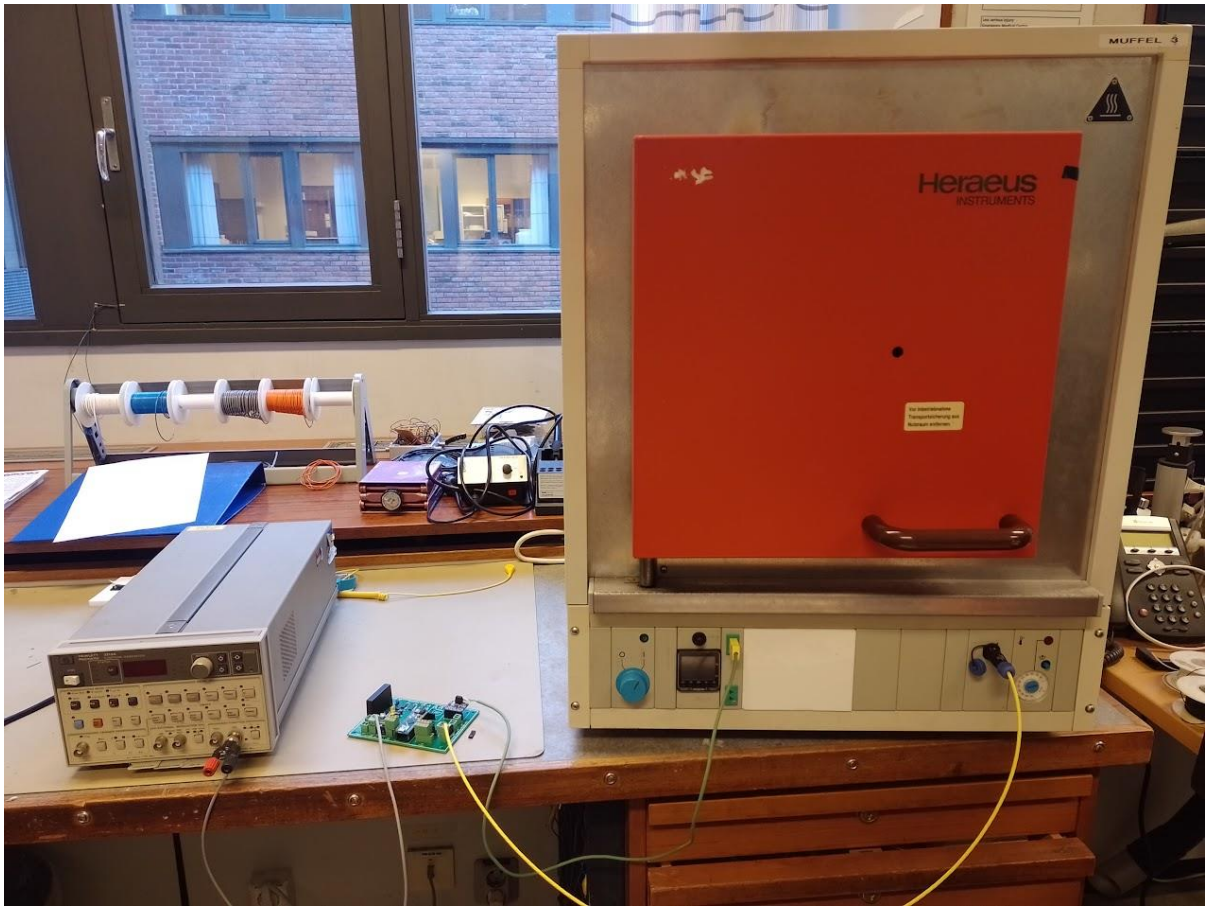
Fagprøve Cyrus



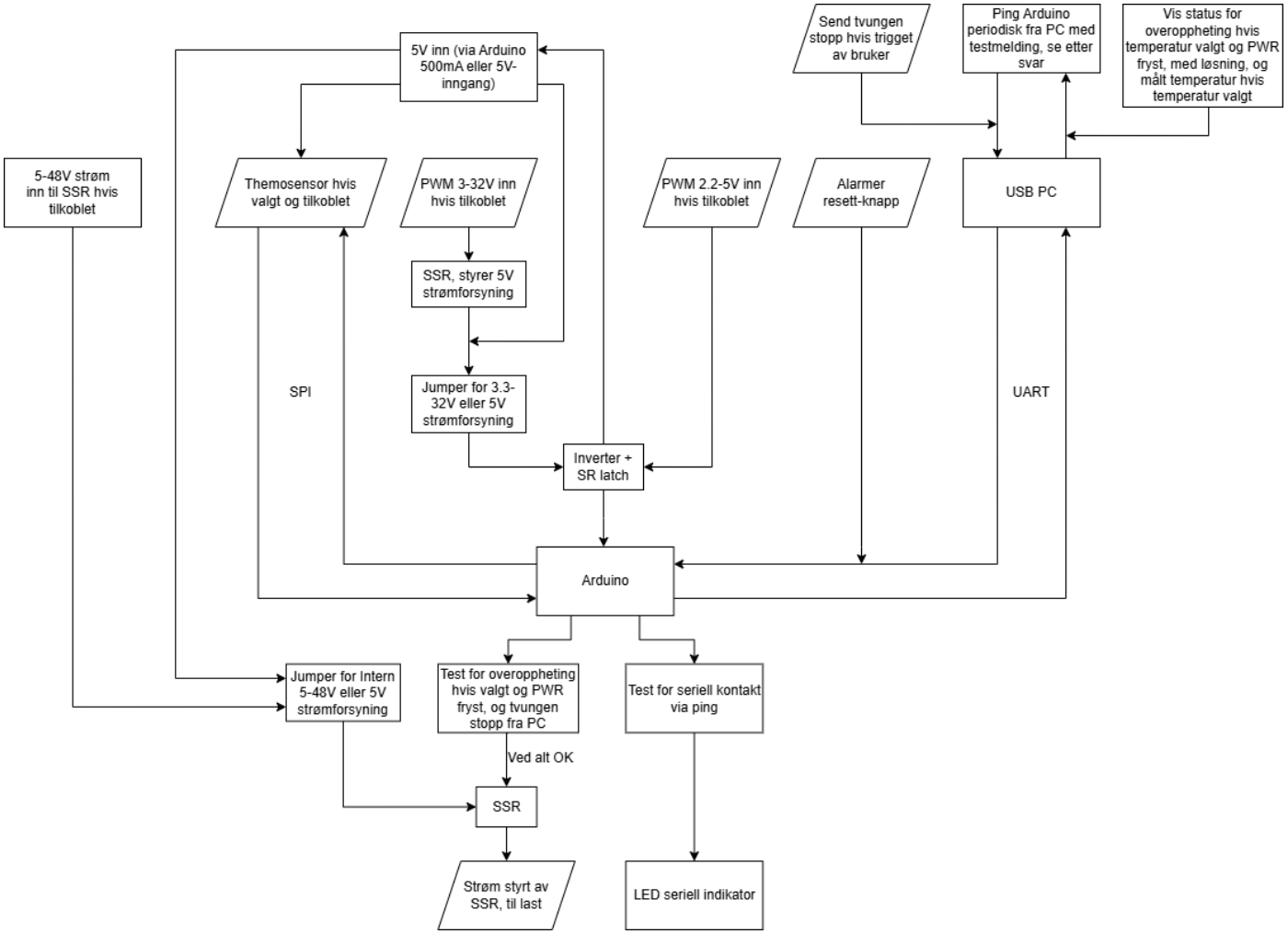
Systembeskrivelse

Systemet bygger på et modulært kretskort designet av Joakim Vigemyr i 2023 med kompakte Arduino Nano som styringsenhet. Strøm på under 500mA kan hentes fra Arduinoens 5V eller eksternt, både Arduino og ekstern spenning kan brukes samtidig. Kortet måler temperatur med MAX31856 og fuktighet med SHT85, filtrerer digitale signaler via innganger fra 3V til 32V med 3.3V ut, og har en watchdog-modul. Arduino kan styre spenningen til en last (som varmeelement via SSR i en furnace) fra Arduinoens 5V eller ekstern 5V–48V/60V AC, hvis et eksternt signal (3V-32V eller 2.2V-5V), som det overvåkede PWM-signalet PWM til furnace SSR, blir låst høyt for lenge. Valg av spenninger avhenger av jumperplasseringer og hva man bruker kortet til, se kretsdiagram (kun watchdog, fullt kort ligger under “diagrammer” i “misc-filer”).

Prosjektet mitt fokuserer på watchdog og kommunikasjon med PC, lagt opp for enkel og modulær videreutvikling. Ny watchdog for overtemperatur av lasten (ved valgt bruk av temperatursensor). Absolutt maks temperatur må settes lavere enn furnace sin reelle grense grunnet treghet. Sensorfeil, låst PWM eller overtemperatur trigger en alarmmodus der furnace SSR-spenningen kuttes, og sendes serielt til PC, sammen med temperaturavlesning hvis brukt. Det vises i et brukergrensesnitt hos PC. Ved temperatursensorfeil settes temperaturen til 2023 grader (etter Labview-standard), og overopphetingsmelding sendes ikke da. Alarm resett-knappen og restart av Python-programmet er nødvendig for alarmklarering.

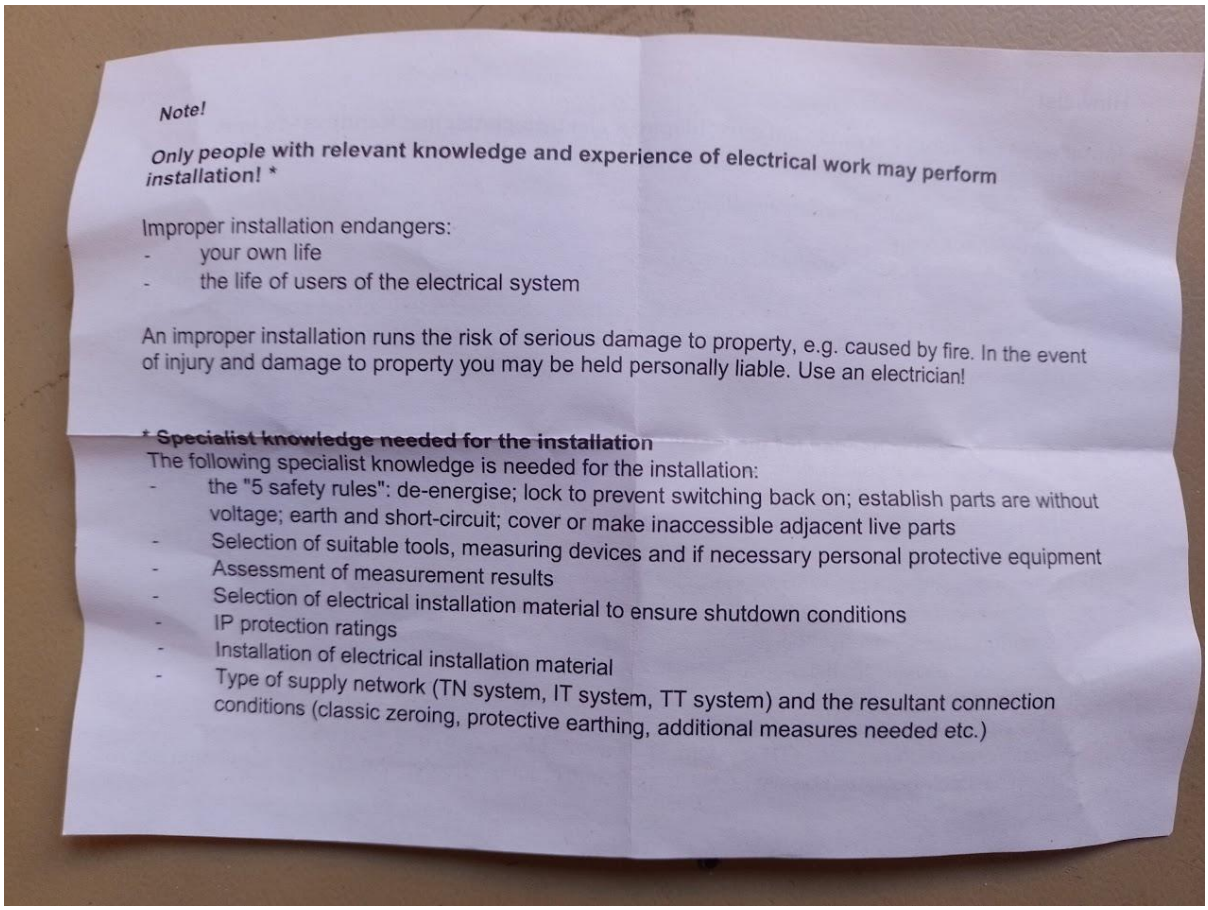
Systemet har automatisk gjenoppretting av seriell USB UART-kommunikasjon via handshake, og feilmeldinger sendes periodisk for pålitelighet. På PC vises data og feilmeldinger i et brukergrensesnitt der brukeren velger COM-port, og sender trigger for alarmmodus (en slags nødstopp). Koden er organisert i gjenbrukbare moduler som for temperatursensor, watchdoger, og seriell kommunikasjon, med en fleksibel, kommaseparert protokoll for enkel modifikasjon. Laget for robustivitet. Watchdog-modulen styrer furnace i real-time, mens Python-programmet fungerer hovedsakelig som brukerinterface med GUI (lagt opp for Labview-integrasjon i tillegg). For øyeblikket er furnace kun en testlast.

Hovedfunksjoner blokkdiagram



HMS

Medfølgende sikkerhetsinstrukser for service av ovnen



Note!

Only people with relevant knowledge and experience of electrical work may perform installation! *

Improper installation endangers:
- your own life
- the life of users of the electrical system

An improper installation runs the risk of serious damage to property, e.g. caused by fire. In the event of injury and damage to property you may be held personally liable. Use an electrician!

*** Specialist knowledge needed for the installation**
The following specialist knowledge is needed for the installation:
- the "5 safety rules": de-energise; lock to prevent switching back on; establish parts are without voltage; earth and short-circuit; cover or make inaccessible adjacent live parts
- Selection of suitable tools, measuring devices and if necessary personal protective equipment
- Assessment of measurement results
- Selection of electrical installation material to ensure shutdown conditions
- IP protection ratings
- Installation of electrical installation material
- Type of supply network (TN system, IT system, TT system) and the resultant connection conditions (classic zeroing, protective earthing, additional measures needed etc.)

Risikovurdering

Risiko 1; Farlig støt:

Sannsynlighet: 3/6. Konsekvens: 6/6.

230V fra lasten (furnace) kan være veldig farlig, spesielt ved hjerteproblemer og hos eldre. 400V er alltid dødelig. Ujordet utstyr kan også bli ædelagt og være farlig.

Konsekvens:

230V kan være livsfarlig, spesielt ved hjerteproblemer og hos eldre, og 400V er alltid dødelig. Ifølge loven skal man ikke eksponeres for høye spenninger i profesjonell sammenheng. Ujordet utstyr kan gå i stykker, eller enda være, føre farlig spenning. Konsekvens: 6/6.

Tiltak:

Bruk lovpålagt automatsikring, og forsikre god jording, som er krav for nyere utstyr. Ledninger og deler må være isolerte, sitte godt, ikke være løse, og utstyret må være fritt for jernspon og annet strømførende skrot. Verifiser at alle strømkabler er frakoblede.

Hvis spenningen absolutt må være på under service, må lærlinger være under oppsyn. Isolerte verktøy skal alltid brukes. Tenk før du gjør ting, og ikke hast. Montér chassis/panelet til utstyret. Forsikre at ledningene til og fra furnace sitt SSR faktisk er koblet til signalsiden av SSR-et. Dobbeltsjekk arbeidet før klarering for alle punktene.

Risiko 2; Brann og giftig røyk:

Sannsynlighet: 1/6. Konsekvens: 6/6

Det skal mye til, men konsekvensen er stor. Feil i systemet eller kortslutninger kan føre til brann med forbrenningsskade og farlig røykutvikling som følge. Hele bygget kan ta fyr. Man kan også brenne seg på furnace.

Tiltak:

Overtemperaturvern som del av varmt utstyr som furnace og i strømkursen (innebygd i automatsikring), er lovpålagt. Test 1-2 ganger i året. Watchdog-modulen er kun ekstra sikkerhet og for kommunikasjon med PC. Furnace er fortsatt varm, og kan bli mye varmere, etter at den er av, grunnet treghet. Furnace må kobles direkte i veggen uten mellomledd. Unngå unormalt høy strømgjennomgang av komponenter som kan påtenne brennbart materiale. Unngå kortslutninger. Gjør deg og andre kjent med rømningsveier og brannslukningsutstyr. Plastrøyk er ekstra giftig, bruk avtrekksvifte ved lodding og ikke brenn ledningsisolasjonen.

Risiko 3; Spenningsskade på elektronikken:

Sannsynlighet: 4/6. Konsekvens: 3/6.

Elektronikken kan skades av feilspenning, kortslutning, eller ESD (Statisk ElektroStatisk Utladning). Dette kan blir dyrt, og forsinker arbeidet hvis man ikke har reserver.

Tiltak:

Ikke send mer spenning til komponenter enn de tåler. Unngå strømledende, kortsluttende skrot og jernspon. Koble komponenter riktig, og unngå å kortslutte via andre komponenters komponentben eller måleprobene under måling. Ikke ta på kontakter, hold på siden av kort og komponenter, og bruk til og med ESD-armbånd og ESD-matte. Ullgenser og nylon fører ofte til ESD.

Risiko 4; Fysisk skade på folk og utstyr:

Sannsynlighet: 4/6. Konsekvens: 5/6.

Tungt utstyr og folk kan skades hvis utstyret går i bakken, man kan brenne seg p[ovnen, og man kan skjære eller kutte seg.

Tiltak:

Hold arbeidsbenken ryddig. Ting kan lett falle grunnet skumping eller flytting. Furnace kan falle av kanten hvis den roteres. Unngå at skarpe verktøy, inkludert skrutrekkere, kan glippe og treffe hånden. Dra furnace sitt underrack forsiktig ut med hånden over på hver side for å unngå klem. Vernesko kan beskytte føttene. Sørg for forsvarlig løfteteknikk. Aldri ta inni ovnen mens varmen er på.

Risiko 5; Psykologisk belastning over tid:

Sannsynlighet: 2/6. Konsekvens: 4/6.

Frustrasjon, støy, for lite pause, og repetitive bevegelser over tid, kan føre til mental utmattelse og redusert søvn, velferd, helse, fokus, og sikkerhet.

Tiltak:

Det er viktig å ta pauser ved behov, som å ta en tur i eller ved bygget, slå av en prat med noen, eller bare gjøre ingenting i en 5-10 minutter, midt i arbeidstiden. Overarbeid og utilstrekkelig hvile er kontraproduktivt, og kan selvforsterkes av stress. Ha realistiske og gode ambisjoner. Husk at det ofte er en ny dag for å fortsette arbeidet. Planlegg arbeidet godt, varier det, og spør kollegaer om hjelp. Dropp arbeidet foreløpig og gå ut av verkstedet ved kraftig støy.

Risikomatrise:

6	**2**		**1**			
5				**4**		
4		**5**				
3				**3**		
2						
1						
	1	**2**	**3**	**4**	**5**	**6**

Brukerdokumentasjon

Kun teknikere skal modifisere ovnen. Sett ovnen på en stabil overflate, ha flere til å løfte og la bena ta vekten, forsiktig ved rotering. Plugg ut kontakter først. Ikke ta inni ovnen når varm. Plugg den direkte i stikkontakten i veggen, uten mellomledd. Ta på sidene av kortet ved transport.

Denne watchdog-modulen styrer et apparat, som en furnace, via en signalinngang på lasten inn til lastens SSR. Lastsignalet kan enten være 5V (trenger ikke ekstern kilde) eller 5-48V. Den kutter lastsignalet hvis enten et enten 3V-32V eller 2.2V-5V signal er høyt over en gitt tid, hvis valgfri temperaturmåling via thermopar fra lasten (krever thermosensor MAX31856) er målt til over satt temperatur, eller thermosensor ikke i orden (sendt temperatur blir 2023 grader, standard for Labview). Du kan for eksempel ha samme signal til lasten som den som overvåkes. Status for temperatur, thermosensor ikke i orden, overtemperaturalarm, og overvåket signal sendes til PC og vises i programmet “interface”. Herfra kan du også sende et tvunget stoppsignal, en slags nødstopp. Trykk på alarm resett-knappen og restart interface for å klarere feilen. Laget for robustivitet og politelighet.

Det var mye fokus på modularitet med gjenbrukbare moduler i koden, og videreutvikling av systemet.

På watchdog-kretskortet er pluss alltid kontakten til venstre med kontaktene pekende mot deg, for pluss og minus-par. Pinnenummere for jumpere går i stigende rekkefølge fra venstre til høyre, og fra øverst til nederst.

1: PWM-styresignalet: Koble pluss og minus på styresignalkilden til CN1 sin pluss og minus og jumper J1 mellom pinne 1 og 2 ved 3V-32V styresignal, eller CN3 ved 2.2V-5V styresignal (her kobles J1 av fullstendig). Koble J1 mellom pinne 2 og 3 for å lure watchdog med PWM låst høy-signal.

2: Intern strømforsyning: Koble 5V inn på CN4 hvis totalt strømtrekk for 5V forsyning er mer enn 500mA (Arduino strømleveranse). Hvis ikke, hopp over dette trinnet.

3: Last SSR utgang: Koble last SSR inngang og utgang inn på pluss og minus på CN7.

4: Styrt spenning til last SSR: Ved 5V strømforsyning ut til last SSR: koble jumper J4 mellom pinne 1 og 2.

Ved 5V-48V ut til last SSR: Koble 5V-48V inn på pluss og minus på CN8 og jumper J4 mellom pinne 2 og 3. Ved bruk av PID blant annet, kan dette være samme som PWM styresignal.

5: Testjumper til last SSR: Koble på jumper J5 mellom dens to pinner for signal til last SSR inngang og utgang.

6: Thermosensor: Ved temperaturmålinger og temperaturwatchdog, sett en Adafruit thermosensor Max31856 breakout i sokkelen U8, og koble lastens thermopar pluss og minus til thermosensor Max31856.

7: Fyre opp strømmen minus Arduino: Ha PWM styresignal på, og slå på furnace og ekstern 5V strømforsyning hvis den er i bruk.

8: Innstillinger: Ved nye innstillinger, konfigureres de i programmet oven_controller hos Arduino. Hvis ikke, hopp over dette trinnet: Bruk av thermosensor1, dens CS-pinne ("SS" betyr default CS for Arduinoen), og ThermoCouple-type. SPI-protokollen som thermosensoren bruker, har unik Chip Select per sensor. Absolutt maks-temperatur for furnace1, tenk på overshoot grunnet treghet. Thermoinnstillingene ignoreres helt uten bruk av thermosensor.

Sett maksimum toleransetid for PWM styresignal låst høyt. Sett innganger og utganger for SSR kontrollpinne, og SR-vippe resett og output-pinner.

Alle pinnene er satt for Arduino mikro som default. Koble så Arduino til PC seriell USB og overfør nytt program til Arduino.

9: Fyre opp systemet: Koble Arduino til PC seriell USB hvis ikke i allerede, og start opp Python-programmet "interface". Følg instruksene. Se "Windows + x" -> "device manager" -> "Ports (COM and LPT)" for aktive COM-porter. Feilmeldinger vises også her.

10: Ved rapportert error: Klarér feilen ved å trykke på alarmresett-knappen til Arduino, og så restarte PC-programmet.

Teknisk dokumentasjon

Virkemåte hardware side

Jeg begynner med 3V-32V PWM-inngangen CN1 i kretsdiagrammet. I likhet med strømmen inn til inngangen CN3 er dette signalet som overvåkes av watchdog. I dette tilfellet er det PWM styresignalet til furnace SSR (Solid State Relay). Den styrer kortets SSR U2, som slipper gjennom 5V fra strømtilførselen via glattekondensator C4 og R1 som sikrer minst påkrevde 20mA fra SSR-et's UT. SSR-et bruker en optokobler, så signalet blir helt rent med C4. Dette er mest til bruk i støyfilterdelen av kretsen, men den er ikke relevant her.

Strømmen går videre til inverter U5 sin 1A (NOT-port), der den inverteres ut til 1Y. J1 må plugges inn mellom pinne 1 og 2. 2 og 3 kan brukes til å "lure" inverteren med høyt-signal til testing, med R4 som pull-up motstand. SR-vippe U6 sin utgang 1Q settes høy via 1S ved høyt-signal fra inverteren, og holder seg sånn til 1R er pulset av U1 Arduino fra dens D4 (sourcing current for Arduino). Sett vil overstyre Reset hvis begge er høy samtidig, derav SR og ikke RS.

1Q-signalet plukkes opp av Arduino sin D5 (sinking current for Arduino). SR-vippen er der for å gi Arduinoen tid til å reagere på CN1 og CN3 sine PWM-signeler, så kan den resette vippen når status er registrert i programmet. Inverteren gjør at vi får en kort høy-puls og ikke en lav-puls. Hvis ikke ville man måtte sende signalet til SR sin resett, og Arduino ville måtte sette vippen og se om den ikke blir resatt. Den er ikke rask nok til denne deteksjonen.

Inn til CN3 sendes 2.2V-5V signal, som tar samme rute som signalet inn til CN1, via inverter 2A og 2Y, SR-vippe 2S og 2Q, og Arduino D6; D4 på Arduino trigger både 1R og 2R samtidig fra D4. CN1-signalet er ikke brukt i Arduino-programmet, men kan lett implementeres. Hvis Arduino bekrefter signalet og, hvis bruker har med temperatursensoren og temperatur er under satt maximum verdi, blir strøm sluppet gjennom SSR U7 ut til utgangen CN7 via glattekondensatoren C8. Den styres av Arduino sin D7-utgang via R6 på 330Ohm. SSR-et skal bare ha 1,64V på styresiden, derav R6 for å danne en spenningsdeler. R9 er en pulldown-motstand for å unngå flytende signal (styrt av støy) fra SSR-et.

Jumper J5 ser ut til å gi mulighet til å koble ut GND etter SSR-et av testgrunner og for å kutte ut-strømmen. SSR brukes til å forsterke opp signalet opp til 5A, siden Arduino sin 20mA ofte er for lite til å styre eksterne SSR-er. SSR er fint å bruke fordi de kan bytte av/på raskt, og fordi de ikke får slitasje, men kan generer støy, spesielt hvis det ikke er

gjennomgangsrelé. Strømmen som SSR styrer kan enten være 5V fra strømforsyning eller 5-48V eksternet fra CN8, førstnevnte hvis man kobler jumper J4 mellom pinne 1 og 2, og sistenevnte mellom pinne 2 og 3. I mitt tilfelle vil PWM-signalet som overvåkes, være det samme signalet som sendes inn gjennom CN8 og styres.

Til CN5 kan en LED kobles til for eventuelle lyssignaler fra Arduino sin IO8, beskyttet av R5 mot 5V-2.2V = 2.8V for mye. Inn til CN4 kan man sende 5V fra ekstern strømforsyning via sikring F1 på 2.5A for å unngå full kortslutning eller hvis Arduino ikke er i bruk, eller hvis mer enn 500mA (maksimum strøm fra seriell USB, via Arduino) trekkes totalt av kortet og last.

Thermosensoren U9 MAX31856 måler temperaturen og sender den til Arduino. Den bruker SPI-protokollen, der man kan koble flere SPI-enheter på SDI, SDO, og felles klokke SCK sine linjer. Én unik signalinngang for CS (Chip Select), så Arduino kan sykle gjennom dem. Den får temperatursignalet fra et termopar, som måler varmen fra furnace og kjemisk konverterer til en veldig lav spenning, som sendes til MAX31856. Resett-knappen inn til CN10 resetter registrer alle aktive alarmer hos Arduino.

Sanntidskontroll og seriell kommunikasjon

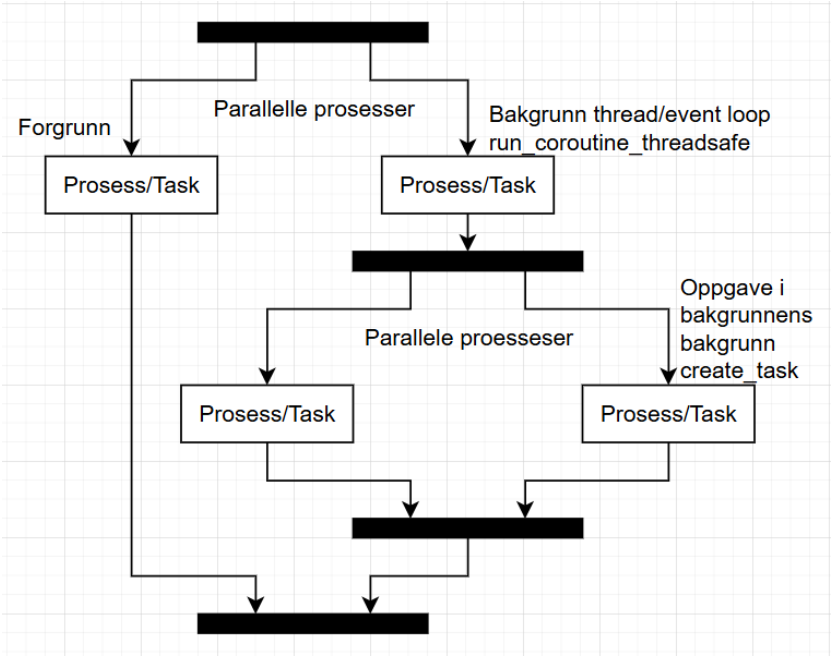
For en oversikt over selve programflyten, se flytskjemaene. De går inn på selve funksjonalitetsdelen i Arduino og Python-programmene, for seriell og handshake, håndtering av alarmstatuser og watchdog, GUI, etc. Som nevnt tidligere: Watchdog-modulen styrer furnace (lasten) i sanntid. Jeg gir her tilleggsinformasjon som ikke passer i flytskjema:

Sanntidskontroll Arduino: Har én hovedløkke som kjører momentant, som kjøres uendelig. Delay() er kun tillatt for ekstremt korte perioder (under 50 millisekunder). Man bruker i stedet mye statuser, if-setninger, og timeouts.

Sanntidskontroll Python: Bruker async og threading for å unngå at forgrunnen (mest Labview og GUI) må vente på operasjoner som sending og mottakelse av serielle meldinger og bakgrunnsrutiner (tasks) i bakgrunnen (se bilde for paralleleksempel). Asynkrone funksjoner (tidkrevende funksjoner som fungerer med async, defineres med “async def”), kjøres i en “event loop” i bakgrunnen, dette vil også funksjoner som den kalte gjøre. Alle event loops må kjøre i en thread.

run_coroutine_threadsafe kjører gitt asynkron metode i gitt event loop fra utenfra event loop-en (brukes ikke for ikke-asynkrone funksjoner), og create_task kjører prosesser parallelt med andre innad i en event loop. Her er det lettest å håndtere feil som oppstår

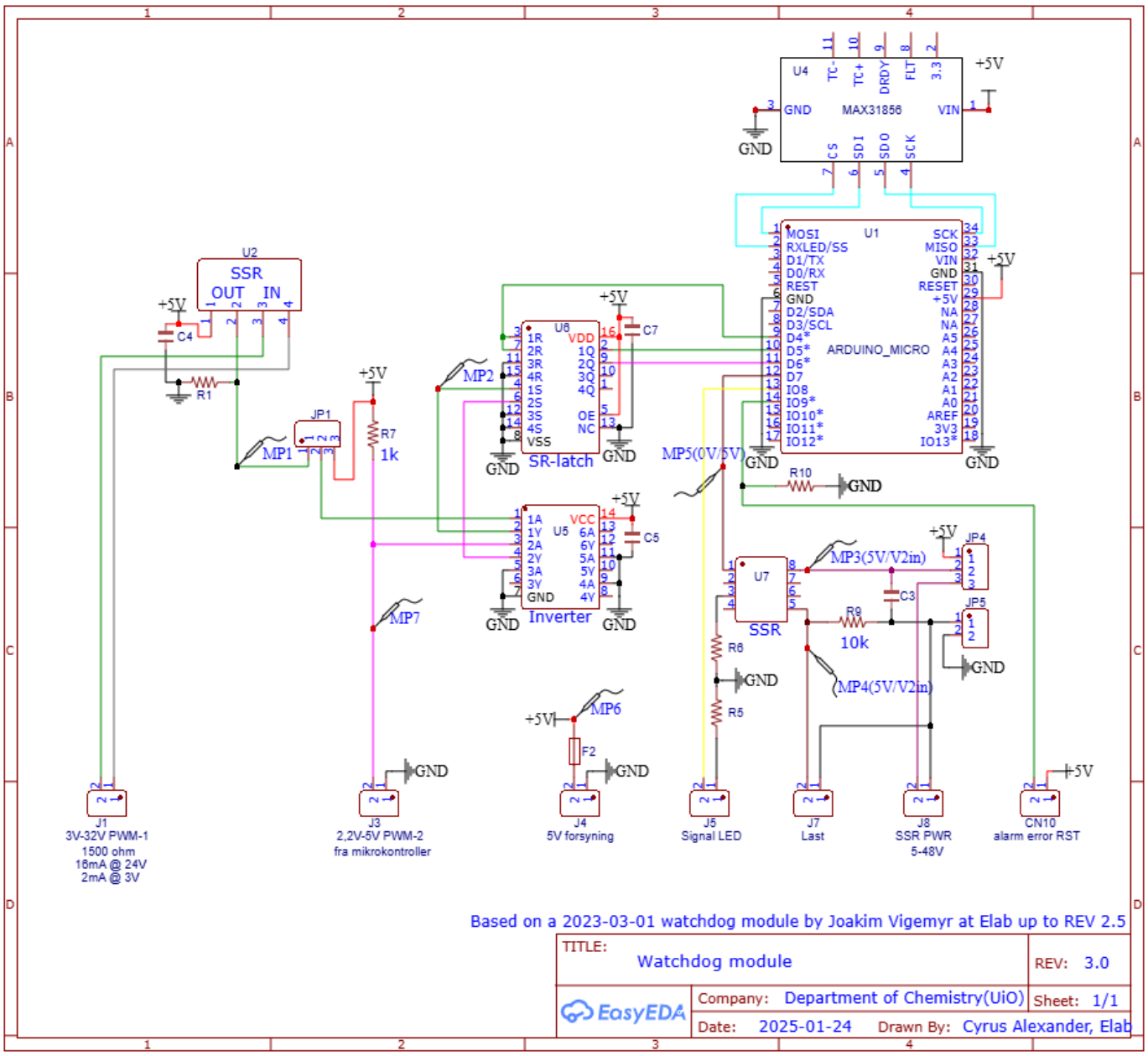
lokalt, eller ignorere den med en print/logging. En “lock” unngår at ressurser ikke endres akkuratt samtidig, som kan skape konflikter. Dette kan være fint for ting som tar litt tid. *Man trenger omtrent ikke å forholde seg til async eller multithreading.*



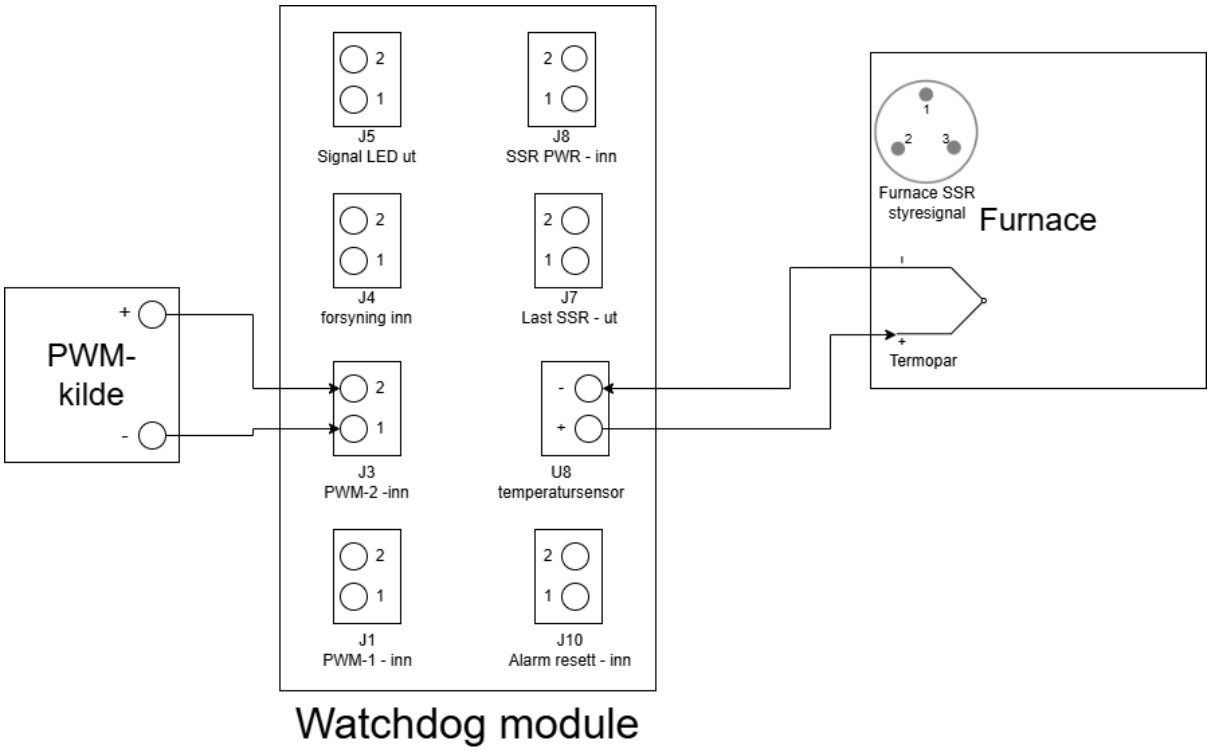
Meldingsformat og håndtering: Meldinger sendes som kommaseparerte komponenter i formatet kategori,kommando,verdi (valgfritt),timestamp (valgfritt, brukes foreløpig ikke). Meldinger dekodes/enkodes i en felles datastruktur og verifiseres i begge mottakende ender. De er lette å jobbe med for koderen. Python lagrer innkommende meldinger periodisk i bakgrunnen så de kan leses når som helst av avlesningsfunksjonen, mens Arduino behandler hver gyldig melding umiddelbart i hovedløkken; seriell er First In First Out i hardware.

Seriell gjenoppretting: I kjernen startes og gjenopprettes seriell kommunikasjon via en handshake der enhetene bekrefter kontakt via meldinger. Etter tre ping-forsøk over 15 sekunder fra PC (som venter svar fra Arduino), antas seriell kontakt tapt.

Kretsdiagram



Koblingsdiagram med last



Programdokumentasjon

Dette virker mye men gjør kodingen mye enklere når man har brukt tid på å forstå det som er relevant. Det kan være bra å lese “sanntidskontroll og seriell kommunikasjon” først.

Abstrakt oversikt over Arduino og Python-moduler

Se filen “abstrakt oversikt over Arduino og Python-moduler" i prosjektmappen for bruk av de viktigste objektene.

Flytskjemaer software

Det ble nevnt at hele koden ikke var ønskelig. Modifiserbare flytskjemaer ligger i filen “Misc -filer” i prosjektmappen som en XML-fil. Importer den i draw.io, Visio, eller andre flytskjemaplatformer som støtter XML. Den vanlige gir en høynivå oversikt, mens den detaljerte forklarer i detalj mekanismene bak.

Signaler

Kommando	Verdi?	Timest amp?	Navn og retning	Håndtering	Intervall i sekunder

					hvis relevant
1,1	Nei	Nei	ping_PC_arduino (handshake) PC til Arduino	Arduino vet at PC kan sende serielle data	5
1,2	Nei	Nei	Ping_Arduino_PC (handshake) Arduino til PC	PC vet at Arduino kan sende serielle data	Sent når seriell sjekk mottas fra PC
2,1	Ja	Nei	temperature_reading Arduino til PC	PC håndterer temperaturen	1
4,0	Nei	Nei	force_emergency_stop PC til Arduino	Start alarm hos Arduino som i tur tvinger varmen av	1
9,0	Nei	Nei	emergency_alarm Arduino til PC	PC håndterer alarm hos Arduino	1
9,1	Nei	Nei	thermosensor_error Arduino til PC.	PC håndterer thermosensor-error hos Arduino	1
9,2	Nei	Nei	watchdog_pwm_frozen Arduino til PC.	PC håndterer Watchdog PWM inn-signal fra SR-vippe frosset høyt	1
9,3	Nei	Nei	furnace_overheat Arduino til PC	PC håndterer overoppheting av last (furnace)	1

Meldingskategori (Kategori og kommando bygger opp en melding)	**Beskrivelse**
1	Serielle sjekker
2	Målinger og avlesninger
4	Spesielle kommandoer
9	Error

Komponentliste

Komponentene til furnace er ikke med

Arduino micro (U1) https://docs.arduino.cc/hardware/micro MPDCD3 (SSR) (U2) https://www.sensata.com/sites/default/files/a/mp-series-ac-pcb-mount-ssr-datasheet.pdf

MPDCD3 (SSR) (U2) https://www.sensata.com/sites/default/files/a/mp-series-ac-pcb-mount-ssr-datasheet.pdf

SN74ACT04D (Hex inverter) (U5) https://www.ti.com/lit/ds/symlink/sn74act04.pdf?ts=1674564486547&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FSN74ACT04%252Fpart-details%252FSN74ACT04D

HEF4043BT,652 (SR-latch) (U6) https://assets.nexperia.com/documents/data-sheet/HEF4043B.pdf

G3VM-61CR1 (SSR) (U7) https://docs.rs-online.com/8f70/A700000007327972.pdf

Adafruit max 31856 (MAX31856) (U9) https://www.elfadistrelec.no/Web/Downloads/_t/ds/Adafruit_MAX31856_amp_eng_tds.pdf

2-pin kontakt for kort (CN1, CN3, CN5, CN7) https://docs.rs-online.com/252a/0900766b81715679.pdf

2-pin kontakt for kort (CN4, CN8) https://docs.rs-online.com/c3eb/0900766b8138b923.pdf

Sikringsholdere (F1) https://docs.rs-online.com/4beb/0900766b81363f6a.pdf

Motstander:

180Ω ERA3AEB181V (R5) https://docs.rs-online.com/0512/A700000008177564.pdf 180Ω

200Ω ERJP08F2000V (R1) https://docs.rs-online.com/ef78/0900766b812cb812.pdf

330Ω ERJP14F3300U (R6) https://docs.rs-online.com/ef78/0900766b812cb812.pdf

1kΩ ERJP14F1002U (R7, R8, R9, R10) https://docs.rs-online.com/ef78/0900766b812cb812.pdf

10kΩ ERJP14F1002U (R10) https://docs.rs-online.com/ef78/0900766b812cb812.pdf

Kondensatorer:

0,1µF (C4, C6-C8) C1206C104M5UACTU https://docs.rs-online.com/15c2/0900766b81707628.pdf

47µF (C3) 293D476X9016C2TE3 https://docs.rs-online.com/4770/0900766b80d97b37.pdf

Prisliste

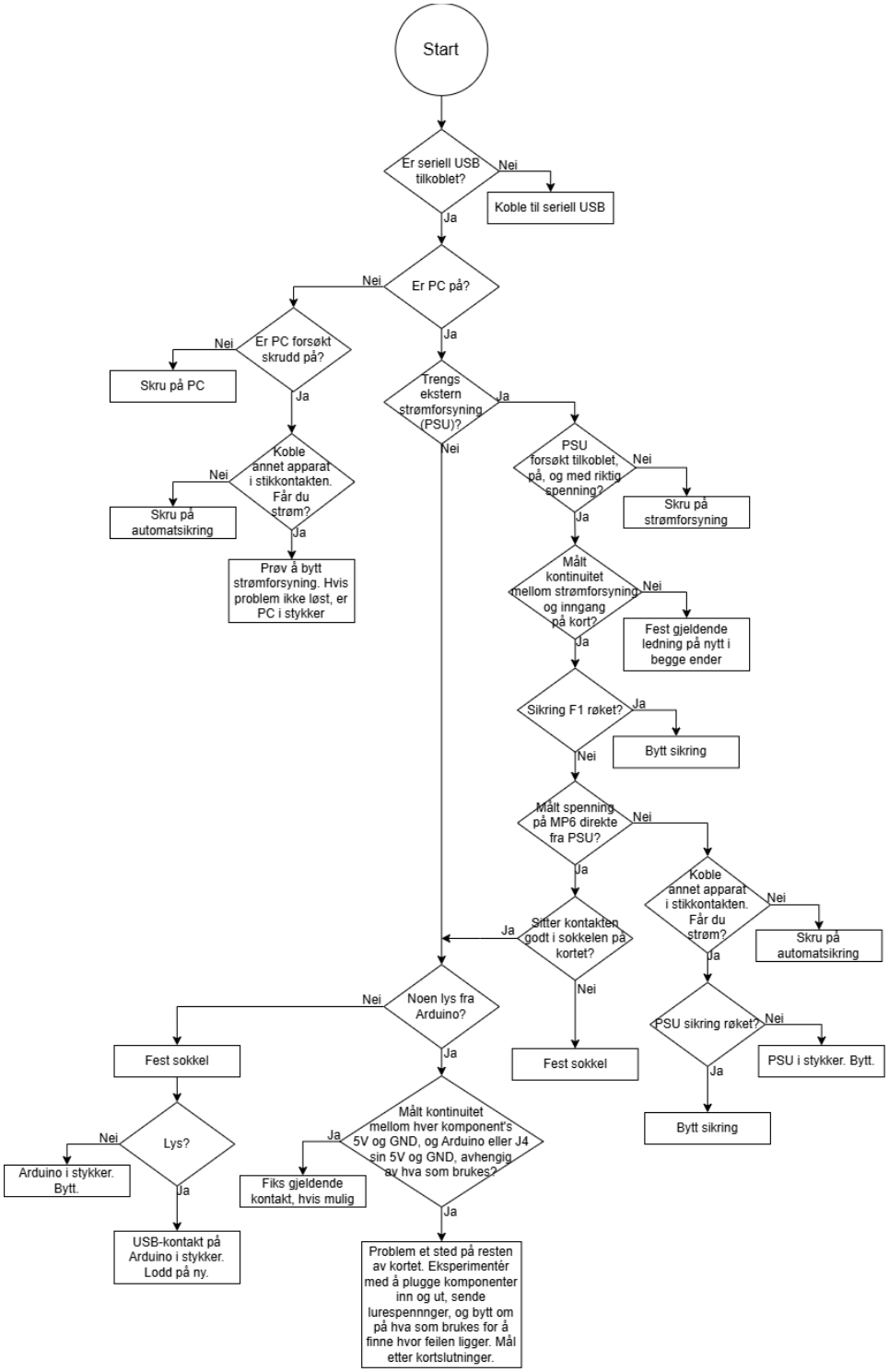
Komponent	Designator	Delnummer	Pris/stk kr ink moms	Antall	Pris kr ink moms	Leverandør
Arduino Micro	U1	771-7667	322,24	1	322,24	RS components
Thermokontroller	U4	3263	194,42	1	194,42	Adafruit
SSR	U2	241-3975	453,65	1	389,95	RS components
SSR	U7	213-2111	129,54	1	153,72	RS components
Hex inverter	U5	249-4979	3	1	7,15	RS components
SR latch	U6	771-HEF4043BT653	4,72	1	4,72	Mouser
Sikringsholder 5x20mm glassikring	F1	NAN	12,2	1	12,2	RS components
2-pin terminal skrue	J4, J8	425-8720	29,89	1	29,89	RS components
2-pin Phoenix contact skrue	J1, J3, J5, J7. J10	220-4658 + 220-4737	34,1 + 11,86 = 45,96	4	170,5 + 49,3 = 219,18	RS components
SMD motstand 180Ω	R5	566-412P	5,15	1	5,15	RS components

SMD motstand 200Ω	R1	721-9765	2,24	1	2,24	RS componets
SMD motstand 330Ω	R6	721-8110	5,23	1	5,23	RS components
SMD motstand 10kΩ	R10	721-8233	3,21	1	3,21	RS components
SMD tantalkondensator 0,1µF	C6-C8	684-5121	8,56	3	25,67	RS components
SMD tantalkondensator 47µF	C3	684-5121	10,7	1	10,7	RS components
PCB kretskort	NAN	NAN	93,32	1	93,32	JLCBCP
Total pris minus transport: 1489,6kr						

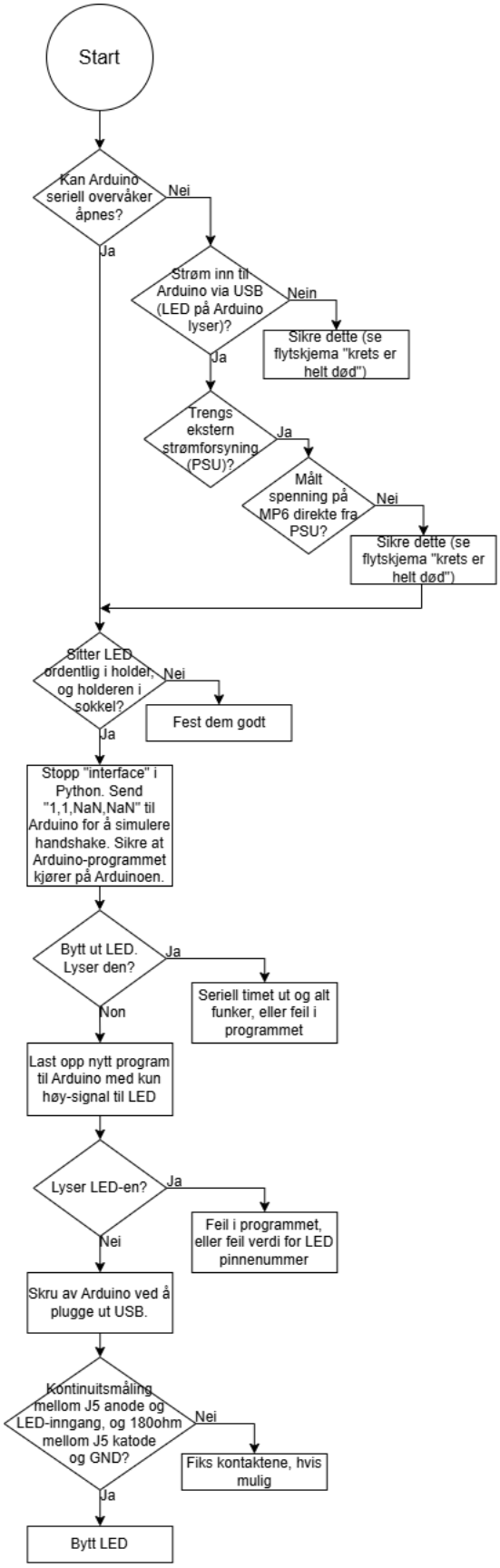
Feilsøkingsflytskjemaer

Jeg ble ikke helt ferdig grunnet IT-problemer i fem timer på siste Mandag. Jeg gjør resten innen midnatt.

Krets er helt død



LED1 lyser ikke


Start
Kan Arduino seriell overvåker åpnes?
Nei
Ja
Strøm inn til Arduino via USB (LED på Arduino lyser)?
Nein
Sikre dette (se flytskjema "krets er helt død")
Ja
Trengs ekstern strømforsyning (PSU)?
Ja
Målt spenning på MP6 direkte fra PSU?
Nei
Sikre dette (se flytskjema "krets er helt død")
Sitter LED ordentlig i holder, og holderen i sokkel?
Nei
Fest dem godt
Ja
Stopp "interface" i Python. Send "1,1,NaN,NaN" til Arduino for å simulere handshake. Sikre at Arduino-programmet kjører på Arduinoen.
Bytt ut LED. Lyser den?
Ja
Seriell timet ut og alt funker, eller feil i programmet
Non
Last opp nytt program til Arduino med kun høy-signal til LED
Lyser LED-en?
Ja
Feil i programmet, eller feil verdi for LED pinnenummer
Nei
Skru av Arduino ved å plugge ut USB.
Kontinuitsmåling mellom J5 anode og LED-inngang, og 180ohm mellom J5 katode og GND?
Nei
Fiks kontaktene, hvis mulig
Ja
Bytt LED

Hjelp av sluttbruker med ustabilt kort grunnet vibrasjoner

Jeg antar at bruker er en forsker med grunnleggende kunnskap om elektronikk.

Sikring: Sjekk om sikringen har røket ved å se etter brudd i tråden gjennom glasset, eller mål med multimeter. Bytt sikring hvis den røk. Går den igjen, koble ut alt inkludert strømforsyning, og løft kortet. Ta kun på siden for å unngå havarering.

Seriell kontakt: Kontroller seriell tilkobling via brukergrensesnittet eller LED-indikator. Sjekk USB-kabelen. Åpne den serielle overvåkeren (forstørrelsesglasset øverst til høyre i Arduino-programmet) og send "1,1,NaN,NaN". Du får "1,2,NaN,NaN" i retur. Dette kan nå brukes til feilsøking i stedet. Kommer ingenting grunnet timeout, send "1,1,NaN,NaN" på nytt.

Alarmresettknapp: Hvis feilen ikke kan klareres med reset-knappen, er feilen trolig fortsatt aktiv. Se etter feilmelding i brukergrensesnitt eller en melding som starter med "9" fra Arduino. Dobbeltsjekk at knappen er på plas i holderen, og holderen i sokkelen. Videre løsning i og under "kortets loddepunkter".

Thermosensorfeil: Ved melding "9,3,NaN,NaN" eller feilmelding i brukergrensesnitt, har du en thermosensorfeil. Sjekk at thermoparet sitter riktig i ovn og thermosensor, og at sensoren er godt festet i sokkelen. Prøv å restarte Arduino (koble fra/på strømkilde) og PC om nødvendig. Videre løsning i og under "kortets loddepunkter".

Signalkontakt til last: Bekreft at signalkontakten er tilkoblet. Hvis ja, bekreft at ledningene sitter riktig i kontakten på kortet med multimeter, eller inspiser. Vedvarer feilen, koble ut, mål kontinuitet med multimeter, eller send 5–48V direkte med last påslått. Lytt, se etter bevegelse, eller føl etter varme etter en stund, men vær veldig forsiktig med den varme overflaten. Hvis ingenting, bytt ut kabelen.

Kortets loddepunkter: Koble ut alt, inkludert strømkilden. Prøv å dytte på alle loddepunktenemed en tynn, stiv plastbit eller skrutrekker (ikke med fingeren, og hold kun på siden av kortet, for å unngå havarering). Lodd kontakten hvis den beveger seg.

Thermosensor: Hvis thermosensorfeilen vedvarer, kan sensoren være defekt og må byttes (hold sensorkortet på siden).

Trykknapp: Ved vedvarende feil med trykknappen, bytt den ut.

Videre hjelp: Er feilen fortsatt ikke løst, lodd alle punktene ved behov. Hvis feilen vedvarer, send e-post til utviklerne på (sett inn e-post her).

Teorispørsmål

1: Sikringer kan enkelt sjekkes med multimeter ved å bruke kontinuitetsfunksjonen (lydsymbol på multimeteret), som sjekker kontakt. Innstill for dette, og sett pluss og minus-probene på hver side av sikringen. Piper ikke multimeteret er tråden inni sikringen brutt, og sikringen er gått. Har man ikke multimeter kan man ofte også se bruddet gjennom glasset på sikringen.

2: Ohm's lov gir en sammenheng mellom strøm, spenning, og resistans (URI). Formelen er U = R * I, der U er spenningen, I er strømmen, og R er resistansen. Har du to, kan du finne den tredje ved å bytte om på formelen. Ekstra: Effektloven gir effekten i watt. Formelen er W = U * I, der W er effekten, U er spenningen, og I er strømmen. Man kan finne W hvis man kjenner to ting fra Ohm's lov (plugg inn i effektloven), eller U og I.

3: En Arduino er en mikrokontrollerplattform med det man trenger for å enkelt lage en mikrokontrollerkrets, i hovedsak strømforsyning, USB, programmerbar mikrokontroller (ofte av typen ATmega), GPIO-pinner med litt beskyttelse, innebygd klokke etc. Arduino Nano og Micro blant annet, er små og fine på kretskort. Raspberry Pi er egentlig en datamaskin med GPIO-pinner; den har Linux, man kan surfe nettet, enkelt koble til ordentlig tastatur og mus…Raspberry Pi 4B er omtrent like dyr som Arduino Uno og kraftigere med mer minne og CPU, men tar opp mer plass enn en Arduino. Arduino er ofte også enklere å jobbe med der man ikke trenger en datamaskin fordi færre ting kan forstyrre eller gå galt (systemkrasj etc).

4: Invertere gir en logisk NOT, altså at den gir et logisk høy-signal ut når den får logisk lav inn, og logisk lav ut når den får logisk høy inn. Hex Inverter har seks slike NOT-porter i én chip. De kommer ofte som CMOS eller TTL. Ekstra: har man NAND-porter (kun høy ut når begge innganger er lave), kan man lage en hvilken som helst logisk krets.

5: (seks var i oppgavene men ikke 5, antar 5): En optokobler, i likhet med blant annet reléer og en hvilken som helst SSR, lar en strøm passere når en strøm sendes inn gjennom styreinngangen. Den bruker en intern LED til styring, noe som gir et galvanisk skille mellom denne hovedstrømmen og styrestrømmen; mer effektivt enn halvlederne til SSR-er. Man unngår strømoverføring og minimerer støy mellom sidene ved riktig dimensjonering, som er fint for signalintegritet. I tillegg er de ganske raske i switchingen, ofte raskere enn nullgjennomgansreléer (type SSR). To ulemper er at de er ofte for svake for sterkere strømmer og spenninger, og at LED-en degraderes over tid.

Forbedringer/ting jeg angrer på

Lage et chassis til kortet med chassisplugger man kan koble seg inn ut og ut fra.

Implementere Labview-styring ved å ha funksjoner i Python, som i sin tur kjører samme funskjonalitet som GUI-en i bakgrunnen. Det finnes noen ressurser på nettet om dette. De må kommunisere sammen med felles delt minne lagret på PC, som en fil, fordi de vil kjøre som to ulike prosesser av samme programmet. Eller kanskje man kan velge å åpne GUI-en via Labview, i samme prosess? Enklere?

Hvis man sender en strøm inn til Arduino sin Vin, kan den kjøre i standalone med ekstern strømforsyning.

Ha en ja/nei som verdi i meldingene for alarmstatuser. Lagre status for aktive alarmer på PC i en fil. Python kan automatisk registrere at feilen er borte ved respektiv mottatt seriell melding.

PC sin serielle USB kan få for mye strømtrekk hvis man bruker denne som strømtilførsel sammen med ekstern 5V, og sikringen ryker. Dette kan skade PC i noen tilfeller.

Send strømmen fra Arduino 5V ut gjennom siring F1 også.

Lage en PID for temperaturregulering. PID-en kan tunes ved tilkobling til PC der det er lettere å kode opp, og resultatene av tuningen for temperaturer kan lagres der. PC styrer furnace av/på. *Eller* så kan alt gjøres og lagres på Arduino sin EEPROM så det fungerer standalone. Dette kan være en billigløsning for realtime-kontrolleren, på et eget av Joakim sine kort. Veldig mye kode kan gjenbrukes her.

En lakasjestrøm gjennom R7 gir 5v spenning til kretsen og dermed Arduino, i en ustabil zombie-modus, fra PWM kontrollsignal. Arduino resettes ikke når man kobler ut USB-en ved lav PWM, som hindrer restart.

Lage setup-prosedyre hvor innstillinger kan settes fra PC og lagres på Arduino automatisk.

Velge COM, eventuelle innstillinger, sendte meldinger, og annet fra en dropdown i GUI i stedet for å skrive inn direkte. Mer robust, mindre feilhåndtering, og enklere for bruker.

Gi status for temperatur, alarmer etc på en OLED eller LCD-skjerm direkte tilkoblet Arduino. Alternativt, re-implemeter blinkekodene fra Joakim sitt prosjekt, men unngå blokkering.

Hos Arduino kan det fungere å sjekke både 5V PWM-pinnen og 32V-pinnen på én gang, så slipper man å sette manuelt. Hvis én går høy er banen klar.

Re-implementere fuktighetsavlesninger som i Joakim sin kode, og kanskje bruke watchdog med dette?

Re-implementere spesifikke errorer for temperatursensor som i Joakim sin kode, den kan gi mange ulike feilmeldinger. Foreløpig er samme temperaturavlesning tre ganger en generell feil.

Hvis en farlig feil oppstår blir feilen løftet når Arduino retartes. Det må også Python-programmet! Hvis ingen alarmer er mottatt hos Python i løpet av en gitt tidsperiode med seriell kontakt, kan feilen løftes der også.

Meldingsprotokollen har ingen indikator for hvilken sensor/furnace/feil av en gitt type meldingen referer til. Vil man ha flere, er dette en enklere måte å gjøre det på. Krever modifisering av funksjonen som koder og dekoder meldingen, og datastrukturen hos Arduino og Python.

Lage automatisk seriell tilkobling fra Python uten at bruker må skrive COM-port når vi først har en handshake. Gjør at det blir valgfritt.

Hvis verdi eller timestamp ikke er med i meldingsprotokollen “kategori, kommando, verdi, timestamp”, sendes den som NaN. En identifikator for komponentene i stedet for å posisjonsbasere er mer effektivt for flere komponenter.

La GUI-en i Python-programmet i lese og tolke seriell status fra SerialConnectionManager direkte (se “abstrakt oversikt over Arduino og Python-moduler"), og la GUI tolke det selv. Man blir kvitt hele gui_queue og slipper trøbbel med instansieringsrekkefølgen (tydelig når man ser på koden).

Gi bruker mulighet til å sende en hvilken som helst seriell melding manuelt fra GUI.

Lister i GUI med PC-ens COM-porter med “Arduino” i beskrivelsen og tilgjengelige meldinger som kan sendes er enklere for bruker og mer robust enn å skrive dem.

Gi bruker mulighet til å skru furnace SSR av eller på manuelt fra GUI.

Bruk signal-LED-en til å gi status om seriell kontakt.

Gi signal fra Arduino til PC om furnace er av eller på.

Ha valg av modus hos Arduino hvor pinnenummere settes automatisk avhengig av valgt Arduino-modell.

Send signalet inn til last SSR tilbake til Arduino for å overvåke om kabelen og kontakter er intakt og koblet til.