLS-en

Målet til oppdragsgiver er å kunne koble sammen mest mulig utstyr på en plattform, og samtidig gjøre systemet mer fleksibelt. Med en PLS installert i anlegget, vil det være mye enklere å koble sammen forskjellig utstyr. PLS-en har mange kommunikasjonsprotokoller og siden den er modulær kan det brukes mange tilkoblinger av forskjellige typer. Wago PLS-en har og skyløsning klart for bruk, så det passer godt å overvåke annet utstyr som blir koblet til. Neste steg for oppdragsgiver blir nok derfor å koble sammen mer av APS-anlegget til PLS-en.

### 7.1.2 Maskinvare

Autopanelet som erstattes bruker 12 VDC, og PLS fra Wago bruker 24 VDC. Gruppen har brukt mellomreleer til testing av programvare og styresystem. Bruk av mellomreleer fører til flere feilkilder og er ikke ønskelig for en permanent løsning, med mindre det er høyst nødvendig. Derfor bør styreskapet bygges om.

I tillegg anbefaler gruppen å bruke 1 A sikringer for strømmåling på energimålingskort (750-494). Eller bytte til et tilsvarende kort med 5 A, og bruke 5 A sikringer. Dette er i henhold til datablad og for å sikre kort og feltbus mot skade.

### 7.1.3 SMS-system

Det er i dette prosjektet utviklet et SMS-system som gir alarmvarslinger, og kan motta kommandoer i henhold til en skreddersydd kommunikasjonsprotokoll. Det er blitt lagt til kommandoer basert på dialog med arbeidsgiver, men det er mulighet for å legge til mange flere kommandoer. Her er mulighetene mange.

Telefonloggen kan holde opptil 10 kontakter. Denne er det mulig å utvide. I tillegg er SMS-filteret noe simpelt. En mulig utvidelse er å filtrere meldinger individuelt for alle kontakter. Dette vil for øvrig kreve mer minne.

### 7.1.4 Batterilading ved hjelp av værmelding

I dette prosjektet har det vært stort fokus på primærenergikilden av APS-systemet. På APS-en som er brukt her er det allerede montert solcellepanel på utsiden for lading av batteribanken. Gruppen har utviklet et enkelt program for værprognose, men som nevnt har ikke dette vært hovedfokus og kan utvikles videre.

Til værprognosen er det utviklet en Python-applikasjon som kan kjøre fra en PC, for eksempel fra kontoret. Kommunikasjon mellom Python-applikasjonen og APS-anleggenes

PLS-er skjer over kommunikasjonsprotokollen OPC UA. Det er mulig Wago Cloud gir bedre muligheter for kommunikasjon, og kanskje et bedre brukergrensesnitt. Gruppen har ikke brukt tid på Wago Cloud da en annen gruppe har som hovedoppgave å se på dette, på vegne av samme arbeidsgiver.

Det mangler en pålitelig måling på batterinivået til hovedbatteriet. I dette prosjektet er det kun batterispenningen som har vært tilgjengelig fra PLS-en. Dette er den største grunnen til at værprognoseprogrammet ikke er ferdig implementert og testet. Gruppen har sett på saken, men konkluderte med at det å konvertere måling av batterispenning til en pålitelig måling av batterinivået er mye vanskeligere enn først antatt. Gruppen anbefaler her å kjøpe en egen komponent fra en leverandør som er spesialisert innenfor dette.

### 7.1.5 Skylagring og datavisualisering

Siden systemet kun består av et lite styrepanel er det ikke lagt til mye visualisering av data i HMI-programmet i form av grafer og liknende. I PLS-en er det likevel mye data som er lagret som kan bli brukt og visualisert ved hjelp av skylagring. Wago har en egen skylagringsfunksjonalitet, Wago Cloud, som kunne vært benyttet, men dette ble ikke prioritert da en annen bachelorgruppe hadde dette som hovedoppgave. En videre utvidelse av vår oppgave kan dermed være å bruke de relevante dataene i PLS-en som alarmhistorikk og målinger, og sende disse dataene over til skyen.

Siden Geteks APS-anlegg befinner seg på lite tilgjengelige steder kan datalogging over skytjenester være et sterkt verktøy å bruke under feilsøking. Teknikker kan da analysere historisk data i forkant av fysisk oppmøte på anlegg for feilsøking. I enkelte tilfeller sparer man seg kanskje for en tur ut med teknikker. I tillegg kan innsamlet data ved hjelp av programvare, regresjon eller en simpel form av kunstig intelligens brukes til å avgjøre om service bør foretas tidligere enn antatt. Her er det altså utrolig mye man kan gjøre, og det å analysere data kan være en hel bacheloroppgave i seg selv.

Eksempel på data som ligger klar i programmet, som kan sendes til Wago Cloud:

- Data knyttet til energimåler. Strøm, spenning, energi, effektfaktor osv.
- Antall startforsøk siste 20 ganger. Lagret i variabel "aiStart_tries_trend".
- Årsak for de 10 siste "start feil" alarmene. Lagret i variabel "asRecent_start_fails"
- Driftstimer: totalt, forrige uke, forrige døgn, siden service.
- Status og verdier på følere.

# Referanser


[1] e!cockpit libraries - overview and migration notes. Fra: https://usermanual.wiki/Document/ecockpit20manual.952137147.pdf. Hentet: 08.05.2022.

[2] What is a web server? Fra: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server, 2022. Hentet: 17.05.2022.

[3] Software AG. Opc ua intergration. Fra: https://www.softwareag.com/en_corporate/resources/what-is/opc-ua.html, 2022. Hentet: 15.05.2022.

[4] Geir Andresen. Aggregat (maskinteknikk). Fra: https://snl.no/aggregat_-_maskinteknikk, 2019. Hentet: 03.05.2022.

[5] Øyvind Ellingsdalen Arthur Sartsen. Dieselmotor. Fra: https://snl.no/dieselmotor. Hentet: 04.05.2022.

[6] Getek AS. Fra: https://getek.no/. Hentet: 30.04.2022.

[7] Fedak W. Anweiler S. Ulbrich R. Jarosz B. The concept of autonomous power supply system fed with renewable energy sources. Fra: http://dx.doi.org/10.13044/j.sdewes.d5.0160, 2017. Hentet: 04.05.2022.

[8] John Baechyel. How diesel engines work: Explaining the function of compression ignition engines. Fra: http://blog.diamondracing.net/how-does-a-diesel-engine-work. Hentet: 08.05.2022.

[9] Martin Bruggink. Programming plcs using sequential function chart. Fra: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.48.8416&rep=rep1&type=pdf, 1999. Hentet: 30.04.2022.

[10] Codesys. Remanente variabler. Fra: https://help.codesys.com/api-content/2/codesys/3.5.12.0/en/_cds_vartypes_retain_persistent/. Hentet: 05.05.2022.

[11] International Electrotechnical Commission. IEC 61131-3:2013 international standard. Fra: https://webstore.iec.ch/preview/info_iec61131-3%7Bed3.0%7Db.pdf, 2013. Hentet: 03.05.2022.

[12] Led Controls. Fra: https://www.ledcontrols.co.uk/finder-relay-base-8-pin-for-46-series-relays-socket-panel-or-35-mm-rail-mount.html, 2022. Hentet: 19.05.2022.

[13] Gregory Duranso. High performance hmi. Fra: https://realpars.com/high-performance-hmi/. Hentet: 11.05.2022.

[14] Norsk elektroteknisk norm. Elektriske lavspenningsinstallasjoner. Fra: https://www.nek.no/wp-content/uploads/2018/06/Preview-NEK4002018no.pdf, 2018. Hentet: 06.05.2022.

[15] Ung Energi. Hvordan virker en generator? Fra: https://ungenergi.no/energibaerere/elektrisitet/generator/. Hentet: 04.05.2022.

[16] Computer Hope. Bus. Fra: https://www.computerhope.com/jargon/b/bus.htm, 2021. Hentet: 14.05.2022.

[17] Ole Christian Lingjærde. Innføring i klasser. Fra: https://www.uio.no/studier/emner/matnat/ifi/IN1900/h18/ressurser/slides/lec16oct2018.pdf, 2018. Hentet: 15.05.2022.

[18] Modbus. Modbus. Fra: https://modbus.org/. Hentet: 10.05.2022.

[19] NEK Norsk Elektroteknisk Komitè. *Maskinsikkerhet Maskiners elektriske utrustning Del 1: Generelle krav*. Number 5. utgave. 2006. Hentet: 16.05.2022.

[20] Overleaf. Norwegian university of science and technology on overleaf. Fra: https://www.overleaf.com/edu/ntnu. Hentet: 05.05.2022.

[21] Codesys Help Page. Continous function chart (CFC). Fra: https://help.codesys.com/api-content/2/codesys/3.5.15.0/en/_cds_f_programming_language_cfc/. Hentet: 03.05.2022.

[22] Codesys Help Page. Data types. Fra: https://help.codesys.com/api-content/2/codesys/3.5.12.0/en/_cds_struct_reference_datatypes/#standard-data-types. Hentet: 03.05.2022.

[23] The Latex Project. Latex – a document preparation system. Fra: https://www.latex-project.org/. Hentet: 03.05.2022.

[24] PySimpleGUI. Python guis for humans. Fra: https://pysimplegui.readthedocs.io/en/latest/. Hentet: 14.05.2022.

[25] Ashley Senior. Dse7410 mkii dse7420 mkii operator manual, dn057-263. Fra: https://www.deepseaelectronics.com/genset/manual-auto-start-control-modules/dse7410-mkii. Hentet: 16.05.2022.

[26] Engineering Spirit. How a diesel generator works -animation. Fra: https://www.youtube.com/watch?v=vqhM6GDidiA&ab_channel=ENGINEERINGSPIRIT. Hentet: 04.05.2022.

[27] Unitronics. What is the definition of "plc"? Fra: https://www.unitronicsplc.com/what-is-plc-programmable-logic-controller, 2022. Hentet: 30.04.2022.

[28] Brent Vollebregt. Auto py to exe. Fra: https://pypi.org/project/auto-py-to-exe/. Hentet: 14.05.2022.

[29] WAGO. Produktside til wago. Fra: https://www.wago.com/no/produkter. Hentet: 30.04.2022.

[30] Wikipedia. Interpolasjon. Fra: https://no.wikipedia.org/wiki/Interpolasjon. Hentet: 09.05.2022.

[31] Wikipedia. Microsoft excel. Fra: https://en.wikipedia.org/wiki/Microsoft_Excel. Hentet: 13.05.2022.

# A. Vedlegg - Maskinvareoversikt

WAGO Smart Designer

https://configurator.wago.com

**Project: Getek APS-anlegg demo**

**Configuration: APS-anlegg**

Path: /Getek APS-anlegg demo

Link: https://configurator.wago.com/index.html?function=loadproject&projectid=309733&date=20220517154953

Version: A [17.05.2022 - 16:49:53 (CET 17.05.2022 - 15:49:53)]

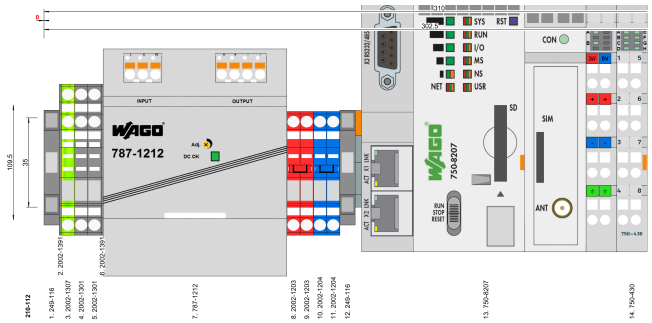
## Comments:

Inkluderer ikke Touch-Skjerm (TP 600, 0762-4303/8000-0002). Denne må også monteres men er ikke med her da den ikke er DIN-skinnemontert.

Småmateriell som rekkeklemmer kan avvike noe fra det som er brukt i demo. Har her forøkt å finne utstyr som passer bedre.

Strømforsyningsmodulen bør antageligvis erstattes i en permanent løsning.

## Preview:




Please note: The illustrations are simplified and may not be strictly true to reality, so that there may be display inaccuracies such as overlays. The same applies to technical data, e.g. dimensions, color or weight, for which we cannot vouch.

## Product Data:

| | |
|---|---|
| Dimensions (L x W x H) | 310 x 109.5 x 72.6 |
| Weight | 989.5 g |
| Length | 310 mm |
| Guide dimension | Start projection |
| Item number | 210-112 |
| Start projection | 0 mm |
| End projection | 0 mm |

## Project data:

| | |
|---|---|
| Project description | Eksempel demo utstyr for erstatning av autopanel.<br>Strømforsyning (230VAC til 24VDC) bør antageligvis erstattes. |

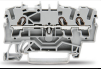
## Configuration data:

| | |
|---|---|
| Created by | E2210 |
| Customer material number | APS-anlegg |

## APS-anlegg

Link: https://configurator.wago.com/index.html?function=loadproject&projectid=309733&date=20220517154953 (Version: A [17.05.2022 - 16:49:53 (CET 17.05.2022 - 15:49:

### Parts list APS-anlegg (Path: /Getek APS-anlegg demo)

| Item quantity | Order quantity | Item Number | Description | Item image |
|---|---|---|---|---|
| 2 | 2 | 2002-1203 | 2-conductor through terminal block; 2.5 mm²; suitable for Ex e II applications; side and center marking; for DIN-rail 35 x 15 and 35 x 7.5; Push-in CAGE CLAMP®<br>**red** | |
| 2 | 2 | 2002-1204 | 2-conductor through terminal block; 2.5 mm²; for Ex e II and Ex i applications; side and center marking; for DIN-rail 35 x 15 and 35 x 7.5; Push-in CAGE CLAMP®<br>**blue** | |
| 2 | 2 | 2002-1301 | 3-conductor through terminal block; 2.5 mm²; suitable for Ex e II applications; side and center marking; for DIN-rail 35 x 15 and 35 x 7.5; Push-in CAGE CLAMP®<br>**gray** |  |
| 1 | 1 | 2002-1307 | 3-conductor ground terminal block; 2.5 mm²; suitable for Ex e II applications; side and center marking; for DIN-rail 35 x 15 and 35 x 7.5; Push-in CAGE CLAMP®<br>**green-yellow** | |
| 2 | 2 | 2002-1391 | End and intermediate plate; 0.8 mm thick<br>**gray** | |
| 1 | 1 | 2002-402/000-005 | Jumper; 2-way; insulated<br>**red** | |
| 1 | 1 | 2002-402/000-006 | Jumper; 2-way; insulated<br>**blue** | |
| 1 | 1 | 210-112 | Steel carrier rail; 35 x 7.5 mm; 1 mm thick; 2 m long; slotted; according to EN 60715; "Hole width 25 mm<br>**silver-colored 310 mm** | |
| 3 | 3 | 249-116 | Screwless end stop; 6 mm wide; for DIN-rail 35 x 15 and 35 x 7.5<br>**gray** | |
| 1 | 1 | 750-430 | 8-channel digital input; 24 VDC; 3 ms<br>**light gray** | |
| 1 | 1 | 750-459 | 4-channel analog input; 0 ... 10 VDC; Single-ended<br>**light gray** | |
| 1 | 1 | 750-463 | 4-channel analog input; For Pt1000/RTD resistance sensors; Adjustable<br>**light gray** | |
| 1 | 1 | 750-494 | 3-Phase Power Measurement; 480 VAC, 1 A<br>**light gray** | |
| 1 | 1 | 750-530 | 8-channel digital output; 24 VDC; 0.5 A<br>**light gray** | |
| 1 | 1 | 750-600 | End Module<br>**light gray** | |
| 1 | 1 | 750-8207 | Controller PFC200; 2 x ETHERNET, RS-232/-485, Mobile Radio Module article cannot be ordered anymore<br>**light gray** | |
| 1 | 1 | 787-1212 | Switched-mode power supply; Compact; 1-phase; 24 VDC output voltage; 2.5 A output current; DC-OK LED | |

*Note: The required order quantity may differ from the amount for some articles due to the form of delivery.*

## APS-anlegg

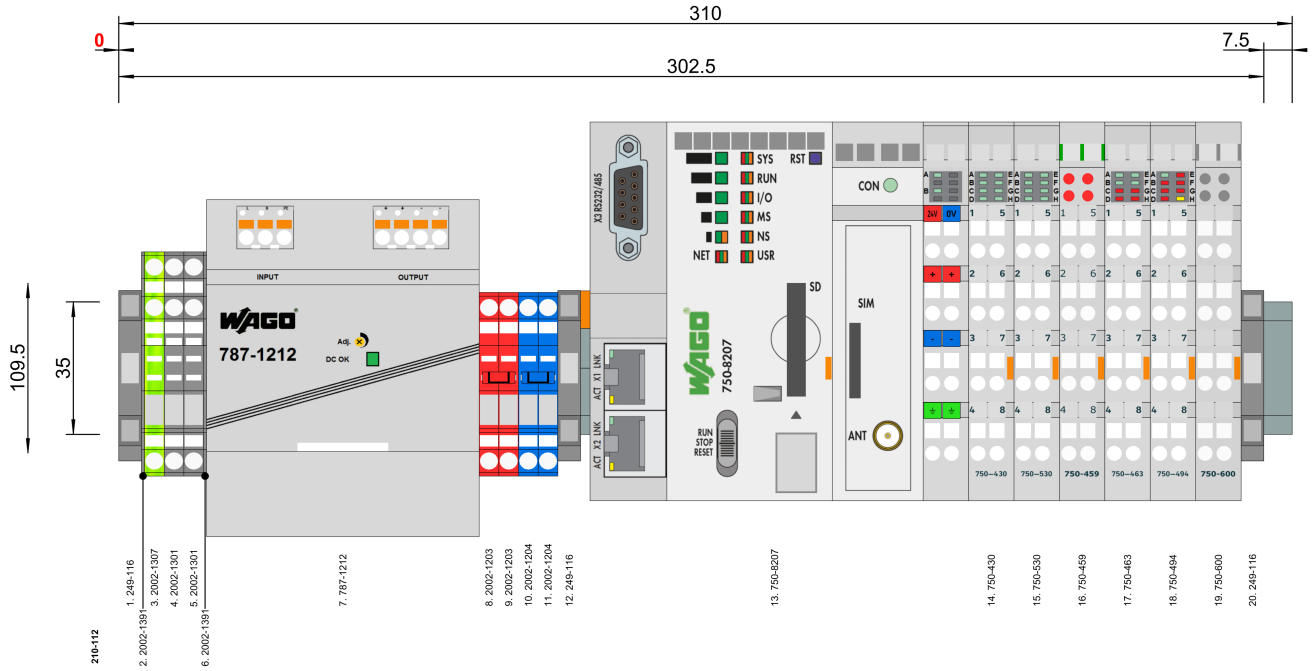
Link: https://configurator.wago.com/index.html?function=loadproject&projectid=309733&date=20220517154953 (Version: A [17.05.2022 - 16:49:53 (CET 17.05.2022 - 15:49:

### Listing of the components according to position APS-anlegg (Path: /Getek APS-anlegg demo)

Carrier rail

| Item quantity | Item Number | Notes | Description |
|---|---|---|---|
| 1 | 210-112 | | Steel carrier rail; 35 x 7.5 mm; 1 mm thick; 2 m long; slotted; according to EN 60715; "Hole width 25 mm<br>**silver-colored 310 mm** |

| Pos. | Item Number | Notes | Description |
|---|---|---|---|
| 1. | 249-116 | | Screwless end stop; 6 mm wide; for DIN-rail 35 x 15 and 35 x 7.5<br>**gray** |
| 2. | 2002-1391 | | End and intermediate plate; 0.8 mm thick<br>**gray** |
| 3. | 2002-1307 | | 3-conductor ground terminal block; 2.5 mm²; suitable for Ex e II applications; side and center marking; for DIN-rail 35 x 15 and 35 x 7.5; Push-in CAGE CLAMP®<br>**green-yellow** |
| 4. | 2002-1301 | | 3-conductor through terminal block; 2.5 mm²; suitable for Ex e II applications; side and center marking; for DIN-rail 35 x 15 and 35 x 7.5; Push-in CAGE CLAMP®<br>**gray** |
| 5. | 2002-1301 | | 3-conductor through terminal block; 2.5 mm²; suitable for Ex e II applications; side and center marking; for DIN-rail 35 x 15 and 35 x 7.5; Push-in CAGE CLAMP®<br>**gray** |
| 6. | 2002-1391 | | End and intermediate plate; 0.8 mm thick<br>**gray** |
| 7. | 787-1212 | | Switched-mode power supply; Compact; 1-phase; 24 VDC output voltage; 2.5 A output current; DC-OK LED |
| 8. | 2002-1203 | | 2-conductor through terminal block; 2.5 mm²; suitable for Ex e II applications; side and center marking; for DIN-rail 35 x 15 and 35 x 7.5; Push-in CAGE CLAMP®<br>**red** |
| 8.1. | 2002-402/000-005 | | Jumper; 2-way; insulated<br>**red** |
| 9. | 2002-1203 | | 2-conductor through terminal block; 2.5 mm²; suitable for Ex e II applications; side and center marking; for DIN-rail 35 x 15 and 35 x 7.5; Push-in CAGE CLAMP®<br>**red** |
| 9.1. | 2002-402/000-005 | | Jumper; 2-way; insulated<br>**red** |
| 10. | 2002-1204 | | 2-conductor through terminal block; 2.5 mm²; for Ex e II and Ex i applications; side and center marking; for DIN-rail 35 x 15 and 35 x 7.5; Push-in CAGE CLAMP®<br>**blue** |
| 10.1. | 2002-402/000-006 | | Jumper; 2-way; insulated<br>**blue** |
| 11. | 2002-1204 | | 2-conductor through terminal block; 2.5 mm²; for Ex e II and Ex i applications; side and center marking; for DIN-rail 35 x 15 and 35 x 7.5; Push-in CAGE CLAMP®<br>**blue** |
| 11.1. | 2002-402/000-006 | | Jumper; 2-way; insulated<br>**blue** |
| 12. | 249-116 | | Screwless end stop; 6 mm wide; for DIN-rail 35 x 15 and 35 x 7.5<br>**gray** |
| 13. | 750-8207 | | Controller PFC200; 2 x ETHERNET, RS-232/-485, Mobile Radio Module article cannot be ordered anymore<br>**light gray** |
| 14. | 750-430 | | 8-channel digital input; 24 VDC; 3 ms<br>**light gray** |
| 15. | 750-530 | | 8-channel digital output; 24 VDC; 0.5 A<br>**light gray** |
| 16. | 750-459 | | 4-channel analog input; 0 ... 10 VDC; Single-ended<br>**light gray** |
| 17. | 750-463 | | 4-channel analog input; For Pt1000/RTD resistance sensors; Adjustable<br>**light gray** |
| 18. | 750-494 | | 3-Phase Power Measurement; 480 VAC, 1 A<br>**light gray** |
| 19. | 750-600 | | End Module<br>**light gray** |
| 20. | 249-116 | | Screwless end stop; 6 mm wide; for DIN-rail 35 x 15 and 35 x 7.5<br>**gray** |




Comments:

Inkluderer ikke Touch-Skjerm (TP 600, 0762-4303/8000-0002). Denne må også monteres men er ikke med her da den ikke er DIN-skinnemontert.

Småmateriell som rekkeklemmer kan avvike noe fra det som er brukt i demo. Har her forøkt å finne utstyr som passer bedre.

Strømforsyningsmodulen bør antageligvis erstattes i en permanent løsning.

Confidential! The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization is prohibited. An offence is punishable and offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Use: | | | | 989.5 g | A4 | | |
| | | | | | | | |
| | | | | Origin | Repl. f.: | Repl. by: | |
| | | | | | Output | | |
| | | | | | Date | | |
| | | | | | APS-anlegg | | |
| Index | Revision no. | Date | Name | | | | |
| 1:2 | Created by | | | E2210 | Configuration | | |
| | Checked | | | | APS-anlegg | | |
| | Released by | | | | | | |
| | | Date | | Name | | | |
| | | en | | | | | |
| | | ID | | | | A | 4 / 4 |

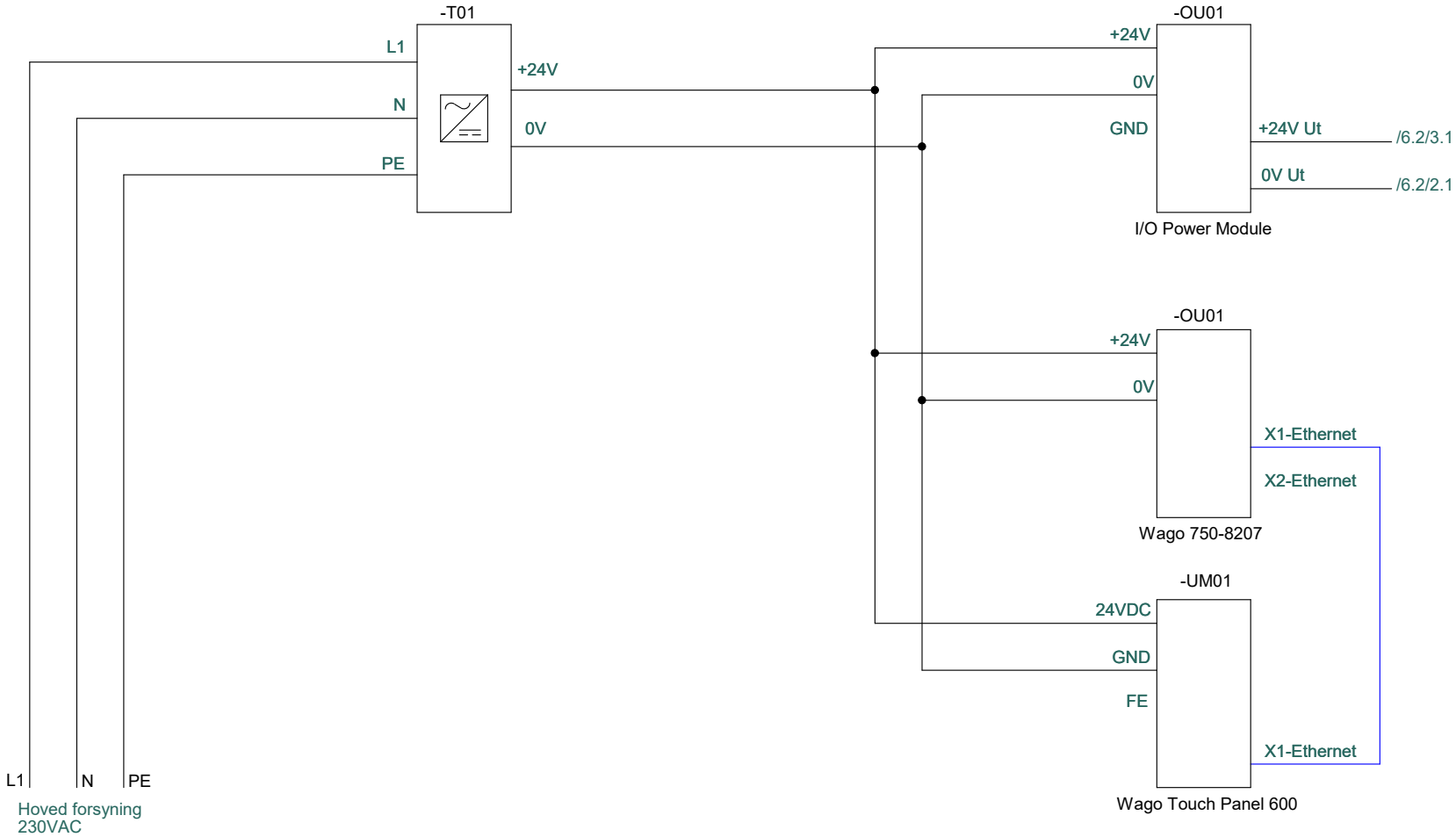
WAGO Smart Designer

https://configurator.wago.com/index.html?function=loadproject&projectId=309733&date=20220517154953 (Version: A [17.05.2022 - 16:49:53 (CET 17.05.2022 - 15:49:53)])

# B. Vedlegg - Styrestrømsskjema

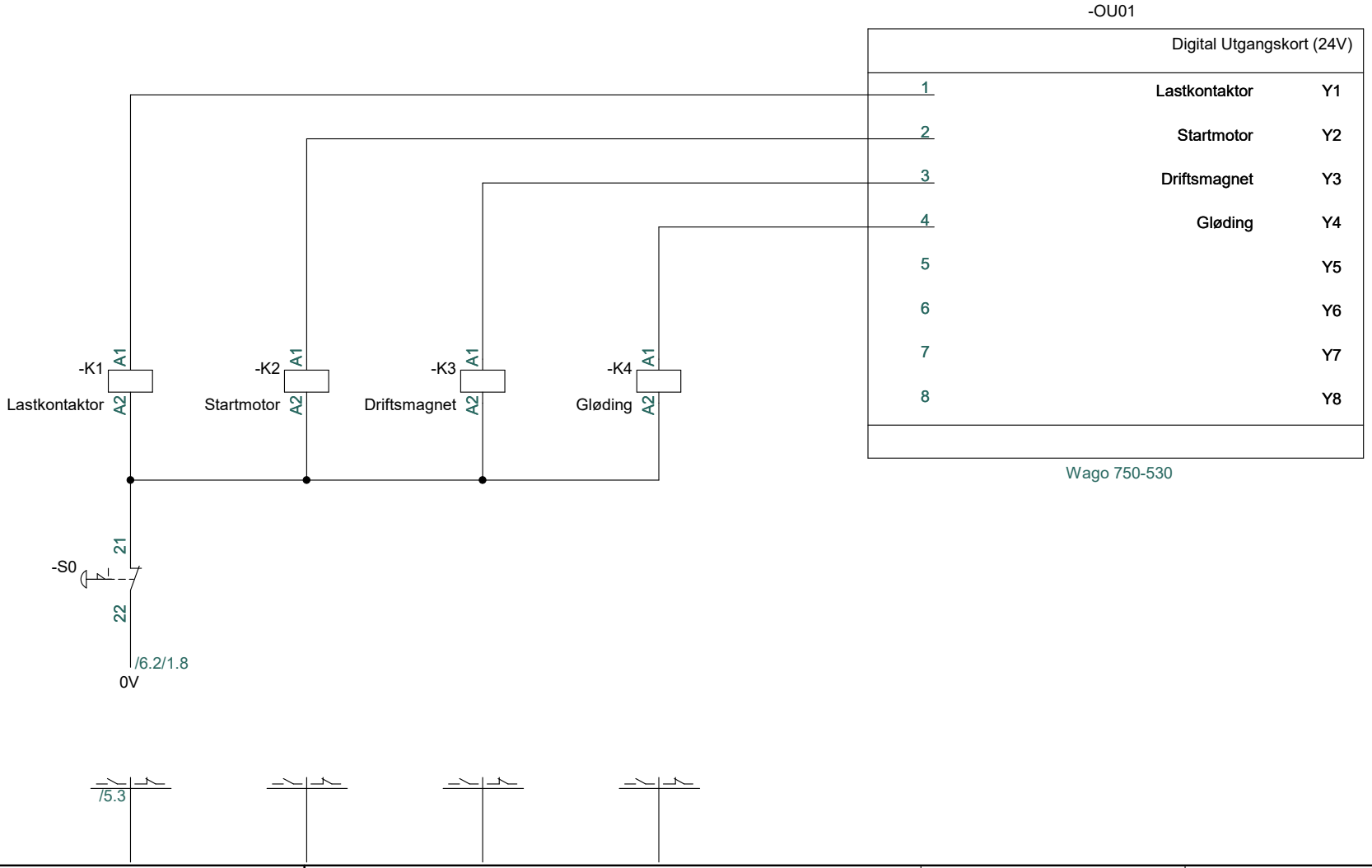
Det eksisterende autopanelet som skulle erstattes brukte 12 VDC forsyning og til digitale innganger og utganger. Den nye styringen bruker 24 VDC. Ved oppkobling og testing av denne oppgaven ble det brukt en rekke mellomreleer. Dette er ikke en god permanent løsning da den ikke er kostnadseffektiv og i tillegg økes antall feilkilder. Utstyret bør derfor plukkes ut av skapet for å montere nye skinner og bygge om. Det er ikke fastsatt rekkeklemmer, ettersom antall sensorer kan variere. Styrestrømsskjema er basert på den informasjon har oppgitt.

Merk at hvilke innganger som er i bruk varierer fra anlegg til anlegg. Det er for eksempel en rekke sensorer som kan være analog, digital eller ikke tilkoblet i det heletatt.



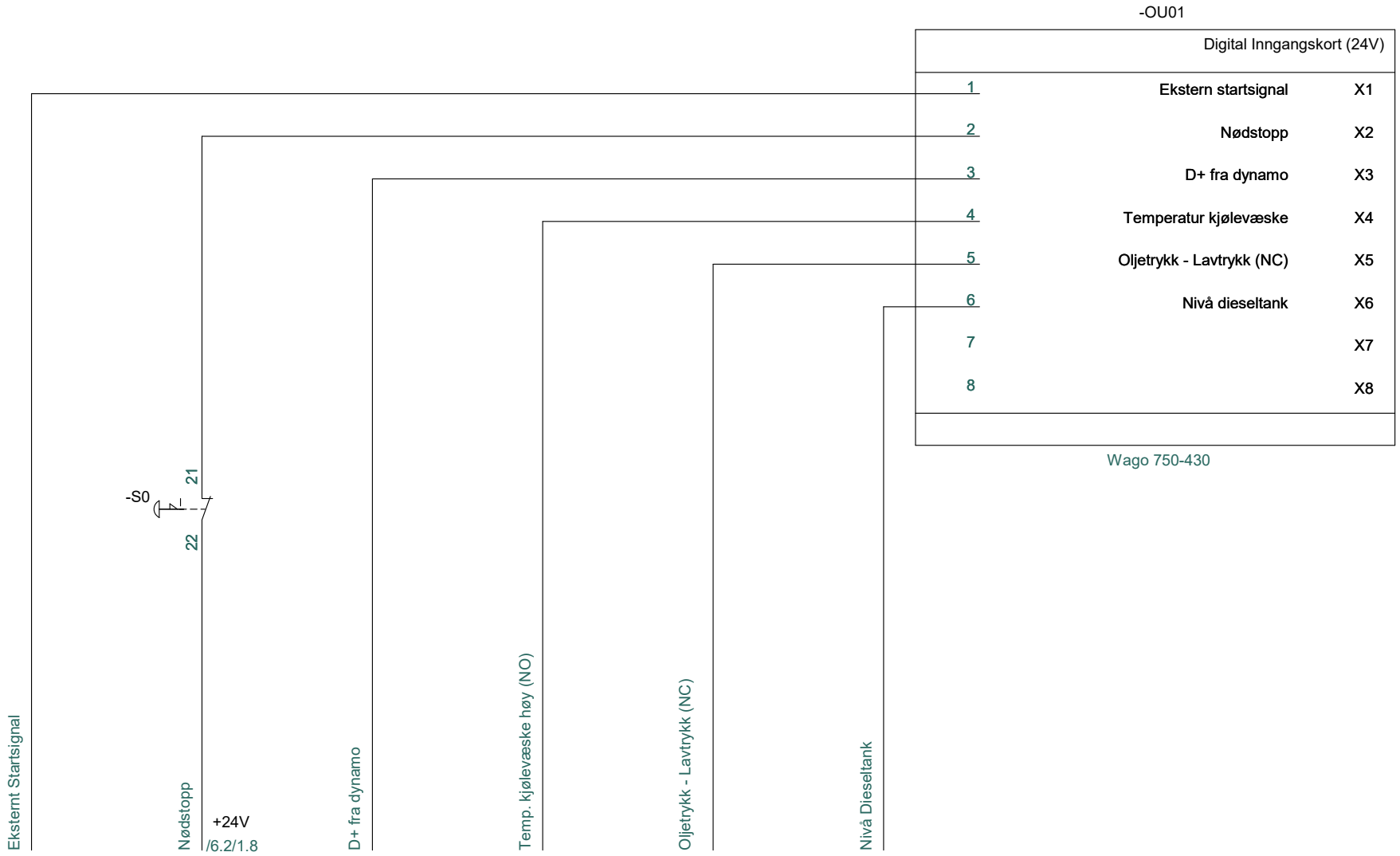

PCSCHEMATIC Automation

| | | | | | |
|---|---|---|---|---|---|
| **Prosjekttittel:** | | **Prosjekt nr.:** | Prosjektrev.: | Side: | 1 |
| Kunde: | Getek AS | DCC: | | Målestokk: | 1:1 |
| Sidetittel: | /@1 | Tegningsnr.: | Siderev.: | Forrige side: | |
| Filnavn: | Styrestrømsskjema | Konstr. (proj./side): | Sist utskrevet: 13/05/2022 | Neste side: | 2 |
| Sideref.: | | Godkj. (dato/init.): | Sist rettet: 13/05/2022 | Totalt antall sider: | 8 |



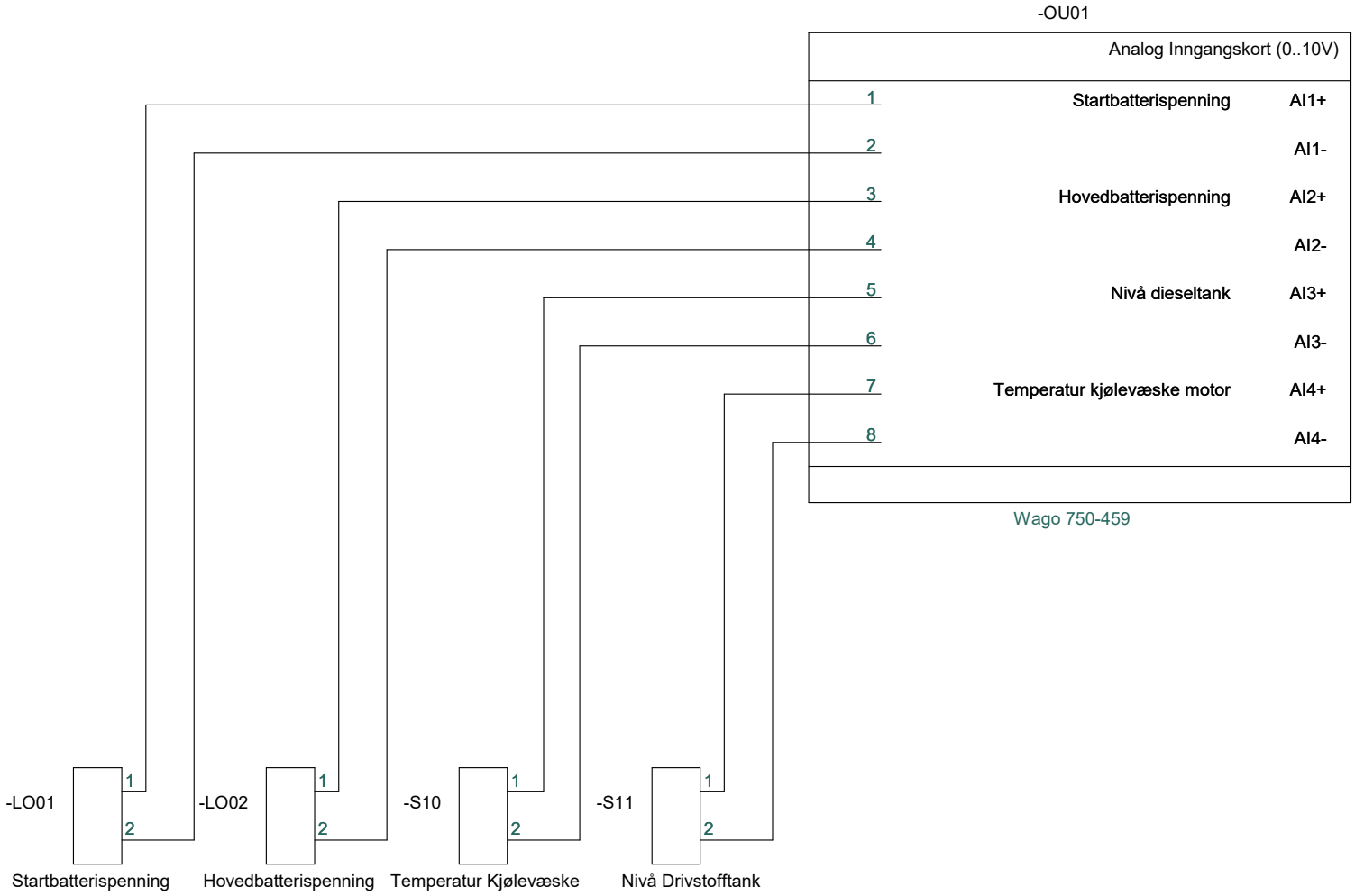

PCSCHEMATIC Automation

| | | | | | |
|---|---|---|---|---|---|
| **Prosjekttittel:** | | **Prosjekt nr.:** | Prosjektrev.: | Side: | **2** |
| Kunde: | Getek AS | DCC: | | Målestokk: | 1:1 |
| Sidetittel: | /@1 | Tegningsnr.: | Siderev.: | Forrige side: | 1 |
| Filnavn: | Styrestrømsskjema | Konstr. (proj./side): | Sist utskrevet: 13/05/2022 | Neste side: | 3 |
| Sideref.: | | Godkj. (dato/init.): | Sist rettet: 26/04/2022 | Totalt antall sider: | 8 |



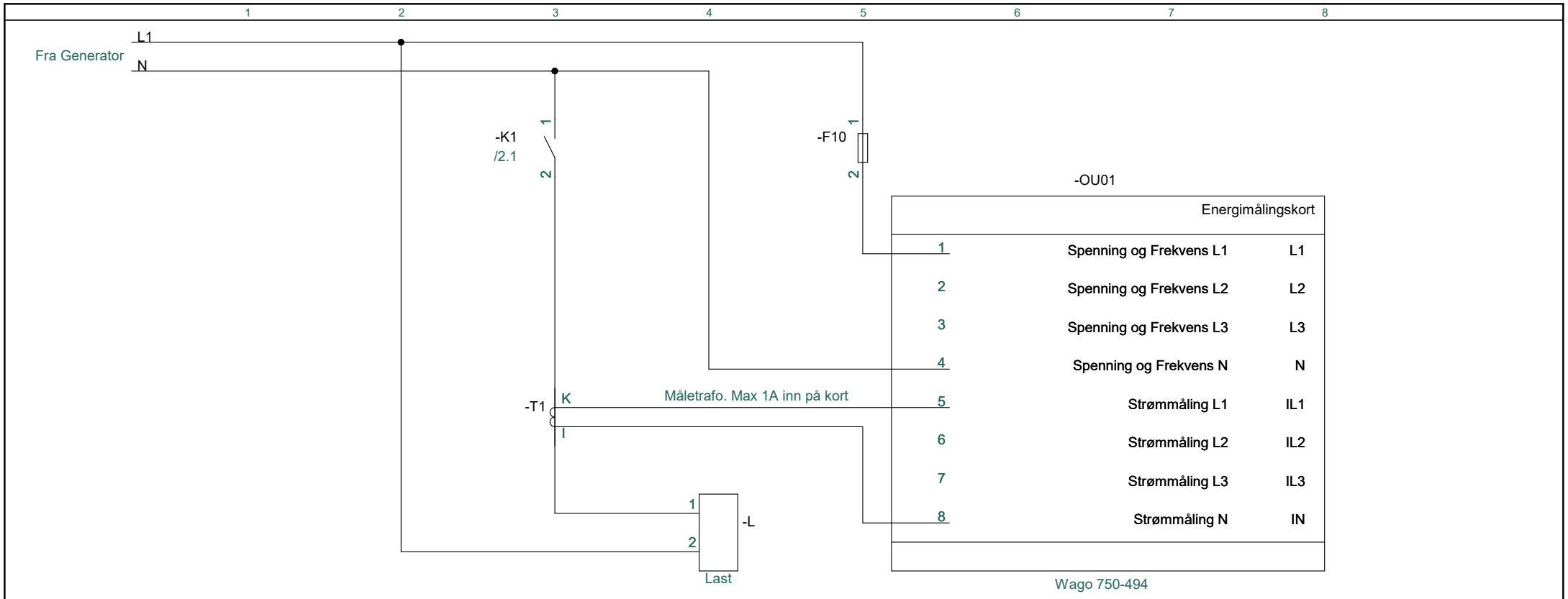

PCSCHEMATIC Automation

| | | | | |
|---|---|---|---|---|
| **Prosjekttittel:** | | **Prosjekt nr.:** | Prosjektrev.: | Side: 3 |
| Kunde: | Getek AS | DCC: | | Målestokk: 1:1 |
| Sidetittel: | /@1 | Tegningsnr.: | Siderev.: | Forrige side: 2 |
| Filnavn: | Styrestrømsskjema | Konstr. (proj./side): | Sist utskrevet: 13/05/2022 | Neste side: 4 |
| Sideref.: | | Godkj. (dato/init.): | Sist rettet: 26/04/2022 | Totalt antall sider: 8 |



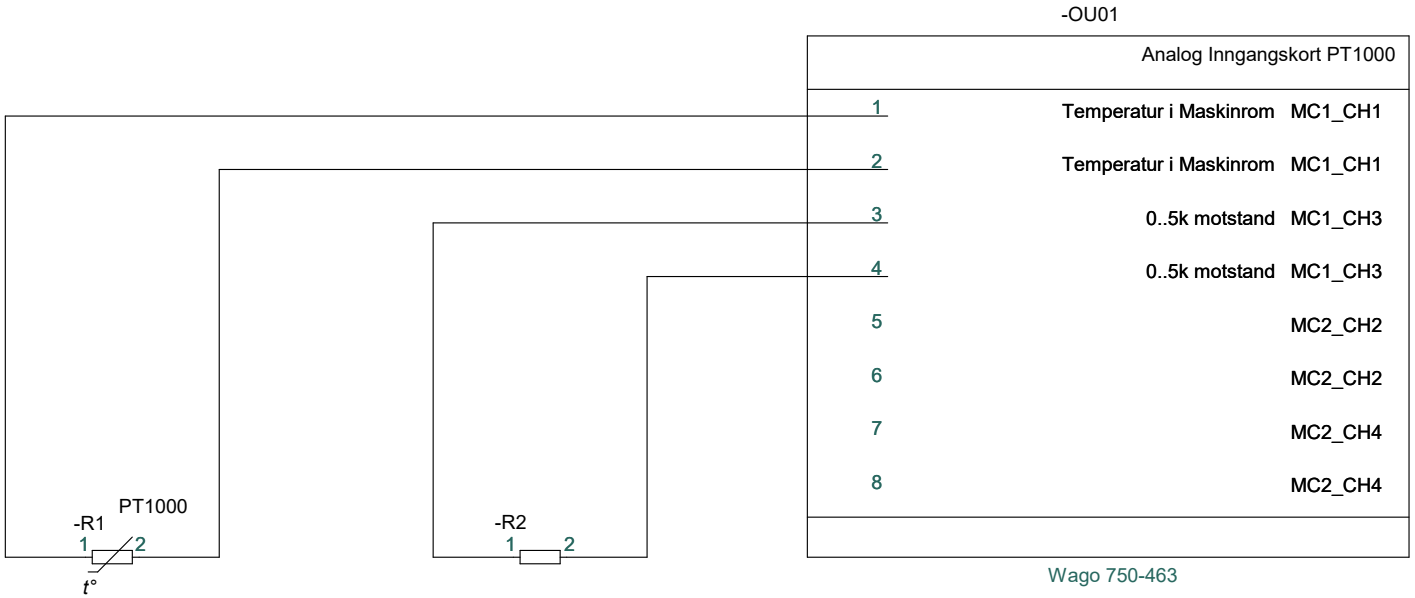

PCSCHEMATIC Automation

| | | | | | |
|---|---|---|---|---|---|
| **Prosjekttittel:** | | **Prosjekt nr.:** | Prosjektrev.: | Side: | **4** |
| Kunde: | Getek AS | DCC: | | Målestokk: | 1:1 |
| Sidetittel: | /@1 | Tegningsnr.: | Siderev.: | Forrige side: | 3 |
| Filnavn: | Styrestrømsskjema | Konstr. (proj./side): | Sist utskrevet: 13/05/2022 | Neste side: | 5 |
| Sideref.: | | Godkj. (dato/init.): | Sist rettet: 21/04/2022 | Totalt antall sider: | 8 |




PCSCHEMATIC Automation

| | | | | | |
|---|---|---|---|---|---|
| **Prosjekttittel:** | | **Prosjekt nr.:** | Prosjektrev.: | Side: | **5** |
| Kunde: | Getek AS | DCC: | | Målestokk: | 1:1 |
| Sidetittel: | /@1 | Tegningsnr.: | Siderev.: | Forrige side: | 4 |
| Filnavn: | Styrestrømsskjema | Konstr. (proj./side): | Sist utskrevet: 13/05/2022 | Neste side: | 6 |
| Sideref.: | | Godkj. (dato/init.): | Sist rettet: 13/05/2022 | Totalt antall sider: | 8 |




PCSCHEMATIC Automation

| | | | | | |
|---|---|---|---|---|---|
| **Prosjekttittel:** | | **Prosjekt nr.:** | Prosjektrev.: | Side: | **6** |
| Kunde: | Getek AS | DCC: | | Målestokk: | 1:1 |
| Sidetittel: | /@1 | Tegningsnr.: | Siderev.: | Forrige side: | 5 |
| Filnavn: | Styrestrømsskjema | Konstr. (proj./side): | Sist utskrevet: 13/05/2022 | Neste side: | 8 |
| Sideref.: | | Godkj. (dato/init.): | Sist rettet: 21/04/2022 | Totalt antall sider: | 8 |

| FROM | TO | Wire no | Signal | Cable | Jumper | Line name | Line type | Routing class |
|---|---|---|---|---|---|---|---|---|
| -S0:1 | -OU01:2 | | | | | | | |
| -K1:A1 | -OU01:1 | | | | | | | |
| -K1:A2 | -K2:A2 | | 0V | | | | | |
| -K2:A2 | -K3:A2 | | 0V | | | | | |
| -K3:A2 | -K4:A2 | | 0V | | | | | |
| -OU01:0V Ut | -K1:A2 | | 0V | | | | | |
| -R1:1 | -OU01:1 | | | | | | | |
| -K2:A1 | -OU01:2 | | | | | | | |
| -R1:2 | -OU01:2 | | | | | | | |
| -LO01:1 | -OU01:1 | | | | | | | |
| -LO01:2 | -OU01:2 | | | | | | | |
| -T01:L1 | -F10:1 | | L1 | | | | | |
| -T01:N | -L:2 | | N | | | | | |
| -K1:1 | -L:2 | | N | | | | | |
| -K1:1 | -OU01:4 | | N | | | | | |
| -K1:2 | -L:1 | | | | | | | |
| -K3:A1 | -OU01:3 | | | | | | | |
| -T1:K | -OU01:5 | | | | | | | |
| -T1:I | -OU01:8 | | | | | | | |
| -LO02:1 | -OU01:3 | | | | | | | |
| -LO02:2 | -OU01:4 | | | | | | | |
| -T01:+24V | -OU01:+24V | | | | | | | |
| -OU01:+24V | -OU01:+24V | | | | | | | |
| -UM01:24VDC | -OU01:+24V | | | | | | | |
| -T01:0V | -OU01:0V | | | | | | | |
| -OU01:0V | -OU01:0V | | | | | | | |
| -UM01:GND | -OU01:0V | | | | | | | |
| -R2:1 | -OU01:3 | | | | | | | |

| FROM | TO | Wire no | Signal | Cable | Jumper | Line name | Line type | Routing class |
|---|---|---|---|---|---|---|---|---|
| -K4:A1 | -OU01:4 | | | | | | | |
| -R2:2 | -OU01:4 | | | | | | | |
| -S10:1 | -OU01:5 | | | | | | | |
| -S10:2 | -OU01:6 | | | | | | | |
| -F10:2 | -OU01:1 | | | | | | | |
| -S11:1 | -OU01:7 | | | | | | | |
| -S11:2 | -OU01:8 | | | | | | | |
| -OU01:+24V Ut | -S0:2 | | +24V | | | | | |
| -UM01:X1-Ethernet | -OU01:X1-Ethernet | | | | | | | |

# C. Vedlegg - I/O-liste

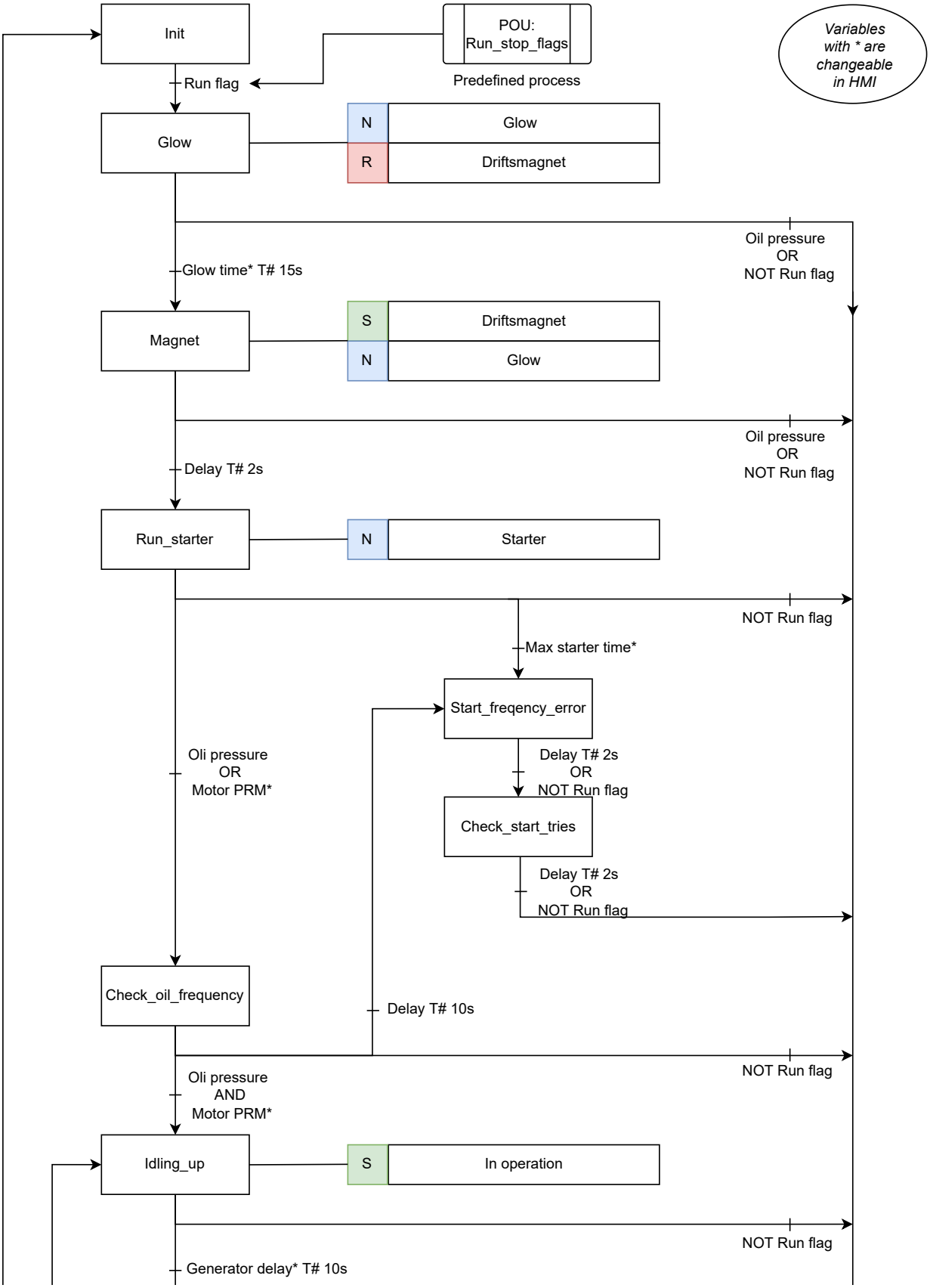
Dato: _____

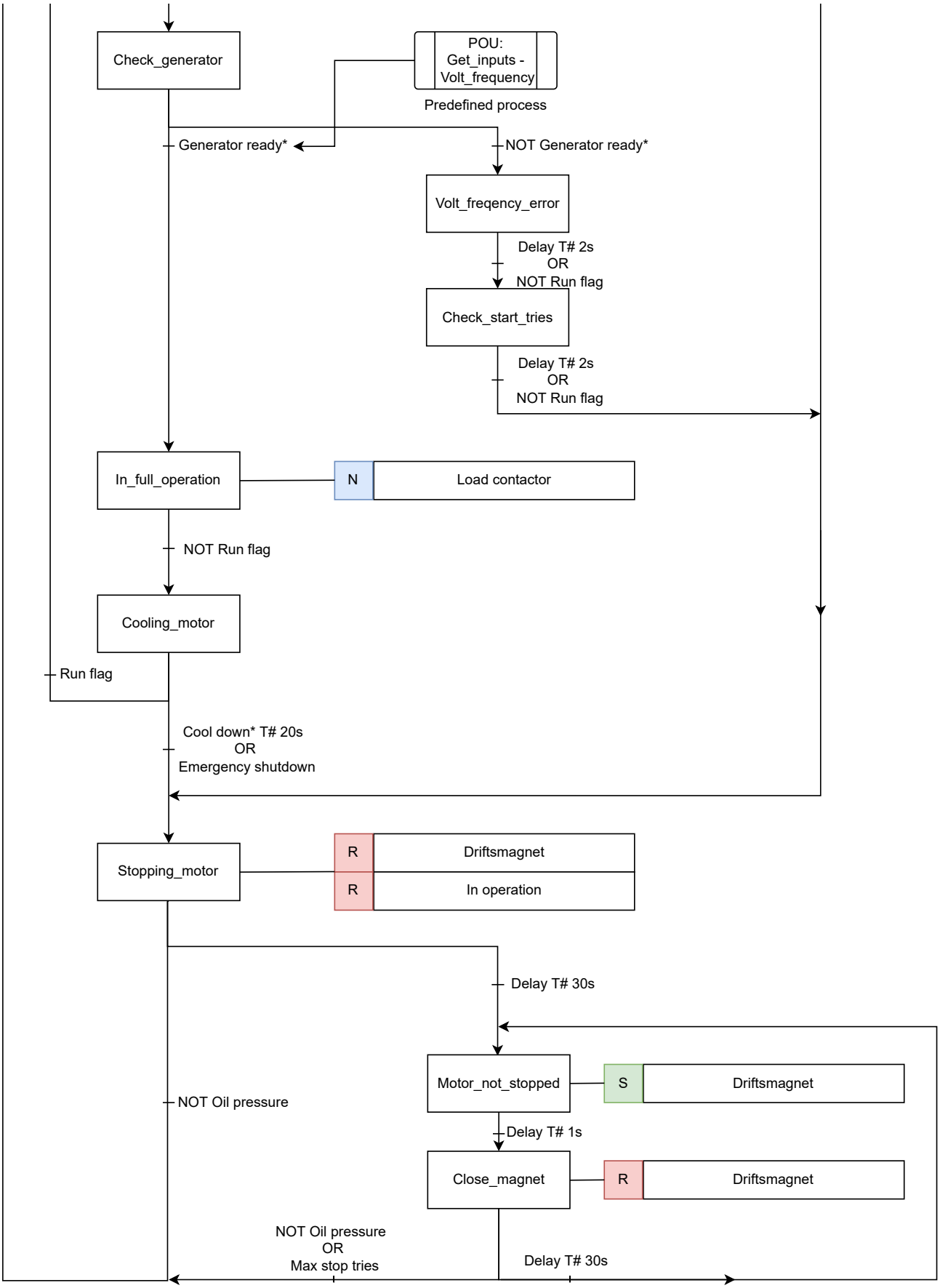
***WAGO PLS: IO-Liste***

*Elementer med blått merke I venstre kolonne er enten Analog eller Digital. Dette varierer fra anlegg til anlegg og kan settes I HMI.*
*Hvilken sensor som går til hvilket input kan settes manuelt I Software/HMI (Advanced Settings). Verdier I tabell er initielle verdier.*

| **Benevnelse** | **Type** | **Komponent** | **PLS** | | | |
|---|---|---|---|---|---|---|
| | | | **Kort** | **Klemme** | **Adresse** | **Variabel (Map)** |
| Spenning (VAC) og Frekvens (Hz) generator | AI | Energimåler | 750-494 | 1 (L1) & 4 (N) | | |
| Strøm generator (A) | AI | Energimåler | 750-494 | 5 (IL1) & 8 (IN) | | |
| Lastkontaktor slå på | DO | Kontaktor | 750-530 | 1 | %QX0.0 | xDO1 |
| Startmotor slå på | DO | Innslag startmotor | 750-530 | 2 | %QX0.2 | xDO2 |
| Driftsmagnet slå på | DO | Driftsmagnet | 750-530 | 3 | %QX0.4 | xDO3 |
| Gløding slå på | DO | Rele gløding | 750-530 | 4 | %QX0.6 | xDO4 |
| Ekstern startsignal / stoppsignal | DI | VR | 750-430 | 1 | %IX1.0 | xDI1 |
| Nødstopp | DI | Bryter | 750-430 | 2 | %X1.2 | xDI2 |
| D+ fra dynamo (usikker) | DI | Dynamo | 750-430 | 3 | %X1.4 | xDI3 |
| Temperatur kjølevæske motor - høy temp (NO) | DI | Sensor temp. | 750-430 | 4 | %X1.6 | xDI4 |
| Oljetrykk - Lavtrykk (NC) | DI | Sensor oljetrykk | 750-430 | 5 | %X1.1 | xDI5 |
| Nivå dieseltank | DI | Nivåsensor | 750-430 | 6 | %X1.3 | xDI6 |
| Oljetrykk | AI | Sensor oljetrykk | 750-459 | INAKTIV | | |
| 12VDC spenning (startbatteri) | AI | Målerverdiomf. | 750-459 | 1 & 5 | %IW1 | wAI1 |
| Hovedbatterispenning | AI | Målerverdiomf. | 750-459 | 2 & 6 | %IW2 | wAI2 |
| Nivå dieseltank | AI | Nivåsensor | 750-459 | 3 & 7 | %IW3 | wAI3 |
| Temperatur kjølevæske motor - høy temp | AI | Sensor temp. | 750-459 | 4 & 8 | %IW4 | wAI4 |
| Temperatur i maskinrom (PT1000) | PT1000 | Sensor | 750-463 | 3 & 4 | %IW5 | wAI_RTD3 |

# D. Vedlegg - Sekvensdiagram




Check_generator
POU:
Get_inputs -
Volt_frequency
Predefined process
Generator ready*
NOT Generator ready*
Volt_freqency_error
Delay T# 2s
OR
NOT Run flag
Check_start_tries
Delay T# 2s
OR
NOT Run flag
In_full_operation
N
Load contactor
NOT Run flag
Cooling_motor
Run flag
Cool down* T# 20s
OR
Emergency shutdown
Stopping_motor
R
Driftsmagnet
R
In operation
Delay T# 30s
Motor_not_stopped
S
Driftsmagnet
NOT Oil pressure
Delay T# 1s
Close_magnet
R
Driftsmagnet
NOT Oil pressure
OR
Max stop tries
Delay T# 30s

# E. Vedlegg - Kommunikasjonsprotokoll



| Kommunikasjonsprotokoll | | |
|---|---|---|
| **Melding Inn** | **Aksjon** | **Returnerer melding** |
| **Fra sender til PLS** | | **Eksempel:** |
| ACK ALL | Kvitterer alle alarmer. | All alarms has been Ackd |
| ACK {kode} | Kvitterer gitt alarmkode. Eksempel: ACK 200 kvitterer alarmkode 200 | The following alarm has been Acknowledged: {Alarmtekst fra alarmkonfigurator} |
| Status | Returnerer sekvenssteg, antall aktive alarmer, liste over aktive alarmkoder, total operasjonstimer og timer siden service | Sequence step: Not Running. #Active Alarms: 2. Active alarm codes: [202] [205]. Total operating hours: 420. Since service: 69. |
| ORTA {minutter} | Står for **O**ver**R**ide **T**emperature **A**larm. Den deaktiverer alarmaksjoner ved temperaturalarm for kjølevæske I gitt antall minutter. Følgende eksempel vil deaktivere alarmaksjonen I 10 minutter: ORTA 10 | Coolant alarm will not perform any alarmactions for 10.0 minutes. |
| ADDTELE {TLF} {Navn} {Nivå} | Legger til kontakt I telefonlogg. Telefonnummer må være 8 siffer, navn kan ikke ha mellomrom, nivå kan være opp til 255.<br>Eksempel: ADDTELE 90154532 Ola_Normann 3 | **Hvis TLF ikke allerede eksisterer I logg:** Added {TLF} to phonebook<br>**Hvis TLF allerede eksisterer I logg:** Updated information for {TLF}<br>**Hvis ingen ledig plass I logg:** New contact written over old contact: {old TLF} {old Name} |
| RESETTELE | Setter nivået til alle kontakter I telefonlogg til 0. | All phonebooklevels was set to 0. No contacts will receive alarmmessages via SMS. |
| **Feilmeldinger** | | |
| **Melding Inn** | **Årsak** | **Returnerer melding** |
| {Ugjenkjennelig kommando} | Bruker ga en ugjenkjennelig kommando | Failed to perform command. |
| ACK {feil kode} | Bruker prøver kvittere et ikke-eksisterende alarmobjekt (feil kode) | Failed to perform command. Could not find that alarmcode! |
| ADDTELE {feil TLF} | Bruker ga et telefonnummer som ikke er 8 siffer | Failed to perform command. Phonenumber should be 8 digits. Ex: 90154532. |
| {Hvilken som helst melding uten admin-prefix} | Bruker er ikke listet I telefonlisten (SMS-vindu på HMI) og har heller ikke brukt admin-prefix før kommando. | Failed to perform command. User not verified. |

Det kreves at senders telefonnummer er oppført i kontaktlogg. Dette kan gjøres via brukergrensesnitt eller via SMS-kommando ADDTELE. Man kan overstyre denne begrensningen ved å bruke administrator-prefix før kommando. Eksempel: "$\{Adminprefix\}\ ACK\ ALL$". Denne funksjonen er ment for Getek som en bakdør, og holdes derfor hemmelig.

# F. Vedlegg - Funksjonsblokker og funksjoner, dokumentasjon

Dokumentasjonen inneholder de mest relevante funksjonsblokker og funksjoner som er programmert av gruppen. Dokumentasjonen inneholder ikke funksjoner eller funksjonsblokker fra innebygde biblioteker, eller metoder og annen programkode programmert av gruppen.

# F.1 FC: FORMAT_STRING

**Grensesnittvariabler**

| Scope | Navn | Type | Initialverdi | Kommentar |
|---|---|---|---|---|
| Input | sMain_string | STRING | | Opprinnelig tekst |
| | sInput_string | STRING | | Tekst som skal fylles inn i opprinnelig tekst |
| | sFormat_string | STRING | | Tekst som skal erstattes av Input-tekst |
| Output | FORMAT_STRING | STRING | | Formatert tekst |

**Funksjon**

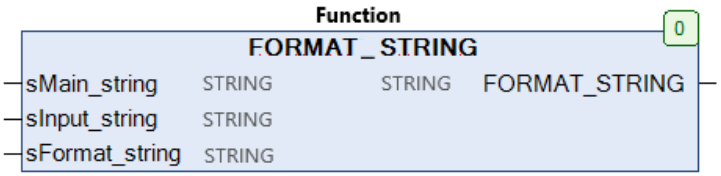
Formaterer String-variabler. Inspirert fra Matlab og Python String-formatering.
Hvis funksjonen ikke finner $sFormat_string$ i $sMain_string$ returneres $sMain_string$, uendret.
Eksempel 1:

$$\begin{aligned} sMain_string &:= ' \ Mitt\ navn\ er\ \%s' \\ sInput_string &:= 'Bjarne' \\ sFormat_string &:= ' \ \%s' \\ FORMAT_STRING &:= ' \ Mitt\ navn\ er\ Bjarne' \end{aligned}$$

Eksempel 2:

$$\begin{aligned} sMain_string &:= ' \ Startforsøk\ feilet < LATCH1 > ganger!' \\ sInput_string &:= '5' \\ sFormat_string &:= '< LATCH1 > ' \\ FORMAT_STRING &:= ' \ Startforsøk\ feilet\ 5\ ganger' \end{aligned}$$

**Grafisk illustrasjon**




**Programkode**

```
(*      Replaces %s with your desired text!
        More or less same as FORMAT in Python.       *)

FUNCTION FORMAT_STRING : STRING
VAR_INPUT
        sMain_string : STRING;
        sInput_string : STRING;        // String to replace sFormat_string with.
        sFormat_string : STRING;       // Ex: %s, <LATCH1> ...
END_VAR
VAR
        index : INT;
END_VAR
-----------------------------------------------------------------------------------------
index := FIND(sMain_string,sFormat_string);

IF index = 0 THEN
        FORMAT_STRING := sMain_string; // Could not be formated!
ELSE
        FORMAT_STRING := CONCAT(LEFT(sMain_string,index-1),
                         CONCAT(sInput_string, RIGHT(sMain_string,LEN(sMain_string)
                         -LEN(sFormat_string)-LEN(LEFT(sMain_string,index-1))))));
END_IF
```

## F.2 FC: TIME_in_ms_to_REAL_in_mins

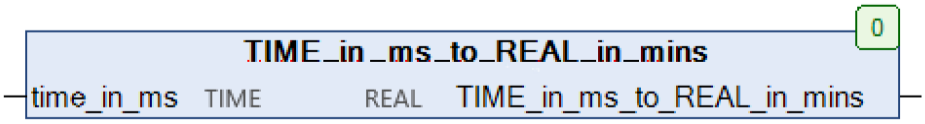
**Grensesnittvariabler**

| Scope | Navn | Type | Initialverdi | Kommentar |
|---|---|---|---|---|
| Input | time_in_ms | TIME | | Tid i millisekunder |
| Output | TIME_in_ms_to_REAL_in_mins | REAL | | Tid i minutter |

**Funksjon**

Konverterer tid (TIME) i millisekunder til tid (REAL) i minutter.

**Grafisk illustrasjon**




**Programkode**

```
FUNCTION TIME_in_ms_to_REAL_in_mins : REAL
VAR_INPUT
        time_in_ms : TIME;
END_VAR
-----------------------------------------------------------------------------------------
TIME_in_ms_to_REAL_in_mins := TIME_TO_REAL(time_in_ms)/60000;
```

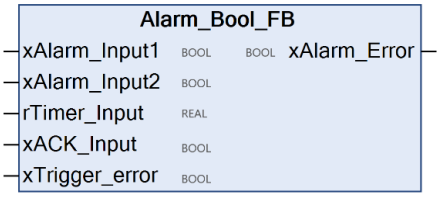
# F.3 FB: BOOL_ALARM

| Scope | Navn | Type | Initialverdi | Kommentar |
|---|---|---|---|---|
| Input | xAlarm_Input1 | BOOL | | Hoved errorbit |
| | xAlarm_Input2 | BOOL | | Set til TRUE hvis ikke i bruk. Alarm hvis både 1 og 2 er høy! |
| | rTimer_Input | REAL | 0 | Tid før alarm error går høy. |
| | xACK_input | BOOL | | Prøver å kvittere alarm når xACK er høy |
| | xTrigger_error | BOOL | | Manuelt trigger xAlarm_error |
| Output | xAlarm_error | BOOL | | |

**Funksjon**

Brukt for alarmdeteksjon av Boolske variabler, programmert i Ladderdiagram. Hvis både $xAlarm_Input1$ og $xAlarm_input2$ er høy, vil $xAlarm_error$ vil gå høy etter $rTimer_Input$ sekunder. Hvis $xTrigger_error$ går høy vil utgangen $xAlarm_error$ gå høy med en gang. Når $xACK_input$ er høy prøver man å kvittere alarmen. Set-reset blokka har set dominans, så $xACK_input$ har ***ikke*** prioritet over andre inputs.

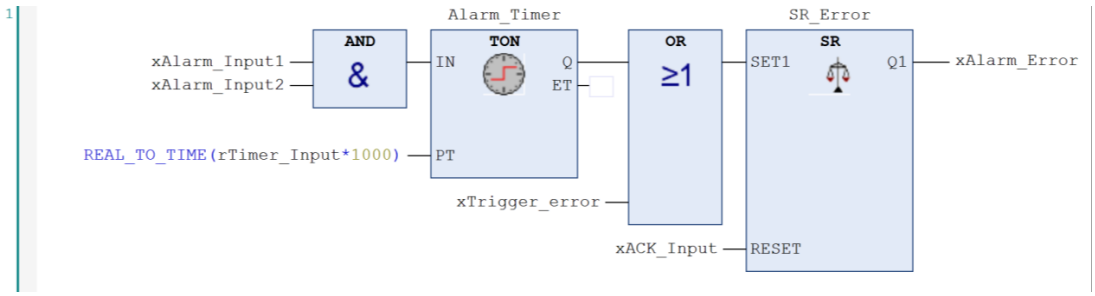
**Grafisk illustrasjon**




**Programkode**

```
FUNCTION_BLOCK BOOL_ALARM
VAR_INPUT
        xAlarm_Input1: BOOL;   // Main error bit.
        xAlarm_Input2: BOOL;   // Set true if not in use. Alarm if both input1 and 2 is
high

        rTimer_Input: REAL := 0; // Time of alarm before error goes high
        xACK_Input: BOOL;      // Set true, if the error should disapear when not "active"
        xTrigger_error: BOOL;
END_VAR
VAR_OUTPUT
        xAlarm_Error: BOOL;
END_VAR
VAR
        Alarm_Timer: TON;
        SR_Error : SR;
END_VAR
```



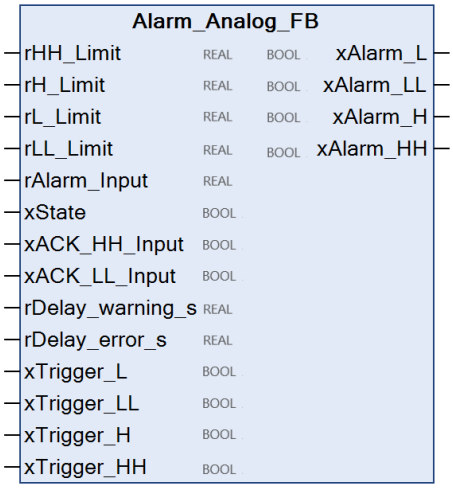
# F.4 FB: Alarm_Analog_FB

| Scope | Navn | Type | Initialverdi | Kommentar |
|---|---|---|---|---|
| Input | rHH_Limit | REAL | | Alarmgrenser |
| | rH_Limit | REAL | | |
| | rL_Limit | REAL | | |
| | rLL_Limit | REAL | | |
| | rAlarm_Input | REAL | | Prosessverdi. Verdi som skal monitores |
| | xState | BOOL | | Må være TRUE for at alarm skal triggeres |
| | xACK_LL_Input | BOOL | | Hvis TRUE: Alarm vil automatisk forsvinne når feilen forsvinner. |
| | xACK_HH_Input | BOOL | | |
| | rDelay_warning_s | REAL | | Antall sekunder alarmen må være utenfor grensen før alarmbits går høy. |
| | rDelay_error_s | REAL | | |
| | xTrigger_L | BOOL | | Brukt for manuell triggering av alarmer |
| | xTrigger_LL | BOOL | | |
| | xTrigger_H | BOOL | | |
| | xTrigger_HH | BOOL | | |
| Output | xAlarm_L | BOOL | | Alarm outputs |
| | xAlarm_LL | BOOL | | |
| | xAlarm_H | BOOL | | |
| | xAlarm_HH | BOOL | | |

**Funksjon**
Brukt for alarmdeteksjon av Analoge verdier. Det er deklarert 4 input for grenser, hvor HH(høy høy) og LL (lav lav) er de «kritiske» alarmene, og L og H er advarsel. Så er det en inngang for verdien/variablen som skal overvåkes. Deretter en inngang kallet «xState». Når denne er høy vil blokken være aktiv og alarmer vil potensielt kunne settes høye. Dette kan for eksempel være at man ikke vil trigge alarmer for frekvens når generatoren ikke kjører o.l.
Så er det to innganger for å kvittere LL og HH alarmene. Deretter er det to innganger av type «real» for forsinkelse innkobling av L/H og LL/HH alarmene. Og laget 4 innganger som styrer utgangene direkte.

Dette betyr at om rAlarm_input er lavere enn L_limit eller høyere enn H_limit og xState er sann i xDelay_warings_s sekunder, vil tilsvarende alarm gå høy.
Om derimot rAlarm_input er lavere enn LL_limit eller høyere enn HH_limit og xState er sann, i mer enn xDelay_error_s sekunder vil L/H alarmen gå lav og man vil kun få HH eller LL alarmen aktiv. Denne alarmen vil forbli høy helt til kriteriene for at alarmen skal gå høy ikke til stede, **og** at man har kvittert for alarmen.

**Grafisk illustrasjon** →



## Programkode

```
FUNCTION_BLOCK Alarm_Analog_FB
VAR_INPUT
        // Limits
        rHH_Limit: REAL;
        rH_Limit: REAL;
        rL_Limit: REAL;
        rLL_Limit: REAL;

        // Alarm inputs
        rAlarm_Input     : REAL;
        xState           : BOOL; // Needs to be on for alarm to trigger
        xACK_LL_Input    : BOOL; // Set true, if the error should disapear when not "active"
        xACK_HH_Input    : BOOL; // Set true, if the error should disapear when not "active"
        xDelay_warning_s : REAL; //in seconds. Value needs to be out of limit for this
        xDelay_error_s : REAL;   //duration. Use to adjust

        // Manual triggering from HMI
        xTrigger_L       : BOOL;
        xTrigger_LL      : BOOL;
        xTrigger_H       : BOOL;
        xTrigger_HH      : BOOL;
END_VAR
VAR_OUTPUT
        xAlarm_L         : BOOL;
        xAlarm_LL        : BOOL;
        xAlarm_H         : BOOL;
        xAlarm_HH        : BOOL;
END_VAR
VAR
        xManual_override: BOOL;
        TON_LL_HH        : TON;
        TON_L_H          : TON;
END VAR
```

```
// Acknowledge
IF xACK_LL_Input THEN
        xAlarm_LL := FALSE;
END_IF
IF xACK_HH_Input THEN
        xAlarm_HH := FALSE;
END_IF

// Prevents alarm to trigger before condition has been met for xDelay_ms milliseconds
TON_LL_HH(IN:=(rAlarm_Input < rLL_Limit) OR (rAlarm_Input > rHH_Limit),
                PT:=REAL_TO_TIME(xDelay_error_s*1000));
TON_L_H(IN:=(rAlarm_Input < rL_Limit) OR (rAlarm_Input > rH_Limit),
                PT:=REAL_TO_TIME(xDelay_warning_s*1000));

// Low alarm
IF (TON_LL_HH.Q AND xState AND (rAlarm_Input < rLL_Limit)) OR xTrigger_LL THEN
        xAlarm_LL := TRUE;
        xAlarm_L := FALSE;
ELSIF (TON_L_H.Q AND xState AND (rAlarm_Input < rL_Limit)) OR xTrigger_L THEN
        xAlarm_L := TRUE;
ELSE
        xAlarm_L := FALSE;
END_IF

// High alarm
IF(TON_LL_HH.Q AND xState AND (rAlarm_Input > rHH_Limit)) OR xTrigger_HH THEN
        xAlarm_HH := TRUE;
        xALarm_H := FALSE;
ELSIF (TON_L_H.Q AND xState AND (rAlarm_Input > rH_Limit)) OR xTrigger_H THEN
        xAlarm_H := TRUE;
ELSE
        xAlarm_H := FALSE;
END_IF

//Does not want alarm when this input is set to 0, or not in use.
IF rHH_Limit = 0 THEN
        xAlarm_HH :=FALSE;
END_IF
IF rH_Limit = 0 THEN
        xAlarm_H :=FALSE;
END IF
```

## F.5 FB: Check_comfault_459

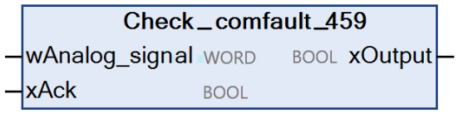
**Grensesnittvariabler**

| Scope | Navn | Type | Initialverdi | Kommentar |
|---|---|---|---|---|
| Input | wAnalog_signal | WORD | | Målesignal |
| | xAck | BOOL | | Kvitteringsinngang |
| Output | xOutput | BOOL | | Error-utgang |

**Funksjon**

Brukt til deteketering av kommunikasjosfeil. Blokken tar inn et ord på inngangen «wAnalog_signal», og leser av de to første bit-ene i ordet (LSB). Hvis begge bit-ene har verdi lik 1, vil det begynne en timer på 2 sekunder. Hvis begge bit-ene er lik 1 i disse 2 sekundene, vil utgangen «xOutput» bli skrevet høy. For å tilbakestille utgangen, leser blokken av en kvitteringsinngang «xAck». Hvis denne inngangen er satt høy, vil utgangen bli tilbakestilt.

**Grafisk illustrasjon**




**Programkode**

```
FUNCTION TIME_in_ms_to_REAL_in_mins : REAL
VAR_INPUT
        time_in_ms : TIME;
END_VAR
-----------------------------------------------------------------------------------------
TIME_in_ms_to_REAL_in_mins := TIME_TO_REAL(time_in_ms)/60000;
```

# F.6 FC: Scale_AI

**Grensesnittvariabler**

| Scope | Navn | Type | Initialverdi | Kommentar |
|---|---|---|---|---|
| Input | rMeasured_value | REAL | | Målt verdi |
| | rZero | REAL | | Nullpunkt, eller LRL (Lower Range Limit). |
| | rURL | REAL | | Øvre målegrense. URL (Upper Range Limit). |
| | rX1 | REAL | | Oppløsning, eller range. Kommer an på antall bit i rMeasured_value. |
| Output | Scale_AI | REAL | | Skalert verdi basert på rMeasured_value. |

**Funksjon**

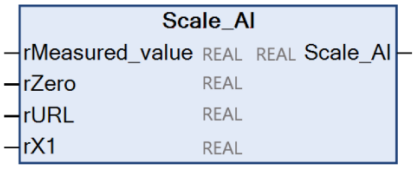
Funksjonen bruker formel:

$$ScaleAI = MeasuredValue \cdot \frac{(URL - Zero)}{x1} + Zero$$

Dette har sammenheng med formel:

$$y(x) = ax + b$$

**Grafisk illustrasjon**




**Programkode**

```
FUNCTION Scale_AI : REAL
VAR_INPUT
        rMeasured_value: REAL;
        rZero          : REAL;
        rURL           : REAL;
        rX1            : REAL;
END_VAR
-----------------------------------------------------------------------------------------
(*      Scales the measured value in a linear fashion.
        f(x)    = ax + b
        f(0)    = zero
        f(x1)   = span *)

Scale_AI := rMeasured_value*(rURL-rZero)/rX1 + rZero;
```

# F.7 FC: Calc_percent_of_total

**Grensesnittvariabler**

| Scope | Navn | Type | Initialverdi | Kommentar |
|---|---|---|---|---|
| Input | rCurrent_value | REAL | | Nåværende verdi |
| | rReference_value | REAL | | Referanseverdien |
| Output | Calc_percent_of_total | REAL | | Utregnet prosentandel |

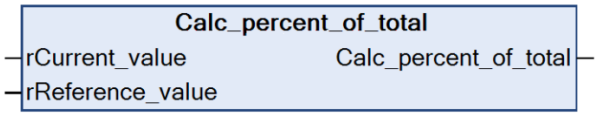
**Funksjon**

Funksjonsblokken regner ut en verdi sin prosentandel sammenliknet med en referanseverdi. Blokken tar utgangspunkt i formelen

$$Calc_percent_of_total = \frac{rCurrent_value}{rReference_value} * 100$$

For å unngå problemer med å dele på 0, er det i tillegg lagt til et unntak hvis rReference_value er lik 0.

**Grafisk illustrasjon**




**Programkode**

```
FUNCTION Calc_percent_of_total : REAL

VAR_INPUT
        rCurrent_value    : REAL;
        rReference_value : REAL;
END_VAR
-----------------------------------------------------------------------------------------
IF rReference_value = 0 THEN  // Prevents division by 0 error
        Calc_percent_of_total := 999;
ELSE
        Calc_percent_of_total := ABS(rCurrent_value/rReference_value *100);
END_IF
```

# G. Vedlegg - Alarmoversikt



## Alarmoversikt

- Alarmobjekt er definert I GVL_alarm.
- alarmgruppe og klasse er en del av Alarmmanager (fra bibiliotek).
- Kvitteringstype oppført I tabell er basert på funksjonsblokker for deteksjon og kvittering. Kvitteringstyper I Alarmmanager er alle av type REP.
- Alle alarmaksjoner har varsling over HMI og SMS (hvis aktivert).

*Alarmkode 210 kvitteres ved å resette operasjonstimer fra motorvindu på HMI. Skal bare gjøres etter utført vedlikehold.

| Alarm-kode | Alarmgruppe | Alarmobjekt | Klasse | Kvitterings-type | Manuell Overstyring | Alarmtekst (Norsk) | Bryter kjøreflagg | Handling / Alarmaksjon |
|---|---|---|---|---|---|---|---|---|
| 100 | Generator_Alarmgroup | Alarm_Freq_LL | Error_PRI_2 | ACK_REP | | [100] Generator frekvens er for lav. | X | Umiddelbart slå ut lastkontaktor → Nedkjøling av motor → Stopp |
| 101 | Generator_Alarmgroup | Alarm_Freq_L | Warning_REP | REP | | [101] Generator frekvens er lav | | Kun varsling |
| 102 | Generator_Alarmgroup | Alarm_Freq_H | Warning_REP | REP | | [102] Generator frekvens er høy | | Kun varsling |
| 103 | Generator_Alarmgroup | Alarm_Freq_HH | Error_PRI_2 | ACK_REP | | [103] Generator frekvens er for høy | X | Umiddelbart slå ut lastkontaktor → Nedkjøling av motor → Stopp |
| 104 | Generator_Alarmgroup | Alarm_Voltage_LL | Error_PRI_2 | ACK_REP | | [104] Generator spenning er for lav | X | Umiddelbart slå ut lastkontaktor → Nedkjøling av motor → Stopp |
| 105 | Generator_Alarmgroup | Alarm_Voltage_L | Warning_REP | REP | | [105] Generator spenning er lav | | Kun varsling |
| 106 | Generator_Alarmgroup | Alarm_Voltage_H | Warning_REP | REP | | [106] Generator spenning er høy | | Kun varsling |
| 107 | Generator_Alarmgroup | Alarm_Voltage_HH | Error_PRI_2 | ACK_REP | | [107] Generator spenning er for høy | X | Umiddelbart slå ut lastkontaktor → Nedkjøling av motor → Stopp |
| 200 | Motor_Alarmgroup | Alarm_Cooling_Temp_High | Error_PRI_1 | ACK_REP | X | [200] Temperatur kjølevæske er for høy. | X | Umiddelbart slå ut lastkontaktor → Nedkjøling av motor → Stopp |
| 201 | Motor_Alarmgroup | Alarm_Diesel_L | Warning_REP | REP | | [201] Drivstoffnivå er lav. | | Kun varsling |
| 202 | Motor_Alarmgroup | Alarm_Diesel_LL | Error_PRI_3 | ACK_REP | | [202] Drivstoffnivå er svært lavt. | | Kun varsling |
| 203 | Motor_Alarmgroup | Alarm_Oil_Pressure | Error_PRI_1 | ACK_REP | | [203] Ikke oljetrykk, aggregat stoppet | X | Umiddelbart slå ut lastkontaktor og stopp motor |
| 204 | Motor_Alarmgroup | Alarm_Start_Battery_L | Warning_REP | REP | | [204] Startbatteri spenning er lav | | Kun varsling |
| 205 | Motor_Alarmgroup | Alarm_Start_Battery_LL | Error_PRI_1 | ACK_REP | | [205] Startbatterispenning er for lav | | Kun varsling |
| 206 | Motor_Alarmgroup | Alarm_Start_Fail | Error_PRI_1 | ACK_REP | | [206] Start av motor feilet **{Antall *startforsøk* }** ganger | X | Stopp motor |
| 207 | Motor_Alarmgroup | Alarm_Emergency_Stop | Error_PRI_1 | ACK_REP | | [207] Nødstopp aktivert. | X | Umiddelbart slå ut lastkontaktor og stopp motor |
| 208 | Motor_Alarmgroup | Alarm_Dynamo | Error_PRI_2 | ACK_REP | | [208] Dynamo lader ikke. | | Kun varsling |
| 209 | Motor_Alarmgroup | Alarm_Stop_Fail | Error_PRI_1 | ACK_REP | | [209] Motor stopper ikke. Oljetrykk høyt | X | Legge inn og ut driftsmagnet 3 ganger |
| 210 | Motor_Alarmgroup | Alarm_Opperation_hours | Warning_REP | REP* | | [210] Tid for vedlikehold. **{Op.timer siden service}** timer siden service. | | Kun varsling |
| 211 | Motor_Alarmgroup | Alarm_Main_Battery_L | Warning_REP | REP | | [211] Hovedbatterispenning er lav | | Kun varsling |
| 212 | Motor_Alarmgroup | Alarm_Main_Battery_LL | Error_PRI_1 | ACK_REP | | [212] Hovedbatterispenning er for lav | | Kun varsling |
| 301 | PLC_COM_Alarmgroup | Alarm_com_energy | Warning_REP | REP | | [301] Feil i energimåler. | | Kun varsling |
| 302 | PLC_COM_Alarmgroup | Alarm_power_loss | Warning_REP | REP | | [302] Strømbrudd oppsto | | Kun varsling |
| 303 | PLC_COM_Alarmgroup | Alarm_com_room_temp | Warning_REP | REP | | [303] Komm.feil med romtemperaturføler | | Kun varsling |
| 304 | PLC_COM_Alarmgroup | Alarm_com_main_bat | Warning_REP | REP | | [304] Hovedbatteri spenning ute av måleområde | | Kun varsling |
| 305 | PLC_COM_Alarmgroup | Alarm_com_start_bat | Warning_REP | REP | | [305] Startbatteri spenning ute av måleområde | | Kun varsling |
| 306 | PLC_COM_Alarmgroup | Alarm_com_diesel_lvl | Warning_REP | REP | | [306] Drivstoffnivå måling ute av måleområde | | Kun varsling |
| 307 | PLC_COM_Alarmgroup | xGSMerror | Warning_REP | REP | | [307] Feil med SMS-klient. | | Kun varsling |
| 308 | PLC_COM_Alarmgroup | Alarm_com_coolant | Warning_REP | REP | | [308] Kjølevæskesensor måling ute av | | Kun varsling |
| 309 | PLC_COM_Alarmgroup | Alarm_com_oilpressure | Warning_REP | REP | | [309] Oljetrykkssensor måling ute av måleområde | | Kun varsling |

# H. Vedlegg - Oversikt over HMI

| | | | HMI-Vindu | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | * Kan aktiveres / dekativeres I [Instillinger]<br>** Kan aktiveres / deaktiveres I [Avanserte Innstillinger]<br>Grønn farge indikerer like skrive og leserettigheter. Oransje indikerer en forskjell mellom logget ut og Admin. | | | | | | | | | | |
| | Logget ut | Admin | Overview | Generator | Motor | Innstillinger | Alarm | Avanserte Innstillinger | Motor Innstillinger | Generator Innstillinger | SMS | SMS Innstillinger | Alarm Test |
| Toppmeny | R | R | X | X | X | X | X | X | X | X | X | X | X |
| [Banner] Startforsøk og steg | R | R | X | X | X | | | | | | | | |
| Alarmbanner | R | R | X | X | X | X | X | X | X | X | X | X | X |
| Startbatteri spenning | R | R | X | | X | | | | | | | | |
| Hovedbatteri spenning | R | R | X | | X | | | | | | | | |
| Drivstoff (0...100%) | R | R | X | | X | | | | | | | | |
| **Temperatur (rom) | R | R | X | | | | | | | | | | |
| [Høy/Lav] Dynamo | R | R | X | | X | | | | | | | | |
| [Høy/Lav] Oljetrykk | R | R | X | | X | | | | | | | | |
| [Høy/Lav] Motortemp | R | R | X | | X | | | | | | | | |
| **[AI] Kjølevæsketempteratur | R | R | X | | X | | | | | | | | |
| Frekvens | R | R | X | X | | | | | | | | | |
| Spenning (RMS) | R | R | X | X | | | | | | | | | |
| **Strøm | R | R | X | X | | | | | | | | | |
| **Aktiv Effekt | R | R | | X | | | | | | | | | |
| **Reaktiv Effekt | R | R | | X | | | | | | | | | |
| **Tilsynelatende Effekt | R | R | | X | | | | | | | | | |
| **Aktiv Energi | R | R | | X | | | | | | | | | |
| **cos(ϕ) | R | R | X | X | | | | | | | | | |
| Lastkontaktor (on/off) symbol | R | R | X | | | | | | | | | | |
| Driftmagnet (on/off) symbol | R | R | X | | | | | | | | | | |
| Knapp for overstyring temperatur | R | RW | | | X | | | | | | | | |
| Driftstimer (totalt) | R | R | | | X | | | | | | | | |
| Timer siden service | R | RW | | | X | | | | | | | | |
| Mørk Modus | RW | RW | | | | X | | | | | | | |
| Motta kommandoer over SMS | RW | RW | | | | X | | | | | | | |
| Send varslinger over SMS | RW | RW | | | | X | | | | | | | |
| Omgivelsestemperatur | | RW | | | | | | X | | | | | |
| Drivstoffnivåsensor: Digital eller Analog | | RW | | | | | | X | | | | | |
| Oljetrykkssensor: Digital eller Analog | | RW | | | | | | X | | | | | |
| Kjølevæskesensor: Digital eller Analog | | RW | | | | | | X | | | | | |
| Strøm måling tilgjengelig | | RW | | | | | | X | | | | | |
| Alarm ved kom.feil Hovedbatteri (V) | | RW | | | | | | X | | | | | |
| Alarm ved kom. Feil Startbatteri (V) | | RW | | | | | | X | | | | | |
| Valg av Analoge Inputs for kort 750-459 | | RW | | | | | | X | | | | | |
| Skalering strømmåler (forholdstall) | | RW | | | | | | X | | | | | |

| | | | HMI-Vindu | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | * Kan aktiveres / dekativeres I [Instillinger]<br>** Kan aktiveres / deaktiveres I [Avanserte Innstillinger]<br>Grønn farge indikerer like skrive og leserettigheter. Oransje indikerer en forskjell mellom logget ut og Admin. | | | | | | | | | | |
| | **Logget ut** | **Admin** | **Overview** | **Generator** | **Motor** | **Innstillinger** | **Alarm** | **Avanserte Innstillinger** | **Motor Innstillinger** | **Generator Innstillinger** | **SMS** | **SMS Innstillinger** | **Alarm Test** |
| Antall Startforsøk | R | RW | | | | | | | X | | | | |
| Makstid starter | R | RW | | | | | | | X | | | | |
| Nåværende Glødetid | R | **RW | | | | | | | X | | | | |
| Startfrekvens motor | R | RW | | | | | | | X | | | | |
| Tidsforsinkelse Oljesjekk | R | RW | | | | | | | X | | | | |
| Tidsforsinkelse Lastkontaktor | R | RW | | | | | | | X | | | | |
| Nedkjøling Motor (tomgang) | R | RW | | | | | | | X | | | | |
| **Linearisering Glødetid | | RW | | | | | | | X | | | | |
| Serviceintervall | R | RW | | | | | | | X | | | | |
| Alarmgrense HH, H, L, LL Frekvens | R | RW | | | | | | | | X | | | |
| Alarmgrense HH, H, L, LL Spenning | R | RW | | | | | | | | X | | | |
| Tidsforsinkelse alarm frekvens | R | RW | | | | | | | | X | | | |
| Tidsforsinkelse alarm spenning | R | RW | | | | | | | | X | | | |
| Tidsforsinkelse advarsel frekvens | R | RW | | | | | | | | X | | | |
| Tidsforsinkelse advarsel spenning | R | RW | | | | | | | | X | | | |
| Alarmtabell med tilhørende knapper | RW | RW | | | | | X | | | | | | |
| [SMS] Tabell over personer som skal ha varsling | R | R | | | | | | | | | X | | |
| [SMS] Knapp for å endre personer I logg | RW | RW | | | | | | | | | X | | |
| [SMS] varslingsnivå til følgende alarmkoder:<br>100, 103, 104, 107, 200, 202, 203, 205, 206, 207, 208, 209, 300, 301, 302, 303, 304, 305, 306 | R | RW | | | | | | | | | | X | |
| Toggle Generator Frekvens LL, L, H, HH | RW | RW | | | | | | | | | | | X |
| Toggle Generator Spenning LL, L, H, HH | RW | RW | | | | | | | | | | | X |
| Toggle Kjølevæske Advarsel, Feil | RW | RW | | | | | | | | | | | X |
| Toggle Drivstoffnivå Advarsel, Feil | RW | RW | | | | | | | | | | | X |
| Toggle Oljetrykk Advarsel, Feil | RW | RW | | | | | | | | | | | X |
| Toggle Start Feilet | RW | RW | | | | | | | | | | | X |
| Toggle Stop Feilet | RW | RW | | | | | | | | | | | X |
| Toggle Nødstopp | RW | RW | | | | | | | | | | | X |
| Toggle Dynamo Feil | RW | RW | | | | | | | | | | | X |
| Toggle Startbatteri Feil | RW | RW | | | | | | | | | | | X |
| Toggle Startbatteri Advarsel | RW | RW | | | | | | | | | | | X |

# I. Vedlegg - Timeoversikt, Gant- og S-diagram

## I.1 Vedlegg - Regnearkinnstillinger

**Settings** | Gannt

**Project Details**

| | |
|---|---|
| Project Title: | Bacheloroppgave, Autopanel |
| Project Manager: | ... |
| Project Priority: | High |

**Project Details**

| Role | Code | # |
|---|---|---|
| Alle | A1 | 1 |
| Deltaker 1 | D1 | 1 |
| Deltaker 2 | D2 | 1 |
| Deltaker 3 | D3 | 1 |
| Deltaker 4 | D4 | 1 |

**Project Details**

| Name | Role | Code |
|---|---|---|
| Alle | Alle | A1 |
| A. Grasbakken | Deltaker 2 | D2 |
| M. Grøterud | Deltaker 3 | D3 |
| K. Halse | Deltaker 4 | D4 |
| H. Bonde | Deltaker 1 | D1 |

**Holidays**

| Date | Comment |
|---|---|
| 01-Jan-22 | Nyttårsdag |
| 14-Apr-22 | Skjærtorsdag |
| 15-May-22 | Langfredag |
| 17-Apr-22 | 1. Påskedag |
| 18-Apr-22 | 2. Påskedag |
| 01-May-22 | Arbeidernes dag |
| 17-May-22 | Grunnlovsdagen |
| 26-May-22 | Himmelspretten |
| 05-Jun-22 | 1. Pinsedag |
| 06-Jun-22 | 2. Pinsedag |
| 25-Dec-22 | 1. Juledag |
| 26-Dec-22 | 2. Juledag |
| | |
| 23-Mar-22 | Eksamensdato |

# I.2 Vedlegg - Gantdiagram, ordinær vs. ny plan

**Bacheloroppgave, Autopanel: Ganntdiagram**

Project Timespan: ############## - 10-Jun-22 (110 Days) Project Manager: ... Settings

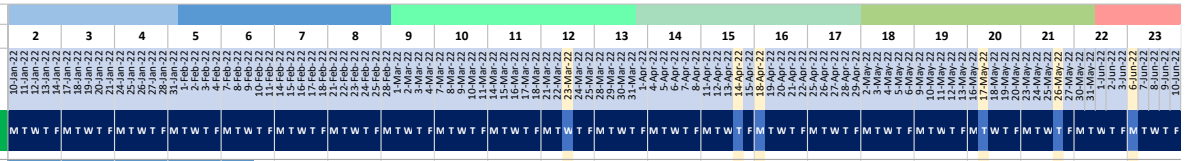
*Input Column*
*Calculated Column*
*Snapshot Column*

## Oridnær plan vs ny plan

Color By: Project Structure | Display: Plan vs Base | Show Progress: No | Highlight: Off

Actual Plan — Updated on: 31-Jan-22

| ids | Type | Item Details: Description | Role | Team Member | Work days | Start | End | % |
|---|---|---|---|---|---|---|---|---|
| | S | **Forprosjekt** | | | 23 | **10-Jan-22** | **9-Feb-22** | 100% |
| 0 | T | Planlegging og arbeid på forprosjektrapport | A1 | Alle | 23 | 10-Jan-22 | 9-Feb-22 | 100% |
| | S | **Hovedprosjekt** | | | 72 | **10-Feb-22** | **20-May-22** | 100% |
| 1 | T | Gjennomgang av anlegg og dokumentasjon | D2 | A. Grasbakken | 5 | 10-Feb-22 | 16-Feb-22 | 100% |
| 2 | T | Programmere PLS | D4 | K. Halse | 32 | 14-Feb-22 | 29-Mar-22 | 100% |
| 3 | T | HMI | D1 | H. Bonde | 29 | 17-Feb-22 | 29-Mar-22 | 100% |
| | M | Software (første utkast) klar for testing | | | | 29-Mar-22 | 29-Mar-22 | 100% |
| 4 | T | Testing og nødvendig utbedring | D2 | A. Grasbakken | 7 | 30-Mar-22 | 29-Apr-22 | 100% |
| | M | Anlegg montert og ferdig testet | | | | 29-Apr-22 | 29-Apr-22 | 100% |
| 5 | T | Videre utvidelser | D3 | M. Grøterud | 16 | 8-Apr-22 | 29-Apr-22 | 100% |
| 6 | T | Dokumentasjon anlegg | D4 | K. Halse | 12 | 14-Apr-22 | 29-Apr-22 | 100% |
| | M | Dokumentasjon av anlegg fullført | | | | 29-Apr-22 | 29-Apr-22 | 100% |
| 7 | T | Hovedrapport | D3 | M. Grøterud | 15 | 2-May-22 | 20-May-22 | 100% |
| 8 | T | Administrativt | D1 | H. Bonde | 72 | 10-Feb-22 | 20-May-22 | 100% |
| | M | Innlevering av hovedprosjektrapport | | | | 20-May-22 | 20-May-22 | 100% |
| | T | Presentasjon Bachelor | A1 | Alle | | 9-May-22 | 10-Jun-22 | 100% |
| 9 | S | **Toukersmøter** | | | 71 | **12-Jan-22** | **20-Apr-22** | 100% |
| 10 | T | Møte 1 (Oppstartsmøte) | A1 | Alle | 1 | 12-Jan-22 | 12-Jan-22 | 100% |
| 11 | T | Møte 2 | A1 | Alle | 1 | 19-Jan-22 | 19-Jan-22 | 100% |
| 12 | T | Møte 3 | A1 | Alle | | 9-Feb-22 | 9-Feb-22 | 100% |
| 13 | T | Møte 4 | A1 | Alle | | 2-Mar-22 | 2-Mar-22 | 100% |
| 14 | T | Møte 5 | A1 | Alle | | 30-Mar-22 | 30-Mar-22 | 100% |
| 15 | T | Møte 6 | A1 | Alle | | 20-Apr-22 | 20-Apr-22 | 200% |



# I.3 Vedlegg - Gantdiagram, ny plan

## Bacheloroppgave, Autopanel: Ganntdiagram

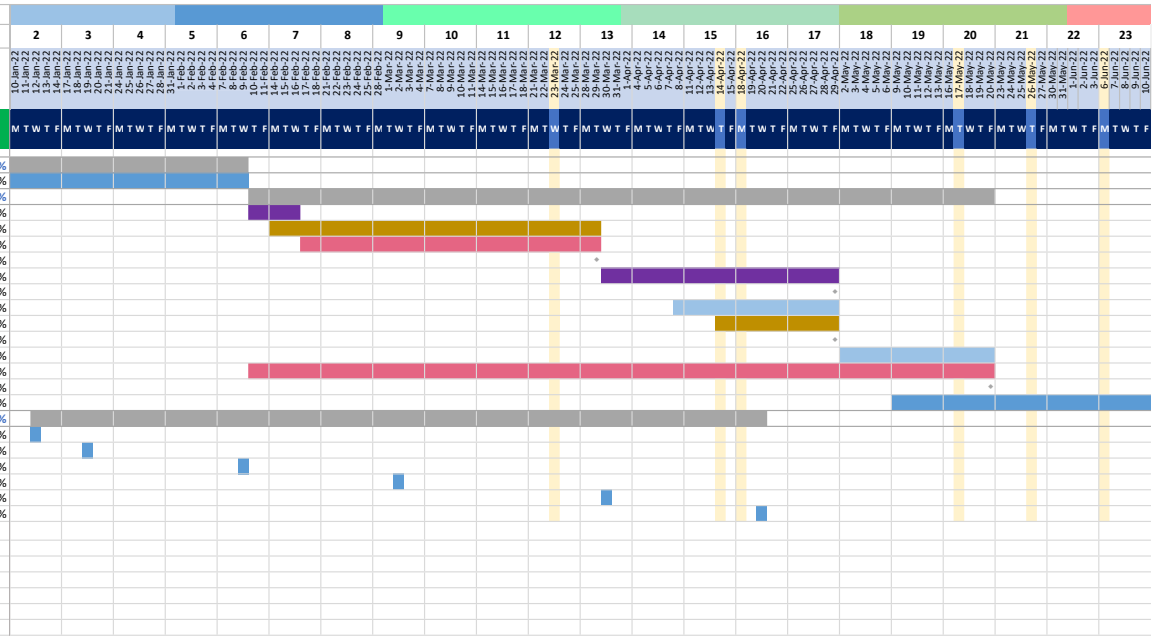
Project Timespan: ############# - 10-Jun-22 (110 Days) Project Manager: ... Settings

*Input Column*
*Calculated Column*
*Snapshot Column*

| Color By | Display | Show Progress | Highlight |
|---|---|---|---|
| Team Roles | Plan | No | Off |

Actual Plan
Updated on:
31-Jan-22

| ids | Item Details | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| id | Type | Description | Role | Team Member | Work days | Start | End | % |
| | S | **Forprosjekt** | | | 23 | **10-Jan-22** | **9-Feb-22** | 100% |
| 0 | T | Planlegging og arbeid på forprosjektrapport | A1 | Alle | 23 | 10-Jan-22 | 9-Feb-22 | 100% |
| | S | **Hovedprosjekt** | | | 72 | **10-Feb-22** | **20-May-22** | 100% |
| 1 | T | Gjennomgang av anlegg og dokumentasjon | D2 | A. Grasbakken | 5 | 10-Feb-22 | 16-Feb-22 | 100% |
| 2 | T | Programmere PLS | D4 | K. Halse | 32 | 14-Feb-22 | 29-Mar-22 | 100% |
| 3 | T | HMI | D1 | H. Bonde | 29 | 17-Feb-22 | 29-Mar-22 | 100% |
| | M | Software (første utkast) klar for testing | | | | 29-Mar-22 | 29-Mar-22 | 100% |
| 4 | T | Testing og nødvendig utbedring | D2 | A. Grasbakken | 7 | 30-Mar-22 | 29-Apr-22 | 100% |
| | M | Anlegg montert og ferdig testet | | | | 29-Apr-22 | 29-Apr-22 | 100% |
| 5 | T | Videre utvidelser | D3 | M. Grøterud | 16 | 8-Apr-22 | 29-Apr-22 | 100% |
| 6 | T | Dokumentasjon anlegg | D4 | K. Halse | 12 | 14-Apr-22 | 29-Apr-22 | 100% |
| | M | Dokumentasjon av anlegg fullført | | | | 29-Apr-22 | 29-Apr-22 | 100% |
| 7 | T | Hovedrapport | D3 | M. Grøterud | 15 | 2-May-22 | 20-May-22 | 100% |
| 8 | T | Administrativt | D1 | H. Bonde | 72 | 10-Feb-22 | 20-May-22 | 100% |
| | M | Innlevering av hovedprosjektrapport | | | | 20-May-22 | 20-May-22 | 100% |
| | T | Presentasjon Bachelor | A1 | Alle | | 9-May-22 | 10-Jun-22 | 100% |
| 9 | S | **Toukersmøter** | | | 71 | **12-Jan-22** | **20-Apr-22** | 100% |
| 10 | T | Møte 1 (Oppstartsmøte) | A1 | Alle | 1 | 12-Jan-22 | 12-Jan-22 | 100% |
| 11 | T | Møte 2 | A1 | Alle | 1 | 19-Jan-22 | 19-Jan-22 | 100% |
| 12 | T | Møte 3 | A1 | Alle | | 9-Feb-22 | 9-Feb-22 | 100% |
| 13 | T | Møte 4 | A1 | Alle | | 2-Mar-22 | 2-Mar-22 | 100% |
| 14 | T | Møte 5 | A1 | Alle | | 30-Mar-22 | 30-Mar-22 | 100% |
| 15 | T | Møte 6 | A1 | Alle | | 20-Apr-22 | 20-Apr-22 | 200% |

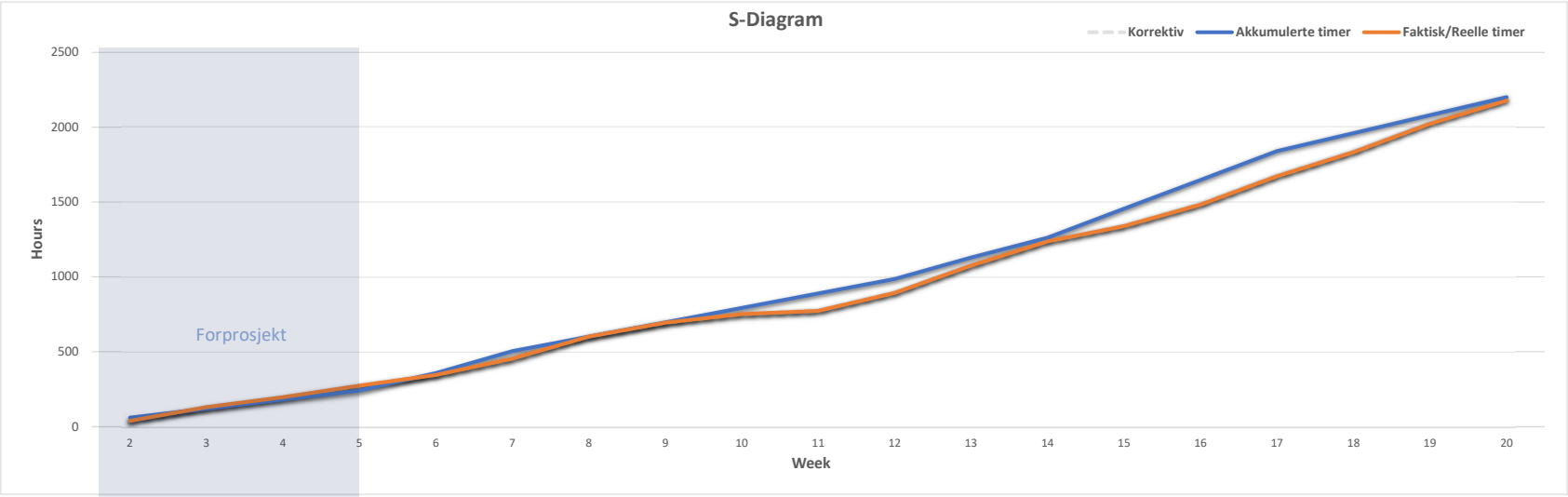


# I.4 Vedlegg - S-diagram

## Bacheloroppgave, Autopanel: S-Diagram

**Planlagt timer brukt på aktivieter**

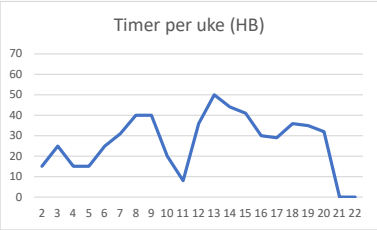
| id | Type | Beskrivelse | Start (Uke) | Slutt (Uke) | Varighet (Uker) | Timer (Planlagt) | Timer/ pakke | Week 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S | Forprosjekt | 2 | 6 | 5 | | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | T | Planlegging og arbeid på forprosjektrapport | 2 | 6 | 5 | 300 | 60.0 | 60.0 | 60.0 | 60.0 | 60.0 | 60.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | S | Hovedprosjekt | 6 | 20 | 15 | | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 1 | T | Gjennomgang av anlegg og dokumentasjon | 6 | 7 | 2 | 100 | 50.0 | 0.0 | 0.0 | 0.0 | 0.0 | 50.0 | 50.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2 | T | Programmere PLS | 7 | 13 | 7 | 330 | 47.1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 47.1 | 47.1 | 47.1 | 47.1 | 47.1 | 47.1 | 47.1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 3 | T | HMI | 7 | 13 | 7 | 300 | 42.9 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 42.9 | 42.9 | 42.9 | 42.9 | 42.9 | 42.9 | 42.9 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 0 | M | Software (første utkast) klar for testing | 13 | 13 | 1 | | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 4 | T | Testing og nødvendig utbedring | 13 | 17 | 5 | 230 | 46.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 46.0 | 46.0 | 46.0 | 46.0 | 46.0 | 0.0 | 0.0 | 0.0 |
| 0 | M | Anlegg montert og ferdig testet | 17 | 17 | 1 | | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 5 | T | Videre utvidelser | 14 | 17 | 4 | 320 | 80.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 80.0 | 80.0 | 80.0 | 80.0 | 0.0 | 0.0 | 0.0 |
| 6 | T | Dokumentasjon anlegg | 15 | 17 | 3 | 180 | 60.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 60.0 | 60.0 | 60.0 | 0.0 | 0.0 | 0.0 |
| 0 | M | Dokumentasjon av anlegg fullført | 17 | 17 | 1 | | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 7 | T | Hovedrapport | 18 | 20 | 3 | 340 | 113.3 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 113.3 | 113.3 | 113.3 |
| 8 | T | Administrativt | 6 | 20 | 15 | 100 | 6.7 | 0.0 | 0.0 | 0.0 | 0.0 | 6.7 | 6.7 | 6.7 | 6.7 | 6.7 | 6.7 | 6.7 | 6.7 | 6.7 | 6.7 | 6.7 | 6.7 | 6.7 | 6.7 | 6.7 |
| 0 | M | Innlevering av hovedprosjektrapport | 20 | 20 | 1 | | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| | | | | | | **Sum** | | 60 | 60 | 60 | 60 | 117 | 147 | 97 | 97 | 97 | 97 | 97 | 143 | 133 | 193 | 193 | 193 | 120 | 120 | 120 |
| | | | | | | **Akkumulerte timer** | | 60 | 120 | 180 | 240 | 356.3 | 503.3 | 600 | 696.7 | 793.3 | 890 | 986.7 | 1129 | 1262 | 1455 | 1647 | 1840 | 1960 | 2080 | 2200 |

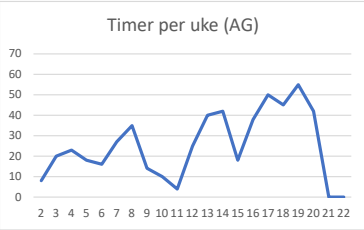


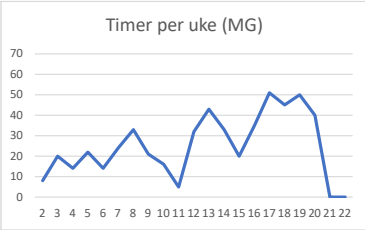
# I.5 Vedlegg - Timeoversikt

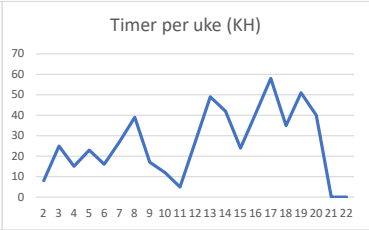
## Bacheloroppgave, Autopanel: Timeoversikt

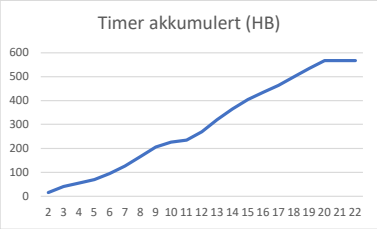
| Antall timer | | |

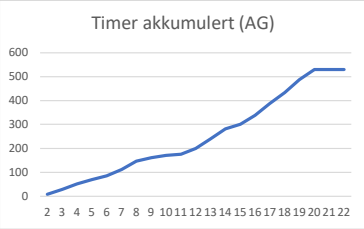


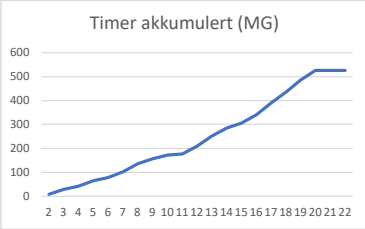



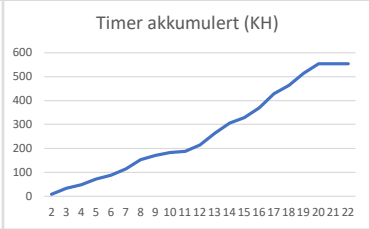



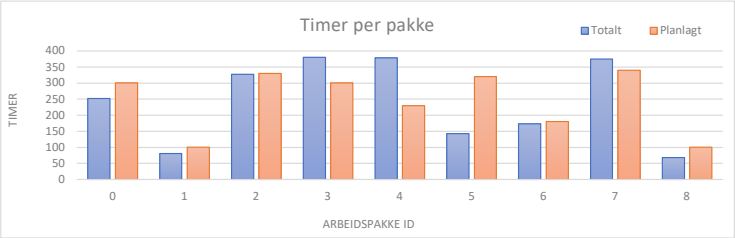




**Akkumulerte Faktiske/Reelle timer brukt per pakke**

| Package id | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Håvard Bonde** | 70 | 7 | 40 | 158 | 71 | 121 | 58 | 26 | 16 | 0 |
| **Amund Grasbakken** | 62 | 26 | 87 | 76 | 108 | 22 | 44 | 92 | 13 | 0 |
| **Magnus Grøterud** | 57 | 15 | 84 | 92 | 90 | 0 | 30 | 134 | 24 | 0 |
| **Kjetil Halse** | 63 | 33 | 116 | 54 | 109 | 0 | 41 | 123 | 15 | 0 |
| **Totalt** | 252 | 81 | 327 | 380 | 378 | 143 | 173 | 375 | 68 | 0 |
| **Planlagt** | 300 | 100 | 330 | 300 | 230 | 320 | 180 | 340 | 100 | 0 |
| **Differanse** | -48 | -19 | -3 | 80 | 148 | -177 | -7 | 35 | -32 | 0 |

| Sum |
|---|
| 2177 Hours |
| 2200 Hours |
| -23 Hours |



# J. Vedlegg - Poster

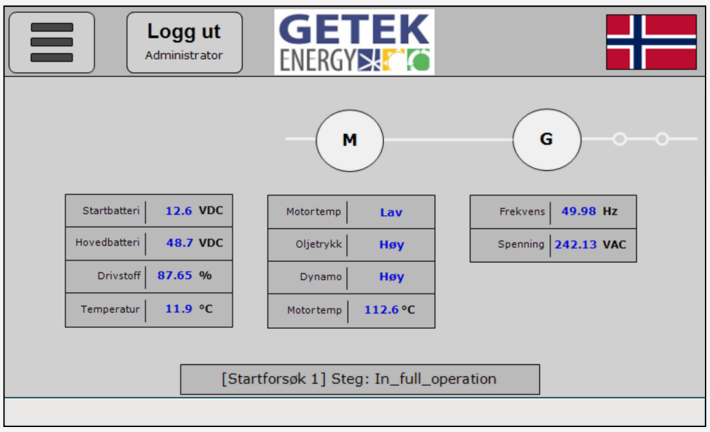
# PLS som autopanel for dieselaggregat i autonomt strømforsyningsanlegg

## Problemstilling

Oppdragsgiver, Getek AS, ønsker å erstatte og videreutvikle sitt autonome styresystem på sine APS-anlegg. Til dette skal det brukes en modulær PLS og annet utsyr fra Wago Norge. Oppdragsgiver ønsker større fleksibilitet og muligheten til å koble sammen diverse utstyr inne i APS-anlegget, noe det gamle styresystemet ikke kunne tilby.

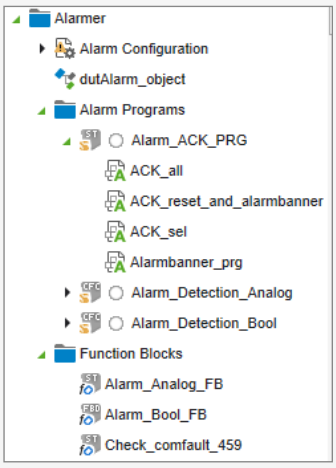
## Brukergrensesnitt

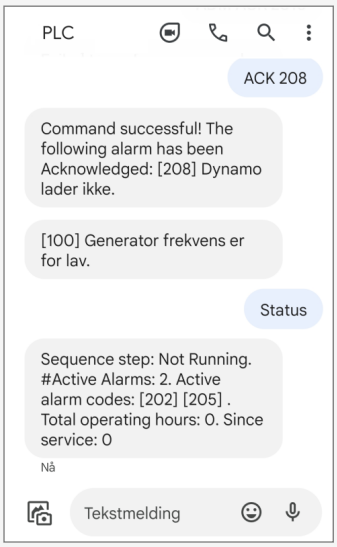
Simpel og oversiktlig brukergrensesnitt med inspirasjon fra High-Performance-HMI, med store knapper ment for en 7 tommer Touch-skjerm.




## Alarmsystem

Skreddersydd alarmsystem bygget rundt den innebygde Alarm-Manager i e!Cockpit.




## SMS-system

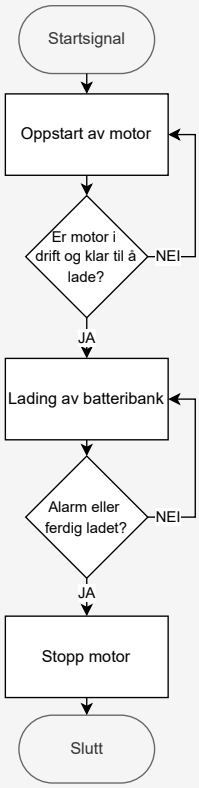
Brukervennlig SMS-system med flere muligheter for alarmvarsling og sending av kommandoer inn til PLS.

## Aggregatsekvens

Oversiktlig aggregatsekvens programmert i SFC ved hjelp av Wagos e!Cockpit.

Denne sekvensen styrer start og stopp av aggregat.

Her er sekvens visualisert svært forenklet i et flytskjema.



## Fysisk testet

Løsningen er blitt fysisk testet på et av Geteks APS-anlegg.

## Konklusjon

Det er blitt utviklet et styresystem med brukergrensesnitt for et dieselaggregat ved hjelp av en PLS. Løsningen har svært mange valgmuligheter i form av innstillinger. Det ble lagt stort fokus på driftssikkerhet og pålitelighet. Det konkluderes med at styresystemet egner seg godt til bruk på en autonom strømforsyningsenhet.


Håvard Bonde
Amund Grasbakken
Magnus Grøterud
Kjetil Halse


WAGO NTNU GETEK ENERGY

NTNU

Kunnskap for en bedre verden