Bacheloroppgave

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

Martin Kristoffer Gløsmyr
Sverre Graffer
Håkon Dahl Mathisen
Sacit Ali Senkaya

Varmereguleringssystem for testing av komponenter i termisk vakuum

Bacheloroppgave i elektro/elektroingeniør
Veileder: Sigurd Gossé
Medveileder: Mari Linnerud

Mai 2022

NTNU
Kunnskap for en bedre verden

Martin Kristoffer Gløsmyr
Sverre Graffer
Håkon Dahl Mathisen
Sacit Ali Senkaya

Varmereguleringssystem for testing av komponenter i termisk vakuum

Bacheloroppgave i elektro/elektroingeniør
Veileder: Sigurd Gossé
Medveileder: Mari Linnerud
Mai 2022

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

NTNU
Kunnskap for en bedre verden

NTNU

Kunnskap for en bedre verden

Institutt for teknisk kybernetikk

Institutt for elektroniske systemer

Bacheloroppgave (IELET2920 og IELET2900)

Bacheloroppgave

E2212 Varmereguleringssystem for testing av komponenter i termisk vakuum

Forfattere:
Martin Kristoffer Gløsmyr, Sacit Ali Senkaya, Sverre Graffer og Håkon Dahl Mathisen

2022

NTNU

Tittelside Bacheloroppgave BIELEKTRO

Oppgavetittel (norsk og engelsk):
Varmereguleringssystem for testing av komponenter i termisk vakuum

Heat control system for testing of components in thermal vacuum

Forfattere:
Martin Kristoffer Gløsmyr
Sacit Ali Senkaya
Sverre Graffer
Håkon Dahl Mathisen

Prosjektnummer: E2212
Innleveringsfrist: 20.05.2022
Gradering: [x] åpen
[ ] lukket

Studium: Elektroingeniør - BIELEKTRO

Studieretning: Automatisering og robotikk
Elektronikk og sensorsystemer

Veileder internt: Sigurd Gossé, sigurd.gosse@gmail.com, 91766106

Institutt: Institutt for teknisk kybernetikk
Institutt for elektroniske systemer

Oppdragsgiver: Orbit NTNU, Trondheim

Kontaktperson: Mari Linnerud, mari.linnerud@orbitntnu.com, 90034988
Fredrik Sommerfelt Grønvold, fredrsgr@stud.ntnu.no, 41606168

Sammendrag (norsk og engelsk):
Orbit NTNU er en teknisk organisasjon som konstruerer kubesatellitter. Før en slik satellitt skytes ut i verdensrommet er det viktig at den blir testet grundig, for å sjekke at den tåler vakuum-tilstanden i rommet og varmestrålingen som befinner seg der. For å teste at en satellitt tåler disse forholdene brukes det et termiske vakuumkammer for å undersøke dette. Orbit NTNU har et eget termisk vakuumkammer som de ønsker å forbedre og automatisere. I denne bacheloroppgaven skal prosjektgruppen se på mulige forbedringer til dette kammeret, utvikle et reguleringssystem som gjør at komponenten i kammeret oppnår ønsket temperatur uten manuell styring og gjøre oppsettet mer brukervennlig.

Orbit NTNU is a technical organization that constructs cube satellites. Before a satellite is launched into space, it needs to be tested thoroughly. The satellite has to withstand the vacuum of space and the heat caused by radiation. To test the satellite under these conditions, a thermal vacuum chamber is used. Orbit NTNU has a thermal vacuum chamber that they want to improve and automate. In this bachelor thesis, the project group is looking into different possibilities for improving the thermal vacuum chamber, developing a control system which regulates the temperature such that it reaches the desired setpoint without human intervention and making the setup more user-friendly.

Stikkord norsk:
elektronikk, programmering, reguleringsteknikk, termodynamikk, vakuum

Stikkord engelsk:
electronics, programming, control systems, thermodynamics, vacuum

Forord

I løpet av det siste semesteret i bachelor-utdanningen for elektroingeniører skal det gjennomføres en bacheloroppgave. Denne bacheloroppgaven utgjør 20 studiepoeng og er det avsluttende arbeidet studentene gjør i denne utdanningen. Bacheloroppgaven ble tildelt i slutten av desember og ble valgt av prosjektgruppa basert på den interessante problemstillingen Orbit NTNU hadde å tilby. Medlemmene i prosjektgruppa kjente hverandre fra før og meldte seg derfor opp i en samlet gruppe ettersom alle syntes dette var en tiltalende oppgave.

Denne bacheloroppgaven tar for seg all relevant teori, samt at den går igjennom alt arbeid som har blitt gjennomført og resultatene av dette.

Bachelorgruppa ønsker å takke alle forelesere og fagpersoner som opp gjennom studiet har gitt oss den kunnskapen vi trengte for å gjennomføre dette prosjektet. En spesiell takk går til Sigurd Gossé (veileder fra NTNU), Mari Linnerud (oppdragsgiver fra Orbit NTNU) og Fredrik Sommerfelt Grønvold (Testansvarlig fra Orbit NTNU), som har vært våre støttespillere i gjennomføringen av dette prosjektet. Disse personene har bidratt med god hjelp og kommet med konstruktive tilbakemeldinger som gruppa har fått mye bruk for. Videre rettes en takk til professor Damiano Varagnolo som hjalp med den matematisk modelleringen av systemet, og introduserte gruppa for LQR. En annen takk går til Arman H. Kermani ved institutt for materialteknologi, som ga oss tilgang til lab for å gjennomføre utgassingstest. Universitetslektorene Cevdet Islek og Torleif Anstensrud mottar også en takk for de konstruktive tilbakemeldingene de ga gruppa angående designet til det grafiske brukergrensesnittet, simuleringen av LQR og systemidentifikasjon. Siste takk rettes til Sindre Herstad, som hjalp gruppa med mekanisk arbeid tilknyttet verkstedet.

Signaturer

Martin K. Gløsmyr
Automatisering og robotikk

Sacit A. Senkaya
Automatisering og robotikk

Sverre Graffer
Elektronikk og sensorsystemer

Håkon D. Mathisen
Elektronikk og sensorsystemer

Innholdsfortegnelse

Liste med bilder og figurer

Definisjoner

Selfie: Selvportrett som i denne sammenheng refererer til at SelfieSat skal kunne ta et bilde av seg selv i bane rundt jorda [1].

Vakuum: Lufttomt rom. Det vil si et rom som ikke innholder molekyler eller andre partikler [2].

FTTR: Forkortelse for "Fancy Thermal Test Rig". Et navn på et system for å både generere varme via varmetråder, samt ta temperaturmålinger i vakuumkammeret.

PPS: Forkortelse for "Programmable power supply". En strømforsyning hvor utgangsspenningen kan settes fra en tilkoblet PC. PPS-en brukes som inngang til pådragsorganet.

TVC: Forkortelse for "Thermal vacuum chamber". Et termisk vakuumkammer er en innretning som brukes til å teste om de ulike satellittkomponentene tåler forholdene verdensrommet har å by på. En annen forkortelse som også brukes om dette er TVAC.

Utgassing: Når visse komponenter utsettes for vakuum vil de begynne å utgasse (miste masse). Denne prosessen kan minne om fordamping.

ECSS: Forkortelse for "European Cooperation for Space Standardization". Et samarbeid som setter brukervennelige standarder for romaktivitet.

Lineær spenningsregulator: Dette er en type spenningsregulator som har tre tilkoblinger. En felles, en for inngangsspenningen og en for utgangsspenningen. Den holder spenningen på utgangen konstant og strømmen til lasten går gjennom regulatoren. Regulatoren kvitter seg med den overflødige spenningen ved at det er et spenningsfall fra inngang til utgang. Dermed har denne typen regulator et stort effekttap sammenlignet med andre regulatorer og dette tapet blir til varme.

Brytertransistor: Den transistoren i transistorkretsen som som kobler inn og bryter strømmen til varmemattene.

BJT: Forkortelse for "Bipolar Junction Transistor". BJT er oppbygd av tre dopede områder: Emittter, Base og Kollektor. Basert på dopingen av de tre områdene kan en BJT være på NPN eller PNP form [3].

MOSFET: Felteffekttransistor hvor en sammenhengende halvlederkanal er isolert fra inngangen (gate). Fungerer ved at feltet fra inngangen påvirker hvor godt kanalen leder strøm.

IGBT: Står for "insulated gate bipolar transistor". Virkemåten kan minne om en kombinasjon av en MOSFET og en BJT, der signalet går inn til MOSFET-en som igjen styrer BJT-en. Disse tåler typisk en høy spenning og strøm.

PWM: Forkortelse for "Pulse-Width Modulation" eller "Puls-bredde modulasjon" som det heter på norsk. PWM er et signal som er formet som et firkantpulstog [4]. Pulslengden på PWM-signalet kan endres på, mens frekvensen forblir konstant. En egenskap ved PWM-signaler er hvor stor andel av periodetiden signalet er høyt. Andelen oppgis gjerne i prosent. Et PWM-signal vil ha en gjennomsnittsspenning som er proporsjonal med andelen av periodetiden hvor signalet er høyt, og kan variere mellom null og amplituden. Å styre effekten i store laster med PWM kan i teorien gi null effekttap i styrekomponentene. Dette hadde ikke vært mulig hvis effekten i lasten ble styrt med en lineær spenningsregulator.

Breadboard: Et kort med et fast mønster med tilkoblingsklemmer som gjør at det er raskt å montere og fjerne hullmonterte komponenter. Fungerer bra til testing av kretser under utvikling. Dette fordi endringer mellom hver test tar kort tid. Mye kortere enn å lodde. Er ikke like pålitelig som loddede kretser. Sårbart for dårlig kontakt.

Stripboard: Dette er et ferdigprodusert kort med hull i et rutenett. På baksiden er det parallelle linjer med kobberbaner. Disse kan brukes til å lage permanente kort med elektronikk. For å bryte eller dele opp de lange banene er det vanlig å bruke en drillbit, gjerne med et håndtak, som er litt større enn bredden på banen. Siden kretsen blir loddet kan de være mer pålitelige enn breadboard.

Perfboard: Ganske likt stripboard. Forskjellen er at her er det en liten "kobberøy" for hvert hull. Da må alt kobles sammen med ledninger.

PID-regulator: En regulator er en algoritme som beregner utgangsverdi (pådragsverdi) på grunnlag av avviket mellom målt verdi og ønsket verdi [5]. Hensikten med en regulator er å fjerne eller minimalisere dette avviket gjennom å påvirke prosessverdien med pådraget fra regulatoren. En regulator kan konstrueres på flere forskjellige måter, men den mest brukte er PID-regulator. En PID-regulator er satt sammen av følgende matematiske operasjoner: P (proporsjonal forsterkning), I (integralvirkning) og D (derivatvirkning).

LQR: Forkortelse for "Linear-quadratic regulator". En optimal regulator som tar hensyn til systemdynamikk og vektingsparametere bestemt av ingeniøren.

Raspberry Pi: Er en ettkortsmaskin på størrelse med et bankkort. Den er lite kostbar og derfor egner den seg godt til grunnleggende programmering for nybegynnere, men fungerer også ypperlig for viderekomne. Fordelen med Raspberry Pi framfor andre mikrokontrollere er at den kan kobles til internett. Dette tillater trådløs kommunikasjon med en ekstern maskin [6].

LCD: Liquid crystal display. På norsk flytende-krystall-skjerm er en tynn skjerm som vanligvis kommer uten bakgrunnsbelysning, og derfor egner seg godt ved lav spenning. En mindre skjerm som dette er brukt i denne oppgaven. Slike skjermer har ofte en begrensing for antall tegn som det er mulig å skrive til den [7].

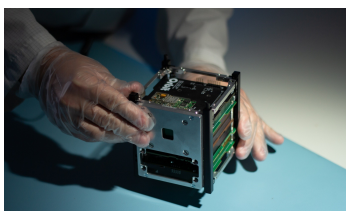
ADC: Analog-to-digital converter. På norsk analog-til-digital omformer. Enkelt forklart fungerer en ADC ved å konvertere en analog spenning eller strøm til et digitalt tall som representerer denne spenningen eller strømmen [8].

I^2C: "Inter-Integrated Circuit" eller I^2C ble utviklet av Philips Semiconductors i 1982 og er en seriell kommunikasjonsbuss, som opprinnelig ble brukt i TV-apparater. Nå brukes den til å koble sammen lavhastighets småelektroniske enheter til mikrokontrollere. I^2C benytter seg av master-slave systemet med linjer for seriell data (SDA) og seriell klokke (SCL) [9].

ROM: "Read-only memory" eller på norsk kun-lese-minne er som navnet tilsier minne for lagring som kun leses. ROM brukes ofte for å lagre fastvare siden den ikke krever strøm for å fungere. I dette prosjektet er det temperaturdata fra digitale sensorer som lagres på ROM [10].

Avstandstykke: Et avstandstykke er en liten hul metallisk sylinder som skal sørge for at to objekter adskilles.

Kubesatellitt: Kubesatellitter er små modulære satellitter som er utformet som kuber, derav navnet kubesatellitt [11].



Figur 1: Kubesatellitt laget av Orbit NTNU [12]

.

1 Innledning

1.1 Bakgrunn

Orbit NTNU er en teknisk studentorganisasjon som er lokalisert på Gløshaugen i Trondheim [13]. Organisasjonen har omtrent 60 medlemmer og holder til i øverste etasje i D-blokka på elektrobygget. Formålet med Orbit NTNU som organisasjon er å skape interesse for romteknologi og trene fremtidens romteknologer. Dette gjøres ved å iverksette og gjennomføre prosjekter som dreier seg om å utvikle ulike kubesatellitter som skal sendes ut i verdensrommet. Disse kubesatellittene bygges i ulike størrelser, alt ifra 1U (10cm x 10cm x 10cm) til 24U. Kubesatellittene skal være operative i verdensrommet, slik at de kan utveksle informasjon med mennesker som befinner seg på jorda. I skrivende stund holder organisasjonen på med to satellittprosjekter. Et av disse satellittprosjektene er "SelfieSat". Dette prosjektet omhandler en satellitt som skal kunne ta "selfie" av seg selv mens den går i bane rundt jorda. Satellitten har en liten skjerm montert på kroppen og et kamera montert på en stang. Skjermen kan vise bilder som er sendt fra jorda. Kameraet kan ta bilde av skjermen på satellitten med jorda i bakgrunnen.

Figur 2: Logoen til studentorganisasjonen Orbit NTNU [13].

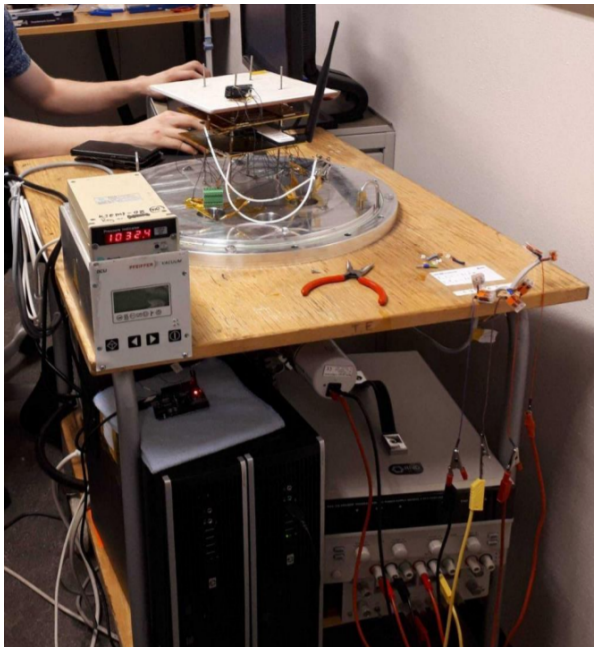
Før en eventuell satellittoppskytning er det viktig å vite at satellitten tåler forholdene som befinner seg i verdensrommet. De to mest nevneverdige forholdene som satellitten til en viss grad må kunne tåle er vakuum og varme i form av stråling. Det finnes flere andre forhold i tillegg til de nevnte (f.eks. partikkelstråling), men disse er ikke særlig relevante for denne bacheloroppgaven. Vakuum er et forhold som er viktig å ta hensyn til ettersom det kan føre til at visse komponenter starter å utgasse. Utgassing gjør at disse sensitive komponentene taper masse. Denne tapte massen kan legge seg på ømfintlige komponenter og potensielt ødelegge disse. Varme i form av stråling kan derimot føre til at elektronikken ombord oppvarmes til en temperatur som gjør at satellitten stopper å fungere.

For å forhindre at dette skjer er det nødvendig å teste satellittkomponentene nede på jorda før de skytes ut i verdensrommet. Testing av disse komponentene gjennomføres i termiske vakuumkammere. Disse testene er standardiserte og innebærer en del ulike krav som er spesifisert i ECSS. Orbit NTNU har allerede utformet ett slikt vakuumkammer, men dette kammeret er tregt og tungvint å operere.

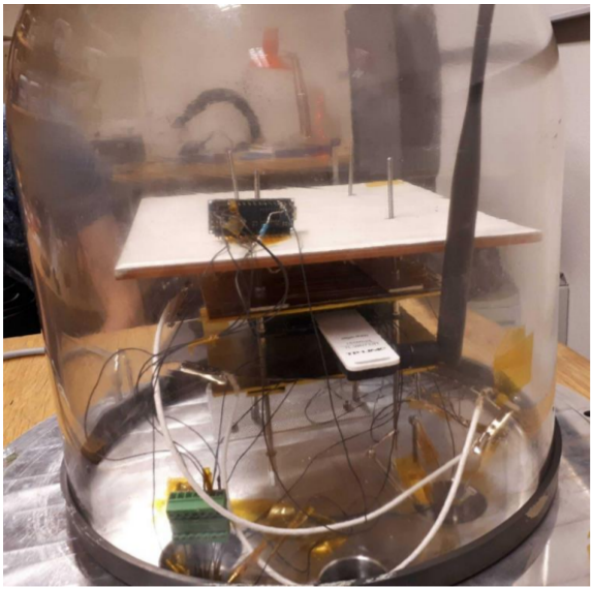
Hovedhensikten med denne bacheloroppgaven er å automatisere bruken av det termiske vakuumkammeret til Orbit NTNU, slik at testingen av kubesatellittene blir enklere å gjennomføre og tar mindre tid. En særedeles viktig del av dette er å innføre et reguleringssystem slik at komponenten som blir testet oppnår spesifisert testtemperatur uavhengig av menneskelig tilsyn. Det er også hensiktsmessig at reguleringssystemet oppfyller spesifikasjonene som er stilt og at ECSS standardene (ECSS-Q-ST-70-02C og ECSS-Q-ST-70-04C) følges. Et annet viktig moment er at systemet skal kunne styres- og overvåkes på en brukervennlig måte.

1.2 Oppgaveteksten

Oppgaveteksten som ble gitt av Orbit NTNU er lagt ved bacheloroppgaven som et vedlegg. Kort oppsummert sier oppgaveteksten at prosjektgruppa skal se på muligheten til å automatisere bruken av det termiske vakuumkammeret. Enten ved å ta utgangspunkt i det eksisterende oppsettet eller å utvikle et eget som er innenfor den økonomiske begrensningen Orbit har satt. Løsningen av prosjektet skal resultere i et system som er kompatibelt med vakuum og har plass inne i det eksisterende kammeret. Et viktig moment er at vakuum-delen av systemet allerede er tatt hånd om via en ferdig løsning (diverse pumper etc.). Det er varmedelen som er sentral for denne bacheloroppgaven.



Figur 3: Det tidligere oppsettet.



Figur 4: Det tidligere vakuumkammeret i test.

1.3 Rapportens oppbygging

Bacheloroppgaven er delt inn i elleve hovedkapitler hvor hvert kapittel tar for seg de viktigste delene av prosjektet.

- "Teknisk del" - kapittel 2, formidler all teknisk informasjon knyttet til forståelse rundt prosjektforslaget. Dette innebærer også prosjektmål, spesifikasjoner og problemområder.
- "Arbeidspakker" - kapittel 3, foretar en nedbryting av prosjektet i flere arbeidspakker. En "arbeidspakke" er definert som en identifiserbar selvstendig delaktivitet. Hver arbeidspakke tar for seg hvilke delaktivitet som skal utføres og hvor mye tid hver arbeidspakke krever.
- "Prosjektorganisering" - kapittel 4, beskriver alle forhold omkring organisering av prosjektet. Alt ifra prosjektdeltakere til tidsplanlegging.
- "Design" - kapittel 5, beskriver alt om design og oppsett av løsningen laget av bachelorgruppa. En gjennomgang av komponenter og hvordan disse fungerer er sentralt her, samt hvordan alt er koblet sammen.
- "Teori" - kapittel 6, beskriver alt om matematikken og fysikken bak systemet. Modellering og simulering utgjør mesteparten av dette kapittelet.
- "Regulering" - kapittel 7, beskriver alt om hvordan reguleringssystemet implementeres og hvordan det innstilles for å oppnå ønsket resultat.
- "Brukergrensesnitt" - kapittel 8, beskriver alt om hvordan brukergrensesnittene implementeres og hvordan de er designet for å være mest mulig brukervennlig.
- "Resultat" - kapittel 9, beskriver resultatene av arbeidet som ble utført.
- "Diskusjon" - kapittel 10, tar for seg mulige forbedringer og alternativer til arbeidet som har blitt utført.
- "Konklusjon" - kapittel 11, resultatet og diskusjonen blir konkludert i dette kapitellet.

2 Teknisk del

2.1 Problemstilling

Vakuumkammeret gruppa overtok i starten av bacheloroppgaven hadde vært gjennom tre iterasjoner tidligere. Den tredje versjonen av det termiske vakuumkammeret var relativt godt dokumentert med egne manualer som prosjektgruppa har fått tilgang til [14]. Gruppa fikk i tillegg en praktisk demonstrasjon av det termiske vakuumkammeret helt i starten av prosjektarbeidet.

Versjon tre av vakuumkammeret besto av FTTR, en PPS, to pumper (backing og turbo) med vannkjøling, en trykksensor og selve kammeret. FTTR er noe Orbit hadde utviklet selv, og besto av to pådragsorganer og et system for datainnsamling. De to pådragsorganene var to ulike plater som hadde hver sine varmekabler viklet rundt seg. Varmen kom av at pådragsorganene ble påtrykt en effekt fra PPS-en. FTTR ble effektstyrt manuelt ved å endre på spenninga til PPS-en [15]. PPS-en ble igjen styrt av en Raspberry Pi 3A+. Denne Raspberry Pi-enheten tok seg også av temperaturdatainnsamlinga som ble gjort av fem digitale temperatursensorer som ble plassert rundt omkring inni kammeret (testobjekt, pådragsorgan etc.).

Figur 5: PPS-en som blir brukt i denne bacheloroppgaven.

Vakuum-tilstanden blir oppnådd ved bruk av to pumper. Først kjøres backing pumpa som trekker trykket i vakuumkammeret fra 1 atm til omlag 0,3 mBar. Når kammeret har nådd 0,3 mBar tar turbopumpa over og trekker trykket helt ned til 100 nBar. Pumpene styres manuelt. Trykksensoren i oppsettet brukes til å lese av trykket i kammeret, som blir fremstilt som spenning på et multimeter. Spenningen tilsvarer et trykk som leses av på en tabell.

Figur 6: Backingpumpa som brukes i oppsettet.

Erfaringene Orbit hadde med bruken av denne versjonen av det termiske vakuumkammeret var først og fremst at det var veldig tregt. Det tok lang tid for å først senke trykket slik at man oppnådde vakuum, og deretter varme opp testobjektet til ønsket temperatur. Dessuten var den manuelle effektstyringa tidskrevende ettersom en person hele tida måtte stå og følge med på kammeret. Oppsettet var også relativt lite brukervennlig.

Ideen med å automatisere TVC var svært god og det fantes flere mulige løsninger til denne oppgaven. Prosjektgruppa bestemte seg tidlig for å lage en ny versjon av det termiske vakuumkammeret. Denne nye versjonen vil innføre nye og bedre pådragsorganer, samt et eget system for regulering av komponenttemperatur. Denne reguleringen skal utføres ved hjelp av en selvkodet PID-regulator som sender PWM-signal til en egenutviklet transistorkrets, som igjen forsyner pådragsorganet via PPS-en. I tillegg skal det utvikles brukergrensesnitt som gjør vakuumkammeret enklere å betjene og mer brukervennlig. Disse forslagene er gjennomførbare basert på tidligere kunnskap som studentene har opparbeidet seg gjennom studiet.

2.2 Prosjektmål

2.2.1 Effektmål

Effektmålene beskriver hva oppdragsgiver vil oppnå med prosjektet. Følgende effektmål ble satt:

- Effektivisere bruken av det termiske vakuumkammeret slik at tidsforbruk forbundet med operasjon reduseres.
- Brukervennlig oppsett som gjør det enklere å gjennomføre testing.
- Forbedre dagens oppsett med å implementere automatisk styring og bedre pådragsorganer.
- Løsningen skal være enkel å ta over i etterkant, slik at den eventuelt kan videreutvikles i fremtiden.

2.2.2 Resultatmål

Resultatmålene beskriver hva som konkret skal foreligge som resultat når prosjektet er ferdig. Følgende resultatmål ble satt:

- Et reguleringssystem som følger ønsket temperatur automatisk, slik at man skal slippe å styre effekten manuelt. Systemet skal oppfylle de spesifikke kravene fra oppdragsgiver.
- Bytte ut pådragsorganene med noe som vil fungere bedre og innehar en lavere tidsforsinkelse.
- Brukergrensenitt som logger data og framviser disse på en grei måte. Skal også kunne interagere med det termiske vakuumkammeret (f.eks. endre parametere).
- Løsningen skal være kompatibel med vakuum.

2.2.3 Prosessmål

Prosessmålene beskriver forventet effekt av prosjektarbeidet for gruppemedlemmene. Følgende prosessmål ble satt:

- Bli bedre kjent med fysikken tilknyttet termodynamikk.
- Benytte tidligere lært kunnskap i en reell problemstilling.
- Bli vant til å jobbe for en ekstern oppdragsgiver, og drive med prosjektadministrasjon.

2.3 Spesifikasjoner

I det første møte med oppdragsgiver ble følgende spesifikasjoner presentert for prosjektgruppa:

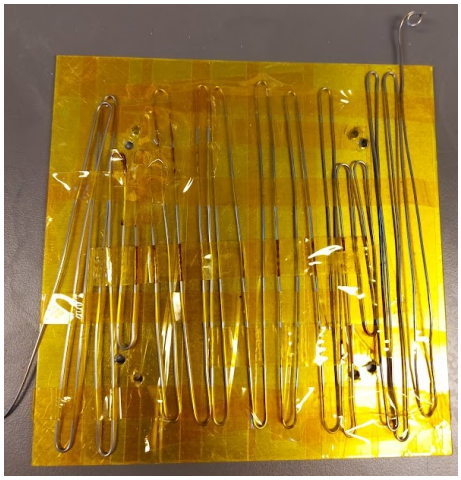
- Brukervennlig oppsett
- Prosjektet skal være enkelt å ta over i etterkant av at bacheloroppgaven er ferdig.
- Skal følge ECSS.
- Skal være kompatibelt med vakuum.
- Reguleringssystemet skal regulere temperaturen til testobjektet slik at temperaturen stabilt holder 120 grader (± 5 grader).
- Reguleringssystemet skal ha et innsvingningsforløp med minimalt oversving.

2.4 Problemområder

- Det er ikke mulig å innføre et eget system for avkjøling.
- I tillegg til generell komponentmangel i verden kan kinesisk nyttår i månedskiftet januar/februar føre til forsinkelser.
- Usikkerheten til trykksensoren kan føre til skade på turbopumpen dersom avviket på avlesningen blir for stor.
- Ulike ideer for forbedring av pådragsorganene kan være gode, men kan bli vanskelige å gjennomføre med tanke på vakuum-kompatibilitet.
- Emnet Ingeniørfaglig systemtenkning kan tar mer tid en forventet. Det vil dermed bli mindre tid til å gjennomføre bacheloroppgaven.

2.5 Prosjektbeskrivelse

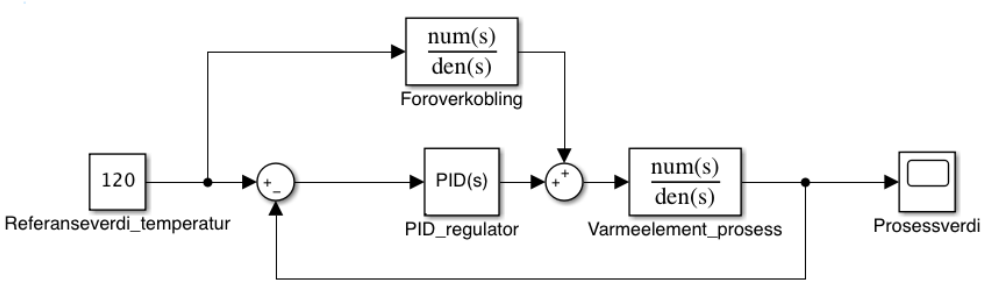
Ut ifra spesifikasjonene ovenfor ble det bestemt ganske tidlig hva gruppa skulle jobbe med. Å anskaffe nye pådragsorganer ble ansett som ganske viktig tidlig i oppgaven, ettersom de eksisterende var av svært dårlig kvalitet. Disse pådragsorganene var egenproduserte og hadde veldig varierende varmeutstråling. Et annet problem med dem var at de veldig lett kunne kortslutte ettersom varmeledningene ikke var isolert, og kun var teipet fast til metallplatene.



Figur 7: Et av pådragsorganene som ble brukt i versjon 3 av det termiske vakuumkammeret.

Videre ble det bestemt at gruppa skulle lage en matematisk modell av systemet for å forstå fysikken bak bedre. Denne modellen kan senere brukes til å simulere systemet, og finne riktige parametre for PID-regulatoren. Modellen kan senere også brukes til å implementere en modellbasert regulator (f.eks. LQR).

Etter at den matematiske modellen er på plass skal gruppa implementere et reguleringssystem for å regulere temperaturen til testkomponenten som skal testes i kammeret. Løsningen gruppa valgte å gå for innebærer å utvikle en egen separert elektronisk transistorkrets. Denne mottar et PWM-signal fra en PID-regulator implementert på Raspberry Pi, som igjen styrer hvor lenge pådragsorganene skal være aktive. Gruppa skal i tillegg prøve å innføre foroverkoblinger i koden til PID-regulatoren, for å oppnå mer robust regulering. Hvis gruppa får mer tid til overs skal det prøves å implementere en LQR for optimal regulering.



Figur 8: Figur som viser tenkt utforming av reguleringssytemet.

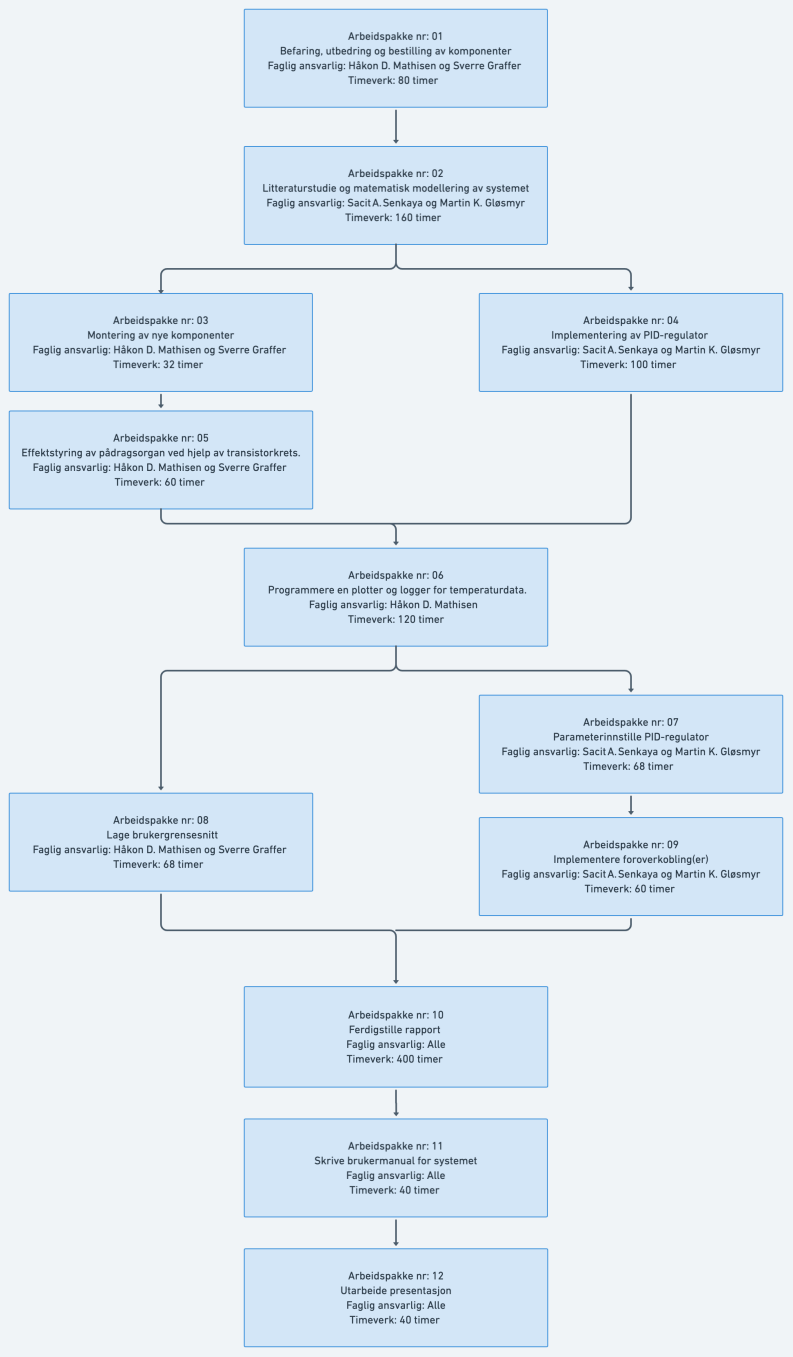
Det siste arbeidet som ble planlagt for oppgaven var å utvikle et brukergrensesnitt. Gruppa hadde lyst til å lage to slike: et fysisk bestående av LCD-skjerm og knapper, og en GUI (Graphical User Interface) på en skjerm koblet til Raspberry Pi. Det fysiske brukergrensesnittet skulle være relativt simpelt i forhold til det grafiske, som skal inneholde sanntidsplotting og tilgang til mer avanserte innstillinger.

GUI			Det fysiske		
Variabel	Leses	Skrives	Variabel	Leses	Skrives
Referanse	X	X	Referanse	X	X
Prosessverdier	X		Prosessverdier	X	
Manuelt pådrag	X	X	Manuelt pådrag	X	X
Auto/manuell	X	X	Auto/manuell	X	X
Regulatortype	X	X	Regulatortype	X	
Pådragsverdi	X		Pådragsverdi	X	
Foroverkobling	X	X	Foroverkobling	X	
Parametre	X	X	Parametere		

Figur 9: Figur som viser lese- og skriverettigheter til de to brukergrensesnittene.

3 Arbeidspakker

De ulike arbeidsområdene tilknyttet dette prosjektet ble nedbrutt i såkalte arbeidspakker. En arbeidspakke er en selvstendig arbeidsaktivitet med en spesifisert avsatt tidsperiode, timeverk og prosjektmedarbeidere. Arbeidspakkene er nummerert etter rekkefølgen på aktivitetene. Dette prosjektet ble delt inn i 12 arbeidspakker. Forprosjektet inngår ikke som en arbeidspakke, siden oppretting av arbeidspakker er en del av forprosjektet. I bildet nedenfor visualiseres rekkefølgen på arbeidspakkene i et flytskjema.



Figur 10: Flytskjema som viser rekkefølgen til arbeidspakkene.

Alle arbeidspakkene er lagt til i PA-mappen. Der står det mer i detalj om selve aktiviteten, prosjektarbeidere, timeverk etc.

4 Prosjektorganisering

4.1 Prosjektdeltagere

Figur 11: Bilde av Håkon.

Håkon Dahl Mathisen er 24 år og kommer fra Mo i Rana i Nordland fylke. Han gikk studiespesialiserende med retning realfag på videregående skole. Etter videregående gikk han ett år på folkehøgskole og ett år med å ta opp fag som privatist. I 2019 startet han på elektroingeniør ved NTNU i Trondheim og valgte retning elektronikk og sensorsystemer. Han har praktisk erfaring fra prosjekt i emnet IELET2109 Elektronikkonstruksjon og IELET3109 Avanserte sensorsystemer. Prosjektet i Elektronikkonstruksjon gikk ut på å designe og produsere et kretskort for spenningsregulering. Prosjektet i Avanserte sensorsystemer gikk ut på å designe og produsere et kretskort for å måle EKG-signaler fra kroppen. I bacheloroppgaven har Håkon medansvar for det elektriske ved systemet og hovedansvar for brukergrensesnittet.

Martin Kristoffer Gløsmyr er 21 år og kommer fra Skien i Telemark fylke. Han gikk studiespesialiserende med retning realfag på Skien videregående skole. Etter videregående startet han med ingeniørstudier på NTNU i Trondheim. Der studerte han elektro og spesialiserte seg innenfor automatisering og robotikk. Martin har erfaring som sommervikar hos ABB i Skien. Der jobbet han med å montere 12-24 kV bryteranlegg. Han har også praktisk erfaring fra Automatiseringsprosjekt-emnet (IELET2104). Automatiseringsprosjektet dreide seg om å implementere en PID-regulator på en vanntank, og lage et brukergrensesnitt der man kunne styre denne prosessen. I denne bacheloroppgaven har Martin medansvar for det reguleringstekniske, altså å implementere PID-regulator og finne parametere for denne.

Figur 12: Bilde av Martin.

Figur 13: Bilde av Sacit.

Sacit Ali Senkaya er 21 år og kommer fra Drammen i Viken fylke. Han gikk studiespesialiserende med retning realfag på Akademiet VGS Ypsilon. Etter videregående startet han på elektroingeniør ved NTNU i Trondheim med spesialisering innenfor automatisering og robotikk. Sacit har også erfaring fra Automatiseringsprosjekt-emnet (IELET2104). Han hadde ansvaret for feilhåndtering i prosjektet. Han har også prosjekterfaring fra datateknikk-emnet IELET1002. Der han programmerte et lite ubemannet bakkekjøretøy (Zumo32U4). I bacheloroppgaven er Sacit medansvarlig for det reguleringstekniske med prosjektet.

Sverre Graffer er 21 år og kommer fra Sundvollen i Hole kommune. På videregående gikk han vg1-elektro og vg2/vg3-data og elektronikk med spesiell studiekompetanse. Etter videregående begynte han på elektroingeniør ved NTNU. Fra og med fjerde semester valgte han spesialisering i elektronikk og sensorsystemer. Sverre har erfaring fra en sommerjobb hos Statens Vegvesen, hvor han hjalp til å følge opp byggingen av ny E16 fra Eggemoen til Olum. Her hadde han fokus på det elektriske anlegget. Dette innebar blant annet jording av betongbruer, rør til kabler, jordsammenkoblinger og lyktemaster. I bacheloroppgaven har Sverre hovedansvar for det fysiske systemet og medansvar for brukergrensesnittet.

Figur 14: Bilde av Sverre.

4.2 Utstyr og ressurser

Prosjektgruppa har hatt tilgang til TVC, FTTR og Orbit-lokalet tirsdag, onsdag, torsdag og fredag fra klokka 8 til 12 hver uke. Orbit-lokalet innehar også diverse verktøy som blant annet loddebolt, lupe og 3D-printer som gruppa kunne ta i bruk. Organisasjonen har også en del medlemmer som bidro med hjelp, tips og tilbakemeldinger som gruppa fikk bruk for.

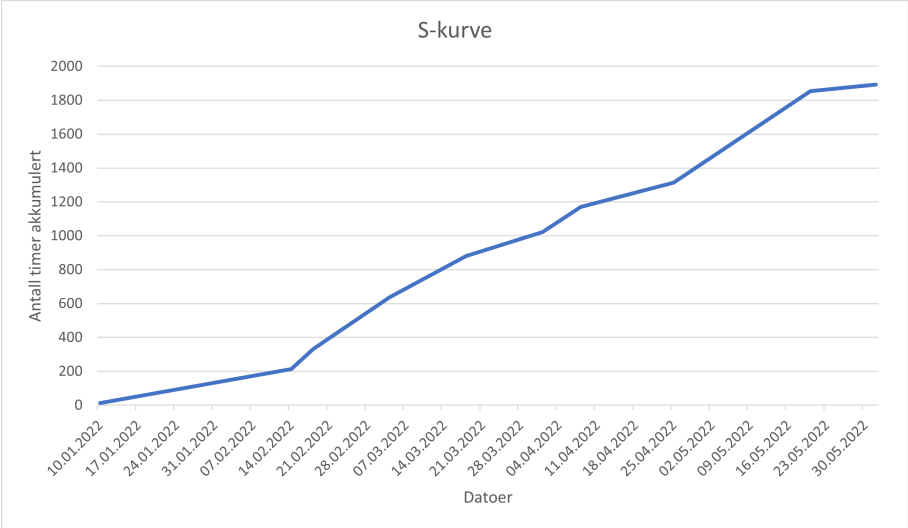
Python, matlab/simulink og KiCAD er tekniske programvarer som gruppa har brukt for å løse denne bacheloroppgaven. Disse programvarene har gruppa tilgang til gjennom NTNU. Av administrative programvarer har gruppa brukt GanttProject til tidsplanlegging, Whimsical for flytskjemaer, Zoom for gjennomføring av digitale møter og Excel for føring av timeregnskap og budsjett.

Orbit NTNU avsatte 8.000 kroner til prosjektgruppa, slik at det gikk an å kjøpe inn diverse komponenter og annet materiell. Gruppa så for seg å kjøpe en del komponenter, men ingenting av det var noe særlig dyrt sett opp mot budsjettet. Omega Verksted var en av utsalgsstedene gruppa særlig tok i bruk. Der er komponentprisene relativt lave og budsjettet holdt dermed med god margin. Bachelorgruppa brukte totalt 2872,14 kroner på oppgaven. Regnskapet er vedlagt rapporten.

4.3 Tids- og kostnadsplan

Figur 15: Gantt-diagram som viser når de ulike arbeidspakkene skal bli utført i prosjektperioden.

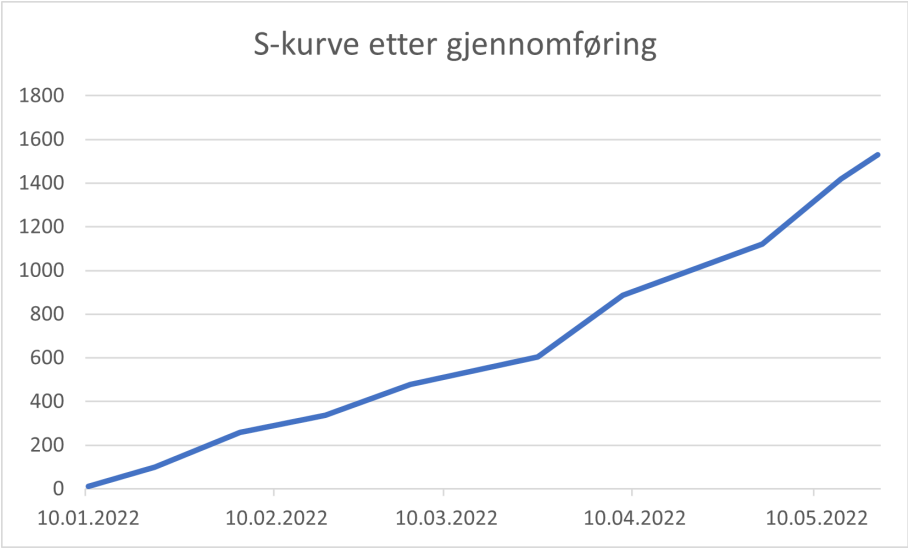
Gantt-diagrammet ovenfor visualiserer den planlagte rekkefølgen og når i prosjektforløpet ulike arbeidspakker blir gjennomført. Diagrammet inneholder også forprosjektet og diverse andre milepæler. De oppskrevne milepælene er levering av forprosjekt, fastsetting av prosjekttittel, levering av bacheloroppgaven og ferdigstilling av sluttpresentasjonen. I timeplanleggingen ble det tatt hensyn til at prosjektgruppemedlemmene hadde en eksamen i Ingeniørfag Systemtenkning (INGT2300), og at det var påskeferie.



Figur 16: S-kurve som viser planlagt akkumulert tidsforbruk i løpet av prosjektperioden.

S-kurven ovenfor visualiserer antall timer akkumulert basert på timeverket oppsatt i arbeidspakkene. Grunnen til at denne typen kurve har fått navnet S-kurve, er fordi kurven starter og slutter ganske flatt og stiger mest i midten av prosjektperioden. Ut fra arbeidspakkene og S-kurven har det blitt planlagt 1893 timer. Det blir cirka 473 timer per person.

I realiteten brukte bachelorgruppa noe mindre tid enn planlagt. Til sammen brukte gruppa 1529,5 timer totalt. Gjennomsnittstiden per medarbeider ble dermed 382,4 timer. Et mer detaljert timeregnskap er lagt ved i appendiks. S-kurven etter gjennomføring ble seende slik ut:



Figur 17: S-kurve som viser akkumulert tidsforbruk i løpet av hele prosjektperioden.

Hovedgrunnen til at det ble mindre timer en planlagt skyldes emnet "Ingeniørfaglig systemtenkning". Gruppa hadde et prosjektarbeid i dette emnet, som tok en del tid i perioden omkring mars.

4.4 Kvalitetssikring

4.4.1 Statusrapportering

For å opprettholde en god kvalitet ved gjennomføring av bacheloropppgaven har gruppa drevet med statusrapportering for å holde intern veileder og oppdragsgiver oppdatert. Dette ble gjort

for at gruppa skulle kunne få eksterne tilbakemeldinger som sørget for at framdriften i prosjektet ble opprettholdt, og at kvaliteten på arbeidet ble som forventet. Statusrapporteringa skjedde via jevnlige møter med veileder og oppdragsgiver der en grundig møteinnkalling ble sendt på forhånd for å informere om hva møtet skal dreie seg om. Møtereferenten hadde i etterkant av disse møtene ansvaret for å skrive et møtereferat som oppsummerte hva som ble diskutert. I tillegg til dette skjedde statusrapporteringa med toukersrapporter. Toukersrapportene var statusrapporter som prosjektgruppa skrev hver andre uke. Disse rapportene inneholdt det som hadde blitt gjort, avvik som oppsto og hva gruppa planla å gjøre neste periode. Disse dokumentene ble lagt ut i PA-mappa gruppa opprettet på Google Disk. Denne mappa hadde gruppemedlemmene, veiledere og oppdragsgiverne tilgang til. Timeregnskap ble også ført opp som en slags statusrapportering. Dette regnskapet ble ført i Excel og sammenlignet med Gantt-diagrammet slik at gruppa hadde oversikt over tidsavviket underveis i prosjektet.

4.4.2 Standardiserte skjemaer

Flere av skjemaene som ble brukt til kvalitetssikring og dokumentasjon var standardiserte. Gruppa brukte malene som ble lagt ut på emnesiden til bacheloroppgaven. Dette er malene:

- Møteinnkalling
- Møtereferat
- Toukersrapport
- Timeregnskap
- Testplan
- Gantt-diagram
- S-kurve

4.4.3 Testplan

Før tester ble gjennomført utarbeidet gruppa testplaner slik at den aktuelle testen som skulle bli gjennomført ble avholdt på en forsvarlig måte. Dette minimerte risikoen for at det skulle oppstå kritiske feil og skader på komponenter, mennesker og omgivelser.

4.5 Risikovurdering

Som følge av at dette prosjektarbeidet var praktisk og at det ble jobbet med maskinvare, var det en viss risiko knyttet til dette arbeidet. Arbeidet som ble utført var elektrorelatert og risikoen dreide seg om strømgjennomgang og potensiell brannfare. Komponentødeleggelse var en annen mindre risiko tilknyttet det elektriske arbeidet. Dette kunne ført til forsinkelse i arbeidet eller småskader på enkeltpersoner. Andre risikoer var tilknyttet oppsettet. Dette kom av at pådragsorganene produserte varme, og potensielt kunne utløse en brann ved utilsiktig bruk.

Det var også noe risiko i forbindelse med bruk av vakuumkammeret. Det var viktig at TVC-testprosedyren ble fulgt nøye slik at utstyret ikke ble ødelagt. Utgassing var en annen risiko som i verste fall kunne ført til forekomst av farlige gasser og røykutvikling. Dette kunne være helseskadelig ettersom vakuumpumpa vil sugd ut gassen fra kammeret og inn i rommet rundt der det befant seg mennesker.

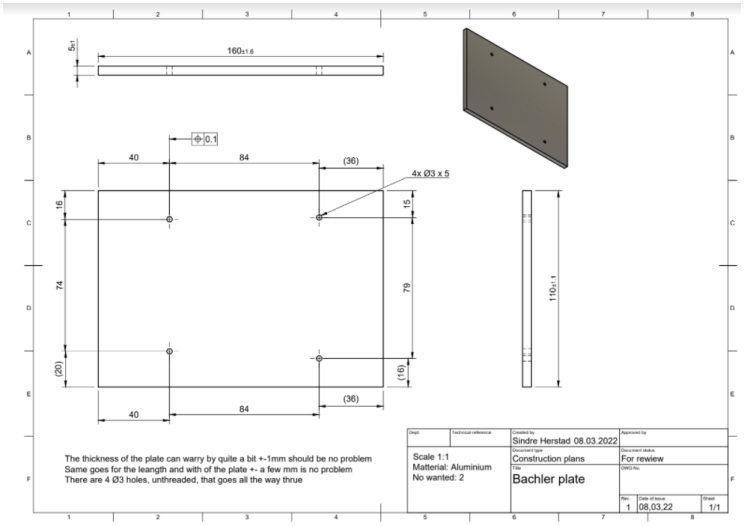
Prosjektgruppa hadde ulike ideer til gjennomføring av prosjektet. En risiko som oppstå her var at ikke alle av disse ideene lot seg gjennomføre. Det var dermed viktig å legge flere planer slik at man alltid hadde noe å falle tilbake på.

En annen risiko tilknyttet gjennomføringen av prosjektet var COVID-19 pandemien. Det kunne blitt nedstegning eller begrensning av arbeidstid/plass hos Orbit NTNU, noe som ville ført til forsinkelse i arbeidet eller at det ble for lite tid til å implementere alt. Faren for at gruppemedlemmene hadde utviklet alvorlig sykdom av COVID-19 var forsvinnende liten, ettersom gruppemedlemmene er unge og tilstrekkelig vaksinert mot viruset. COVID-19 pandemien satte heldigvis ikke noen begrensninger for gjennomføringa av denne bacheloroppgaven.

5 Design

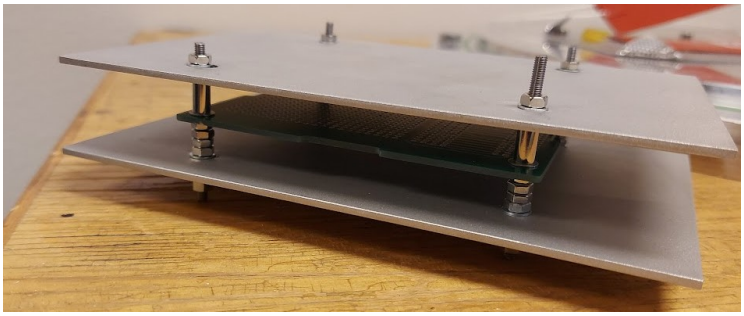
5.1 Fysisk oppsett

Prosjektgruppa tok utgangspunkt i oppsettet til Orbit under designfasen av en ny versjon av det termiske vakuumkammeret. Orbit sitt oppsett besto av to aluminiumsplater med hvert sine pådragsorgan teipet fast til dem. De to platene hadde hver sine fire hull, slik at fire stenger kunne holde dem på plass. Disse stengene fungerte også som festepunkter for komponentene som skulle testes. Prosjektgruppa bestemte seg tidlig for å videreføre dette designet, og startet med å anskaffe to nye plater til å feste de nye pådragsorganene på. De gamle platene var gamle og slitne. Dessuten var FTTR teipet fast til dem og det var ønskelig å arkivere dette. De gamle platene var også litt vaglete og litt klønete å arbeide med. Det ble dermed designet to nye plater med noen dimensjonensendringer slik at de nye pådragsorganene skulle passe bedre. Disse nye platene ble produsert av Sindre Herstad ved mekanisk verksted. Hullene på platene er dessuten plassert etter hullene på testkomponentene slik at disse kan fint festes til de fire stengene.



Figur 18: Figuren viser dimensjonene til de to nye platene som ble produsert. Skissen er tegnet av Sindre Herstad

For å få satt opp denne innretningen må først en av platene bli tredd inn på de fire stengene gjennom hullene. Deretter settes testkomponenten på plass. Platen og testkomponenten adskilles med avstandsstykker. Til slutt tres den siste plata på plass slik som ble gjort tidligere. Denne platen skal også skilles fra testkomponenten med avstandsstykker. Oppsettet vil fungere ved å påtrykke en elektrisk effekt på pådragsorganene som er festet til de to platene. Platene vil da bli varmet opp, og etterhvert gi fra seg nok varmestråling til å varme opp testkomponenten til ønsket temperatur. Testkomponenten vil dermed motta varmestråling fra begge sider, noe som sørger for relativ jevn varmefordeling.



Figur 19: Bildet viser det fysiske oppsettet. Pådragsorganene er ikke tatt med i dette bildet.

5.2 Valg av nytt pådragsorgan

5.2.1 Utvalg

Å finne et nytt pådragsorgan var en stor del av arbeidet som ble utført. Prosjektgruppa hadde mange ulike forslag, men mange ble valgt bort på grunn av spesifikasjonene gitt av Orbit NTNU og av gjennomførbarheten til dem. Det ble lenge vurdert å innføre en varmelampe som pådragsorgan. Dette valget ville kunne simulert varmestrålingen ute i rommet bedre, men ble valgt bort på grunn av sikkerhet. Det var uvisst hvordan varmelampen eventuelt ville reagere hvis det ble satt inn i et vakuumkammer. Det verst tenkelige scenarioet ville være at pæra knuste. Da kunne glassbitene ha blitt sugd inn i pumpene og muligens ødelagt dem. Prosjektgruppa og veileder konkluderte dermed at dette ikke burde testes.

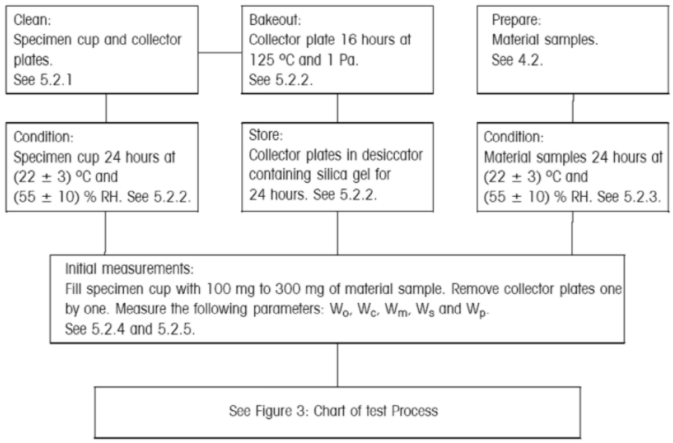
Letinga etter nye pådragsorganer startet dermed på ny. Flere nye forslag ble fremmet, blant annet "heating rods", med til slutt falt valget på varmematter eller "heating mats". Det var mange mulige alternativer til spesifikke modeller av varmematter, men gruppa var interessert i matter som var innenfor 14x14 cm i størrelse. Dessuten var det viktig at mattene kunne oppnå en temperatur på over 120 grader, og at den var dimensjonert til PPS-en. Altså at den maksimalt trengte en spenning på 30 volt og strøm på 5 ampere. Valget falt derfor på RS PRO Silicone Heater. Denne varmematten er 5x15 cm i størrelse, krever 12 volt og har en maks effekt på 60 watt. Dette pådragsorganet kan i følge databladet varmes opp til 180 grader [16].

Figur 20: Bildet viser det nye pådragsorganet prosjektgruppa valgte. (RS Components [16])

Denne varmematte-modellen kom dessuten med varmebestandig lim, slik at den kunne limes direkte på metallplatene. Gruppa slapp dermed å finne en festingsmetode, ettersom den allerede hadde en som var meget brukbar. Databladet til varmematten.

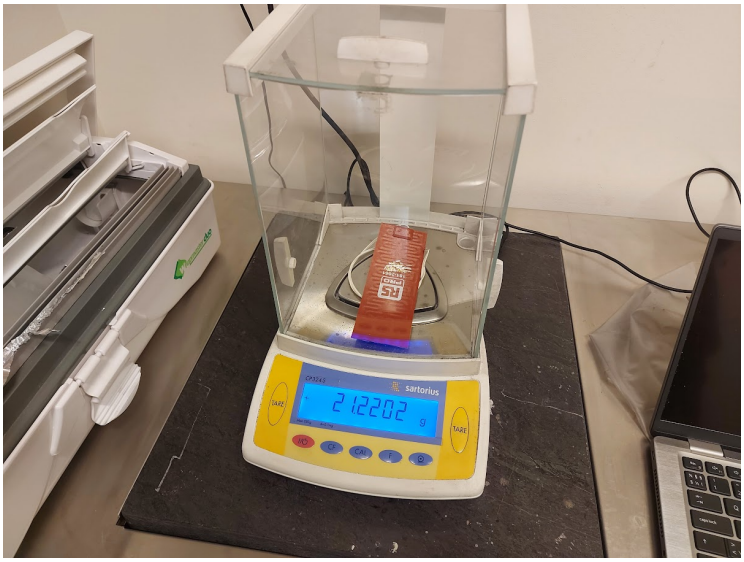
5.2.2 Utgassingstest

For å sørge for at pådragsorganene ikke utgasset for mye var det nødvendig å gjennomføre en utgassingstest. Dette hadde man sluppet å gjøre dersom man visste helt sikkert hva slags type materialer komponenten består av. I vårt tilfelle ble det oppgitt at materialet var av silikon, men det sto ikke hva slags type silikon det var. Noen silikontyper utgasser, mens andre ikke gjør det [17]. Det var derfor nødvending å gjennomføre en utgassingstest. Før selve testen ble gjennomført ble det utarbeidet en testplan. Kort oppsummert tar prosedyren utgangspunkt i figur 21 fra ECSS.



Figur 21: Flytskjemaet viser prosedyren til en utgassingstest. (ECSS-Q-ST-70-02C, s.12)

Testprosedyren starter med rengjøring av testobjekt som i dette tilfellet var en varmematte. Rengjøringen ble gjort med teknisk sprit og en mikrofiberklut. Deretter skal testobjektet veies, og dette ble gjort med en svært nøyaktig vekt som gruppa fikk tilgang til ved institutt for materialteknologi. Da massen til varmematten ble kjent kunne gruppa legge objektet i vakuumkammeret og teste det. Standardprosedyren for vakuumkammeret ble fulgt, og testobjektet ble liggende der i 24 timer slik som det er spesifisert i ECSS. Gruppa og oppdragsgiver bestemte seg også for å kjøre vakuumtesten samtidig som "bakeout"-testen. Ettersom dette ville spare mye tid, og at det ikke var noe poeng å bake et varmeelement i en eller annen ovn for å sjekke at den tålte temperaturen. Gruppa fant en spenningsverdi som holdt temperaturen til varmematten konstant og lot den stå slik.



Figur 22: Bildet viser vekten som gruppa brukte for å finne massen til varmematta.

Etter at varmematta hadde ligget i vakuumkammeret i 24 timer kunne den tas med tilbake til materiallaben for ny veiing. I følge ECSS kan ikke massen endre seg mer enn 1%. Hvis den had-

de gjort det ville massetapet vært et resultat av utgassing. Gruppa fikk følgende resultat etter gjennomføring av utgassingstesten:

Testnummer	Masse før vakuum-test
1	21.2200g
2	21.2204g
3	21.2205g
4	21.2202g
5	21.2200g

$$Gjennomsnitt = \frac{21.2200g + 21.2204g + 21.2205g + 21.2202g + 21.2200g}{5} = 21.22022g$$

Beregning av maksimalt massetap for å bestå utgassingstest:

$$Maksmassetap = 21.22022g \cdot 0.01 = 0.2122022g$$

Testobjekt kan ikke miste mer enn 0.2122022 gram. Hvis den gjør det vil ikke utgassingstesten være godkjent.

Testnummer	Masse etter vakuum-test
1	20.8900g
2	20.8896g
3	20.8901g
4	20.8902g
5	20.8902g

$$Gjennomsnitt = \frac{20.8900g + 20.8896g + 20.8901g + 20.8902g + 20.8902}{5} = 20.89002$$

$$Differanse = 21.22022g - 20.89002g = 0.3302g$$

Massetap fra utgassingstest tilsvarer 1.556 %. Dette er noe høyere enn kravet som ble satt. Dette skyldes antageligvis en rekke feilkilder ved gjennomføring av forsøket. Et problem som oppsto under forsøket var at vannavkjøling til turbopumpa ble skrudd av. Dette førte til at turbopumpa måtte nødstoppe, og trykket dermed steg litt i løpet av de 24 timene. Trykket steg opp til 0.3 mBar. Dette skjedde på kvelden så prosjektgruppa ble ikke varslet før dagen etterpå. På grunn av dette ble det bestemt å forlenge forsøket. Forsøket ble startet kl 13:00 den 30.03.2022 og ble avsluttet 01.04.2022 kl 09:00, og varte dermed i 44 timer. Dette er vesentlig lengre enn planlagt, og kan være noe av årsaken til at den tapte mer masse enn antatt. Det er rimelig og anta at en del av massen som utgasset var fra limet, ettersom varmematten som gjennomgikk en utgassingstest var svært lite klebrig i forhold den som ikke ble utsatt for dette, men dette kan en ikke si med sikkerhet.

Selv om komponenten mistet noe mer masse enn spesifisert i ECSS ble det bestemt sammen med oppdragsgiver at prosjektgruppa burde gå videre med disse pådragsorganene, ettersom tidligere versjoner av TVC utgasset mer en dette.

5.3 Transistorkrets for styring av pådragsorgan

5.3.1 Innledning

I denne bacheloroppgaven er det nødvendig å kunne kontrollere effekten i et resistivt varmeelement. I tillegg ønsket gruppa å fortsette og kunne bruke den eksisterende strømforsyningen, men den nye løsningen for effektstyringen skulle ikke være avhengig av en programmerbar strømforsyning. Noen alternativer for effektstyring ble vurdert: å justere spenningen til varmeelementene ved at regulatoren programmerer hvilken spenning strømforsyningen skal levere, bruke en justerbar lineær regulator mellom pådragsorganet og en fast spenning fra strømforsyning, DC-DC "switch mode" omformer mellom strømforsyning og pådragsorgan og en form for bryter mellom en fast spenning fra strømforsyningen og pådragsorganet. Denne bryteren kan skru av og på pådragsorganet i takt med et PWM-signal. Den første løsningen tilfredsstiller ikke kravet om at systemet skal kunne brukes uten den programmerbare strømforsyningen. Den andre ville ha gitt et stort effekttap i den lineære regulatoren. Dette ville både ha vært sløsing med energi, men ikke minst ville det ha vært en stor utfordring å finne en regulator som tåler effekten. Å kvitte seg med varmen og å unngå overoppheting ville heller ikke ha vært enkelt. Det tredje alternativet kunne ha fungert godt, men det kunne også ha blitt vanskelig å implementere. Valget falt derfor på det siste alternativet.

5.3.2 PWM

PWM (pulsbreddemodulasjon) er en modulasjonsform som brukes for å modulere et firkantpulstog. Firkantpulstoget har en fast frekvens og dermed også fast periodetid. Periodetiden til firkantpulstoget kan deles opp i en periode der signalet er høyt og en hvor det er lavt. Deler man den tiden i en periode hvor signalet er høyt på den totale periodetiden får man arbeidssyklusen [4]. PWM går ut på å styre arbeidssyklusen til firkantpulstoget med et inngangssignal. PWM-frekvensen kan være hvilken som helst, men valget av frekvens er likevel ikke uten betydning. Hvis den er for lav vil "switchingen" bli merkbar i prosessverdien. Er den for høy vil ikke pådragsorganet klare å følge signalet.

5.3.3 Valg av bryterkomponent/ønskede egenskaper

Det første spørsmålet var hva slags bryter som skulle brukes til å slippe gjennom/bryte strømmen i varmeelementet. To åpenbare alternativer var mekaniske brytere (rele og kontaktor) og halvledere. Fordelen med å bruke en mekaniske komponent ville ha vært lavt spenningsfall over bryteren. Siden vi ønsker å bruke PWM blir frekvensen for høy for disse. Derfor er det nok lurere å bruke en halvleder til jobben. Disse er raske og har lang levetid.

Siden gruppa hadde valgt å bruke en halvleder som bryter for å skru av og på lasten, måtte gruppa velge en spesifikk komponent. Det gikk med en del tid til å undersøke ulike komponenter for å se om de kunne fungere i vårt system. Strømmen gjennom komponenten måtte ikke gi for stort spenningsfall over den. Litt spenningsfall kan for noen komponenter løses med en kjøleribbe, men ikke alle har mulighet for montering av en slik. I tillegg er det begrenset hvor høyt vi kan sette spenningen på strømforsyningen for å kompensere for dette spenningsfallet. Komponenten må også kunne holde igjen hele forsyningsspenningen i de periodene der signalet er lavt. I vårt tilfelle er denne spenningen 26,5V. Strømmen som komponenten må tåle når PWM er 100% er ca. 5A. Disse verdiene vil endres dersom man utvider til flere varmematter i fremtiden.

Følgende komponenttyper ble undersøkt: BJT, MOSFET, IGBT, eller kombinasjoner av disse. To slike kombinasjoner gruppa vurderte var Darlington-par og Sziklai par. Begge disse kombinerer to BJT-er for å øke strømforsterkningen [18].

5.3.4 Alternativer

BJT: Dette er en tradisjonell bipolar transistor. Disse fås i varianter som tåler store strømmer. De har tre tilkoblinger kalt kollektor, base og emitter. Strømmen som kan gå fra kollektor til emitter styres av strømmen fra base til emitter. Denne strømforsterkningen kalles ofte h_{FE}, eller β. Effekttransistorer som tåler mye strøm har vanligvis en relativt lav β. Dette kan være en utfordring da basestrømmen blir relativt stor. Spenningen fra basen til emitter er vanligvis cirka et diodefall [19]. Dette er en relativt lav spenning sammenlignet med det strømforsyningen er stilt inn på.

Enten må denne strømmen gå via lasten, men da kan ikke transistoren gå i metning. Hvis den hadde gjort det ville det ikke vært nok spenning til å overkomme diodefallet fra base til emitter. Dermed må man akseptere noe spenningsfall over transistoren og effekttapet som følger med. Darlington og Sziklai hører til i denne kategorien.

Det andre alternativet er å hente basestrøm som ikke går gjennom lasten. Da vil man kunne få transistoren i metning, men spenningsfallet fra strømforsyningen ned til basen vil være ganske stort. Siden basestrømmen også er betydelig, må komponenten mellom forsyningen og basen avgi en god del effekt. Dette er sløseri og i tillegg kan det være vanskelig å finne en komponent som tåler effekttapet og varmen som følger med. Denne komponenten ville typisk ha vært en motstand.

MOSFET: MOSFET ble lenge vurdert som brytertransistor. De har en høy inngangsimpedans og krever derfor lite strøm fra signalet. Dette er en stor fordel sammenlignet med BJT. Den strømmen som likevel kreves går til å lade opp kapasitansen mellom "gate" og "source". Riktignok krever de mer spenning fra "gate" til "source"for å skru seg skikkelig på, enn det BJT trenger fra base til emitter. Fem volt er ikke alltid nok. Mange varianter kan håndtere nok spenning og strøm, men ikke like mange kan gjøre det uten å få et relativt stort effekttap. Når en MOSFET er fullt påslått kan den modelleres som en liten motstand mellom "drain" og "source". Den kalles ofte $R_{DS(on)}$ [20]. På de komponentene gruppa hadde tilgengelig fra Omega Verksted var denne for stor og komponenten ville kunne ha blitt overopphetet når PWM var 100% over tid. En av komponentene fra Omega Verksted som gruppa vurderte var IRF520. Det er ikke er så lett å regne på hvor effektiv kjøleribbene er med isolator og/eller kjølepasta mellom. Gruppa hadde heller ikke tilgang til eksakt datablad for kjøleribbene fra Omega Verksted. Gruppa fant ut at det var mulig å bestille en MOSFET fra RS med veldig lav $R_{DS(on)}$, som ville fungert i kretsen. For eksempel vurderte gruppa å bruke $IRF2804PBF$, men fant i stedet en annen løsning. Derfor ble det aldri noe av dette.

IGBT: Dette kunne ha vært en god brytertransistor for gruppa, siden den kombinerer høy inngangsimpedans med lavt spenningsfall ved høy kollektorstrøm. Utvalget fra Omega Verksted på disse komponentene var dårlig, og det ble ikke undersøkt mer etter at gruppa fant en annen løsning.

Valget falt til slutt på BJT-en 2N3055, fordi den er robust og tåler både strømmen og spenningen med god margin. I tillegg var den i TO-3 pakke som har lav termisk motstand fra overgangen til pakken. For 2N3055 er denne $1,52°C/W$. Det er også lett å montere en kjøleribbe fra Omega verksted med passende størrelse til transistoren. Da vil ikke overoppheting kunne skje. Alt dette i kombinasjon med at komponenten var tilgjengelig på Omega Verksted gjorde at gruppa valgte å teste ut en krets med denne komponenten. Senere ble denne kretsen loddet permanent på et stripboard.

5.3.5 Designprosess

Kretsen skulle ha disse egenskapene: headertilkoblinger til Pi-en, bananpluggtilkoblinger til strømforsyning og varmematter. Signalet fra Pi-en skal skilles elektrisk fra de høyere spenningene fra strømforsyningen med en optoisolator. Da vil for eksempel kortslutning i brytertransistor ikke kunne føre til skade på Pi-en. Siden 2N3055 trenger relativt mye basestrøm ble det bestemt at vi skulle bruke en MOSFET fra kollektor til base slik at basestrømmen tas fra lasten. Gruppa aksepterer dermed at brytertransistoren ikke kan gå i metning. Spenningsfallet trenger likevel ikke å bli problematisk. En småsignaltransistor ville ikke ha tålt å forsyne basen på 2N3055. Derfor ble MOSFET-en IRF520 brukt i stedet. For at MOSFET-en skal skrus skikkelig på er det nødvendig å

ha en høyere spenning enn 5V. Det er ikke sikkert at dette er strengt nødvendig for at kretsen skal fungere, men det kan føre til høyere spenningsfall og mer effekttap i 2N3055. Derfor ble $25V$ fra strømforsyningen koblet til en LM7809, slik at $9V$ er tilgjengelig til dette formålet. 4N35 kan drive MOSFET-en direkte, men vi valgte å introdusere en småsignaltransistor mellom optoisolatoren og MOSFET-en. Dette for å ha mindre belastning på 4N35 i tillegg til at signalet blir invertert en ekstra gang. Inverteringen medfører at et høyt signal fra Pi-en fører til at varmemattene tilføres effekt. Gruppa synes dette er mest ryddig, selv om signalet enkelt kunne ha blitt invertert i koden i Pi-en.

Her er utregninger for komponentverdier i transistorkretsen: Først utregnes resistansen i varmemattene og strømmen ved maks effekt:

$$R_{varmematte} = \frac{U^2}{P} = \frac{12V^2}{60W} = 2,4\Omega$$

Strømmen blir da:

$$I = \frac{U}{R_{varmematte}} = \frac{12V}{2,4\Omega} = 5A$$

Utregningene for basestrømmen og effekten i 2N3055:

$$\beta \; ved \; (I_C = 4A) \; er \; min \; 20, \; I_C = 5A, \; I_B < 250mA$$

$$U_{BE} > 0,7V, \; P > 3,5W$$

Effektutviklingen i IRF520 regnes ut slik:

$$U_{GS} < 8,3V, \; T_C < 175°C, \; I_D < 250mA, \; U_{DS} < 0,3V, \; P \approx 75mW$$

Motstandsverdiene rundt BC547B med kollektorstrøm lik $25mA$ blir slik:

$$I_C > 25mA, \; \frac{9V}{I_C} < 360\Omega, \; R2 = 330\Omega$$

$$\beta_{DC} > 200, \; I_B > 125\mu A, \; I_B = 250\mu A$$

$$R3 < \frac{8,3V}{250\mu A} = 33200\Omega, \; R3 = 20k\Omega$$

For 4N35 blir spenningsfallet i dioden og resistansen til inngangsmotstandene slik:

$$I_C = \frac{9V}{20k\Omega} = 450\mu A, \; I_D > 5mA, \; V_f < 1,1V$$

$$U_{R1} > (3,3V - V_f) = 2,2V, \; R1 < \frac{2,2V}{5mA} = 440\Omega, \; R1 = 220\Omega + 220\Omega$$

Gruppa hadde et ønske om å kunne se på kretsen om/hvordan den fungerte. Lysdioder er godt egnet for dette. Det ble valgt å ha en rød lysdiode som lyser når det står spenning på varmemattene, og en grønn som lyser når det ikke gjør det. Siden PWM- frekvensen er ganske høy vil det ikke være mulig å se at lysdiodene blinker selv om de gjør det [21]. Den oppfattede lysstyrken til den røde lysdioden vil være omtrent PWM-andelen, og for den grønne vil det være $100\% - PWM - andelen$. Dermed vil den ene lyse sterkere når den andre lyser svakere og omvendt. Dette gjør det mulig å se på kretsen og gjøre et grovt anslag på hva PWM-andelen er. Dette lysdiodeparet er koblet til kollektoren på 2N3055. Dette gjør at signalet fra Pi-en må komme gjennom hele kretsen vellykket. Dette betyr at diodene ikke vil lyse som forventet med mindre hele kretsen fungerer som den skal. Dette kan gjøre potensiell feilsøking i fremtiden enklere.

Det var ønskelig at begge diodene skulle lyse like sterkt ved 50% arbeisdssyklus. Det ble funnet at de lyste like sterkt når den røde har en strøm på 8,5mA, og den grønne har 0,7mA. Strømmene når diodene er på vil bli satt til disse verdiene. Her er utregninger for disse lysdiodene og kretsene rundt:

$$Rød \; LED: \frac{9V - 2V - 0,7V - 1V}{8,5mA} = 741\Omega, R5 = 270\Omega + 470\Omega$$

$$Grønn\ LED: \frac{25V - 2,2V}{0,7mA} = 32,57k\Omega, R6 = 68k\Omega||68k\Omega$$

Det ble bestemt at det også var ønskelig å ha et identisk lysdiodepar som oppfører seg på samme måte det andre ved PWM-inngangen til transistorkretsen. Dette for å kunne se om signalet i det hele tatt blir generert/kommer seg frem til kortet med transistorkretsen. Dette for å gjøre feilsøking enda litt enklere. Hvis de to lysdiodeparene oppfører seg likt er det grunn til å tro at det ikke er noe galt på selve transistorkretsen. Hvis de ikke lyser som forventet, men likt kan man anta at det er noe galt med signalet som kommer frem til kretsen. Dersom inngangsdiodeparet lyser som forventet, men utgangsdiodeparet ikke gjør det er det sannsynlig at det noe er feil med selve transistorkretsen eller tilkoblingene til den. En normalsituasjon hvor dette vil skje er når Pi-en er påslått og tilkoblet transistorkretsen, men strømforsyningen og/eller varmemattene ikke er det. Da vil utgangsparet ikke lyse som normalt. Hvis begge parene lyser som forventet og likt er det grunn til å tro at alt frem til varmemattene fungerer som normalt. Hvis det likevel ikke kommer varme i mattene kan det være at det er noe galt med dem eller tilkoblingen til dem. Disse lysdiodekretsene har ingen funksjon utover å gi visuell informasjon til brukereren.

Her er utregninger for denne delen av kretsen:

$$Rød\ LED: \frac{9V - 2V}{8,5mA} = 824\Omega, R7 = 330\Omega + 510\Omega$$

$$Grønn\ LED: \frac{9V - 2,2V}{0,7mA} = 9714\Omega, R8 = 10k\Omega$$

For transistoren:

$$I_C < 20mA, R_B < \frac{9V - 0,7V}{20mA/200} = 83k\Omega, R9 = 51k\Omega$$

I ettertid fant gruppa ut at det var dumt å introdusere spenning fra LM7809 på venstre side av optoisolatoren. Dette ville ha gjort at denne delvis ikke lenger har noen funksjon. Derfor ble det prøvd å bruke $5V$ fra Pi-en i stedet til denne delen av kretsen. Dette fungerte brukbart selv om utregningene tar utgangspunkt i at $9V$ er brukt. Konsekvensen er at lydiodene lyser litt svakere enn tenkt. Den endelige kretstegningen ligger i appendiks C.

5.3.6 Utvidelse fra to til fire varmematter

Oppdragsgiver har luftet muligheten for å bruke fire varmematter i stedet for to. Dette er for å kunne varme opp større testobjekter. Gruppa ble spurt om transistorkretsen og resten av systemet kunne fungere sammen med flere matter.

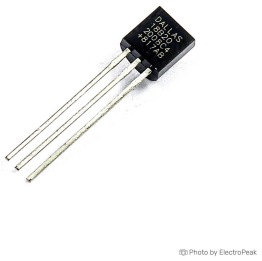
Først må de to nye varmemattene kobles i serie. Deretter kan de enten kobles i serie eller i parallell med de eksisterende varmemattene. Hvis de havner i serie må spenningen fra strømforsyningen økes til cirka det dobbelte, men strømmen gjennom transistoren vil være uendret. V_{CEO} er oppgitt til 60V (2N3055), så dette burde gå fint selv om det nærmer seg grensen. Det vil være omvendt hvis de havner i parallell. Da vil transistoren måtte tåle dobbelt så mye strøm og mer enn dobbelt effekttap. Dette fordi spenningsfallet over transistoren vil øke noe med dobbelt så stor strøm. Spenningen derimot vil være den samme. I dette tilfellet må det også sjekkes om IRF520 kan gi nok basestrøm til denne nye kollektorstrømmen. Å koble de nye mattene i parallell vil sannsynligvis også gå fint, siden 2N3055 har ($I_C(max) = 15A$). I tillegg kjennes den ikke særlig varm ut under drift med to matter, slik det er i dag.

5.4 Sensorsystem

For å etablere automatisk styring er det essensielt å kunne måle den respektive prosessverdien. Data fra sensor sendes direkte inn i en regulator som bestemmer pådraget varmemattene blir tilført. Sensorer bidrar også med nyttig informasjon om tilstanden til systemet. I denne bacheloroppgaven er det valgt å bruke to sensorsystemer; et bestående av fem digitale sensorer og et med to analoge sensorer.

5.4.1 Digitale sensorer

Det digitale sensorsystemet består av fem digitale temperatursensorer av typen DS18B20 fra DALLAS. Disse sensorene plasseres inne i vakuumkammeret på diverse steder for å avlese temperaturer. En av disse sensorene klypes fast til testkomponenten, og brukes til å regulere etter. De fem digitale temperatursensorene er koblet sammen i en 1-leder buss, som er en seriell kommunikasjonsprotokoll. Denne brukes for å kunne overføre temperaturavlesninger til Raspberry Pi gjennom kun en ledning. Sensorene benytter seg av 64-bit ROM, som lagrer den serielle koden til sensorene. Den 64-bit adressen tillater Raspberry Pi-en å kunne motta temperaturavlesninger fra teoretisk sett uendelig antall sensorer over samme inngang. Adressen er unik for de ulike sensorene, slik at Raspberry Pi-en kan forstå hvilken sensor de ulike temperaturavlesningene kommer fra.



Figur 23: Bildet viser hvordan en DS18B20-sensor kan se ut. (ElectroPeak [[22]])

Som figur 25 viser er de digitale sensorene koblet med en enkelt ledning for spenning, en for jord og en for signal. Henholdsvis rød, svart og blå. Sensorene blir koblet i serie. Det er koblet en 10 kOhm motstand mellom spenning og signal. Ledning for spenning går inn på 3.3 V på Raspberry Pi og signalledningen er koblet til en av GPIO-portene.

For å få til temperaturavlesing fra DS18B20, må først "One-wire"-grensesnittet settes opp på Raspberry Pi. Dette kan gjøres ved å følge brukermanualen som ligger vedlagt bacheloroppgaven.

DS18B20-sensorene kan kun måle temperaturer mellom -55 til 125°C. Så det var derfor nødvendig å innføre to ekstra sensorer som kan måle høyere temperaturer for å se på tilstanden til pådragsorganene som vil bli noe varmere.

5.4.2 Analogt grensesnitt

Siden pådragsorganene kan oppnå en temperatur over 125°C ble det sett på muligheten for å bruke analoge sensorer av typen LM35 fra Texas Instruments. LM35 er en presisjons-celcius temperatursensor. Den har en lineær skaleringsfaktor på 10mV/°C. Da blir omregningen fra spenning til temperatur:

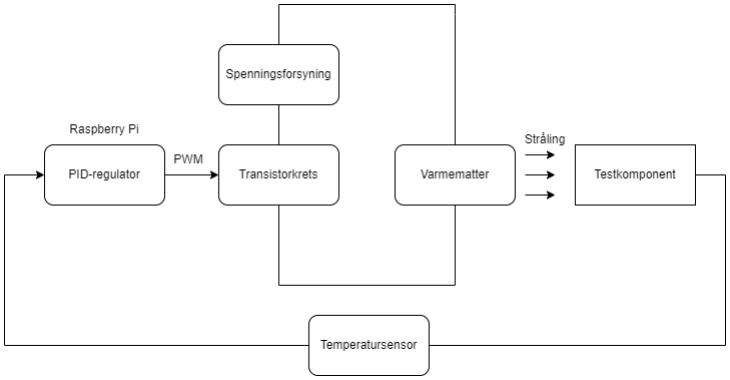
$$T = \frac{V_{out}}{\frac{10}{1000}}$$

For å kunne benytte seg av analoge komponenter må de analoge signalene omgjøres til digitale signaler, som Raspberry Pi-en kan tyde. Dette gjøres med en ADC. I dette prosjektet ble det benyttet en ADC av typen ADS1115 fra Adafruit og Texas Instruments, som enkelt kommuniserer med Raspberry Pi gjennom I^2C kommunikasjonsbuss. ADS1115 har høy presisjon med 16-bit og fire kanaler. De analoge komponentene er som figur 25 viser koblet til hver sin analoge inngang på ADS1115. SDA er koblet til SDA-pinnen på Raspberry Pi (GPIO 2) og SCL er koblet til SCL-pinnen (GPIO 3). ADS1115 er koblet til jord og opererer med logisk nivå på 3.3 V.

For å kunne benytte seg av ADS1115, må først I^2C settes opp på Raspberry Pi. Brukermanualen vedlagt oppgaven går gjennom dette.

5.5 Oppkobling av overordnet system

Gruppas plan for å implementere de nye pådragsorganene innebærer å koble dem i serie, og forsyne dem med en fast spenning fra PPS-en. Effekten fra PPS-en tilpasses av transistorkretsen som mottar et PWM-signal fra regulatoren. Et pådragsorgan er dimensjonert for en spenning på 12 volt og en effekt på 60 watt. Når to matter kobles i serie må man derimot sette spenningen til 24 volt slik at det vil være et spenningsfall på 12 volt på hver av pådragsorganene. Den resulterende totale effekten blir 120 watt. I realiteten ble spenninga satt til 26,5 volt, ettersom det var noe spenningsfall over transistorkretsen.

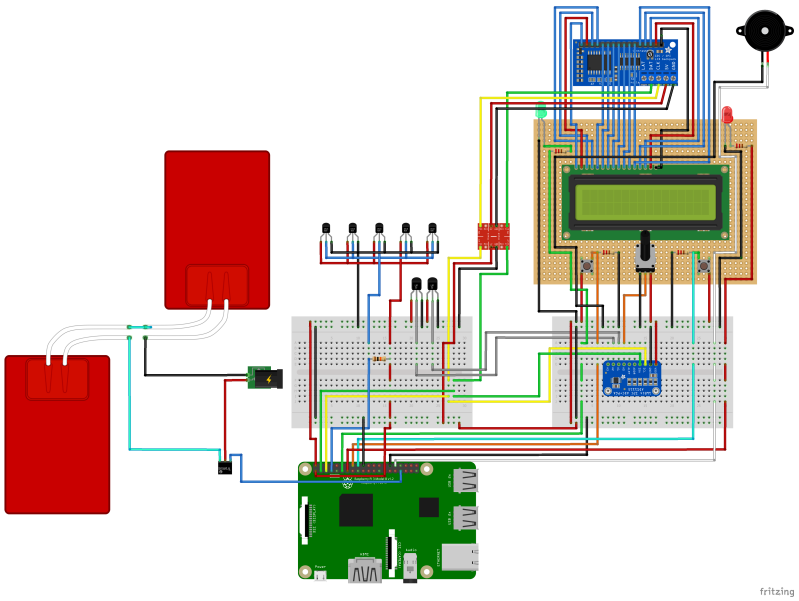


Figur 24: Bildet viser hvordan komponentene implementeres i et blokksjema.

Det fysiske brukergrensesnittet kobles også opp til Rasperry Pi-en som skal kjøre regulatoralgoritmen. Dette brukergrensesnittet skal bestå av en LCD for framvisning av verdier, to knapper og et potensiometer slik at operatør kan endre modus, settpunkt og manuelt pådrag.

Det grafiske brukergrensesnittet som skal inneholde sanntidsplotting og mulighet til å endre på regulatorparametere kjøres direkte fra samme Raspberry Pi. En trenger dermed bare å trekke en HDMI-kabel fra Pi-en til en dataskjerm og koble til mus og tastatur for å få fram dette.

Det endelige koblingsskjemaet med alle komponenter koblet sammen til Raspberry Pi-en ble seende slik ut:

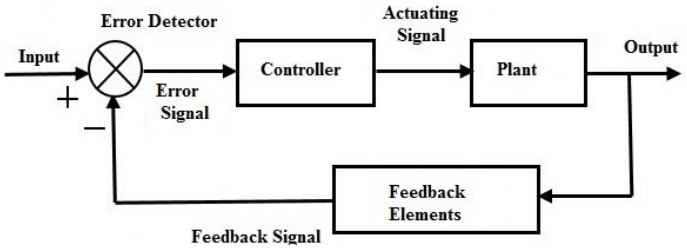


Figur 25: Overordnet koblingsskjema av systemet.

6 Teori

6.1 Reguleringsteknikk

Reguleringsteknikk er en sentral del av denne oppgaven, ettersom teorien fra dette fagområdet er essensielt for å implementere automatisk styring. Varmeelementene som varmer opp testkomponenten til ønsket temperatur, og holder temperaturen der trenger et riktig styringssignal (pådragssignal) for å få til dette. Verdien til dette signalet blir beregnet av en regulator som sammenligner ønsket temperatur med faktisk temperatur. Et slikt oppsett blir ofte kalt for tilbakekobling, og illustreres av en sløyfe.



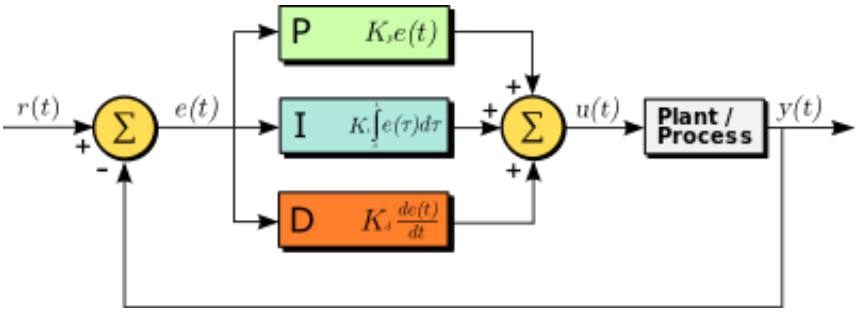
Figur 26: Figuren viser en generisk reguleringssløyfe.(Elprocus [23])

6.1.1 Av/på regulator

Den enkleste formen for regulering er av/på regulering. Termostat er et eksempel på en slik regulator. Pådragsorganet vil bli skrudd på dersom den målte temperaturen er lavere enn den ønskede. Dersom den målte temperaturen er høyere enn den ønskede temperaturen vil pådragsorganet bli skrudd av. Slik vil regulatoren holde på helt til den blir skrudd av. Den resulterende prosessverdien av en slik regulering vil være svært svingende/oscillerende. Dette er ikke ønsket, og en slik regulator vil dermed ikke bli implementert.

6.1.2 PID-regulator

En PID-regulator er en mer avansert regulatortype i forhold til av/på-regulator. Denne regulatortypen består av tre ledd som påvirker pådragsverdien: proporsjonalvirkning, integralvirkning og derivatvirkning, derav navnet PID. Inngangsignalet til PID-regulatoren mottar avviket mellom ønsket- og målt temperatur. Basert på dette avviket beregnes det en pådragsverdi ut ifra vektleggingа av de tre ulike parameterne. Denne pådragsverdien sendes så til pådragsorganet, som igjen påvirker prosessen. Hvis en av PID-parameterene settes til null, vil dette leddet ikke ha noe effekt lenger. Man kan dermed ende opp med flere alternative regulatorer: P-regulator, PI-regulator og PD-regulator.



Figur 27: Figuren viser en PID-regulator implementert i en generisk reguleringssløyfe. (Arturo Urquizo [24])

P-leddet i PID-regulatoren sørger for at pådraget endres proporsjonalt med avviket. Desto større avviket er, desto høyere pådragsverdi vil dette leddet spytte ut. Hvis målt temperatur blir høyere enn ønsket temperatur vil pådragsverdien være negativ. I motsetning til en av/på regulator vil P-leddet kunne forårsake en konstant målt prosessverdi. Selv om den målte prosessverdien er konstant kan det allikevel være et avvik mellom den og ønsket temperatur. Dette avviket kalles for det stasjonære avviket. Den matematiske formen til avviket og P-leddet er slik:

$$e(t) = r(t) - y(t)$$

e = avviket mellom ønsket- og målt temperatur
r = ønsket temperatur
y = målt temperatur

$$u_p(t) = K_p \cdot e(t)$$

u_p = pådragsverdi fra P-leddet
K_p = proporsjonalforsterkning, altså parameteren for P-leddet

I-leddet i PID-regulatoren integrerer tidligere avvik over tid og legger dette til pådragssignalet. Hensikten med dette leddet er å eliminere det stasjonære avviket. Dette skjer ved at så lenge det er avvik vil I-leddet øke eller minke. Hvis avviket er lik null vil I-leddet holde seg konstant. Parameteren for I-leddet er T_i eller integraltid. Denne parameteren sier noe om hvor raskt integrasjonen skal skje. Den matematiske formen til I-leddet er slik:

$$u_i(t) = \frac{K_p}{T_i} \cdot \int_0^t e(t)\, dt$$

u_i = pådragsverdi fra I-leddet
K_p = proporsjonalforsterkning, altså parameteren for P-leddet
T_i = integraltid, altså parameteren for I-leddet
t = tid

D-leddet i PID-regulatoren ser på derivatet til avviket, altså endringsraten. Målet med D-leddet er å motvirke endring av selve prosessverdien slik at prosessen skal holde seg mest mulig rolig. Når prosessverdien er stasjonær vil man dermed ikke få noe pådrag fra D-leddet. Den matematiske formen til D-leddet er slik:

$$u_d(t) = K_p \cdot T_d \cdot \frac{de(t)}{dt}$$

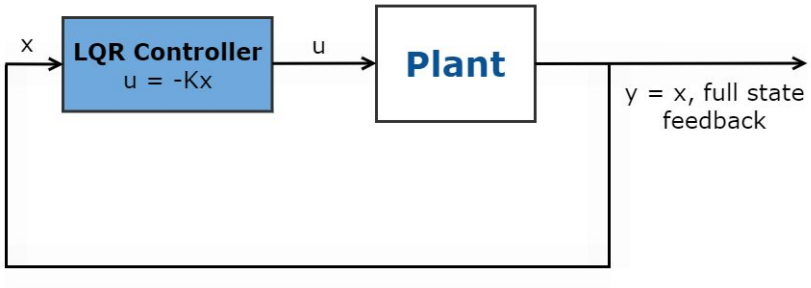
u_i = pådragsverdi fra D-leddet
K_p = proporsjonalforsterkning, altså parameteren for P-leddet
T_d = derivattid, altså parameteren for D-leddet

Det totale pådraget fra PID-regulatoren er summen av de tre leddene. Matematisk uttrykkes dette slik:

$$u(t) = u_p(t) + u_i(t) + u_d(t)$$

Denne type PID-regulator er på det som kalles sumform. Dett finnes andre måter å sette opp PID-regulatorer på, men prosjektgruppa syntes denne er den mest intuitive.

6.1.3 LQR



Figur 28: Optimal kontroll for et dynamisk system.(MathWorks [25])

Sammenlignet med en modellfri regulator som PID, krever LQR en linearisert tilstandsrommodell av et dynamisk system for å beregne en optimal forsterkningsmatrise K. For at en LQR-løsning skal være optimal må totalverdien av kostfunksjonen i ligning (1) være minimal [26]. I form av optimal kontrollalgoritme minimere LQR kostfunksjonen basert på vektingsparametere bestemt av ingeniøren. Konseptet om LQR er direkte knyttet til optimal kontrollteori som handler om å regulere et dynamisk system ved minimum kost [27].

$$J(u) = \int_0^\infty (x^T(t)Qx(t) + u^T(t)Ru(t))dt \tag{1}$$

Her representerer matrisene Q og R vektingen for tilstandsvektoren x(t) og pådragsvektoren u(t) [26]. Ved å justere på Q- og R-matrisene kan hver enkel tilstand og pådrag i et system prioriteres ulikt, slik at totalverdien av kostfunksjonen justeres etter ønsket måling. For et dynamisk temperatursystem vil det være naturlig å optimalisere tilstanden for prosesstemperaturen og effekten ut av pådragsorganet. For en stabil og rask temperaturtransient må verdien på Q-matrisen økes. Å minimere energiforbruket fra pådragsorganet og samtidig oppnå ønsket temperatur uten å tenke på tidsrespons, må verdien av R-matrisene økes. Ingeniøren må sammenligne resultater ved å justere på matrisene for å oppnå ønsket oppførsel.

Etter tuning brukes Q- og R-matrisene til å løse "Algebraic Riccati Equation (ARE)" for å beregne tilbakekoblingsmatrisen S [26].

$$A^T S + SA - SBR^{-1}B^T S + Q = 0 \tag{2}$$

S-matrisen fra "ARE" brukes videre til å beregne en optimal forsterkningsmatrise K.

$$K = R^{-1}B^T S \tag{3}$$

For et optimalisert system brukes forsterkningsmatrise K i kontrolloven under:

$$u = -Kx \tag{4}$$

Fra et praktisk perspektiv er det tidkrevende å løse ligningene over for hånd. En optimal forsterkingsmatrise K kan løses ved å ta i bruk en innebygd funksjon kalt *lqr* i Matlab.

$$K, S, E = lqr(A, B, Q, R) \tag{5}$$

6.2 Termodynamikk

For å kunne utlede en matematisk modell av det termiske vakuumkammeret må man i dette tilfellet ha kunnskap om selve varmeoverføringen. Termodynamikk står dermed sentralt her. Mesteparten av teorien tilknyttet termodynamikk er hentet fra boken "Heat transfer: A Practical Approach" som er skrevet av Yunus A. Cengel.

Termodynamikk er definert som den delen av fysikken som tar for seg sammenhengene mellom energi, varme og arbeid [28]. Et synonym for termodynamikk er varmelære. Termodynamikken baserer seg hovedsaklig på to lover:

- 1. Energi kan ikke oppstå eller forsvinne, det kan kun gå over i andre former.
- 2. Varme kan strømme av seg selv fra et sted med høy temperatur til et sted med lavere temperatur. Dette skjer aldri motsatt vei.

I denne sammenheng er varme definert som den formen for energi som transporteres fra et sted til en annet som et resultat av en temperaturdifferanse. Det vil aldri foregå noe varmeoverføring hvis to objekter har lik temperatur. Hvis en temperaturdifferanse er reel kan varmeoverføringen skje på tre ulike måter; konduksjon, konveksjon og stråling. Disse tre varmeoverføringsmetodene forklares senere i oppgaven. Et annet viktig konsept innenfor termodynamikken er varmekapasitet. Varmekapasitet er et mål på hvor mye varme (energi) som må tilføres et stoff for å heve temperaturen i stoffet [29].

Formelen nedenfor viser sammenhengen mellom varmeoverføring og varmekapasitans:

$$\dot{Q} = C \cdot \frac{dT}{dt} \tag{6}$$

$$C = c \cdot m$$

Parameter	Enhet
$\dot{Q}$ = raten til netto varmeoverføring	$W = J/s$
m = massen til objekt	kg
c = spesifikk varmekapasitet	$J/K \cdot kg$
dT/dt = temperaturendring	K/s

6.2.1 Energibalanse

Når en skal begynne å modellere et termisk system, er det vanlig å sette opp en energibalanse. Energibalansen er en likning som tar hensyn til alle varmestrømmene som går inn og ut av systemet. Likninga baserer seg på termodynamikkens 1. lov.

Energibalansen til et generisk system er satt opp nedenfor:

$$Q_{inn} - Q_{ut} = \Delta Q_{system}$$

6.2.2 Konduksjon

Konduksjon er en av måtene varmeoverføring kan foregå på. Denne formen for varmeoverføring foregår mellom to objekter som berører hverandre. Dette skyldes kollisjon og vibrasjon av molekyler, og diffusjon av frie elektroner. Objektet med høyest temperatur har molekyler som innehar mer kinetisk energi, og disse vil herje med molekylene til det kalde objektet slik at de til slutt vil være

like. Konduksjon gjelder hovedsaklig for faste stoffer, og er årsaken til at platene på kjøkkenet varmer opp grytene. Varmeoverføringsraten via konduksjon avhenger av geometrien mellom objekter (tversnittsarealet), tykkelse, materialtype og temperaturdifferanse. Materialtypen har noe å si for den termiske konduktiviteten. Den termiske konduktiviteten er et mål på hvor godt et stoff leder varme.

Formelen for konduksjon er vist nedenfor:

$$\dot{Q} = k \cdot A \cdot \frac{T_{varm} - T_{kald}}{d}$$

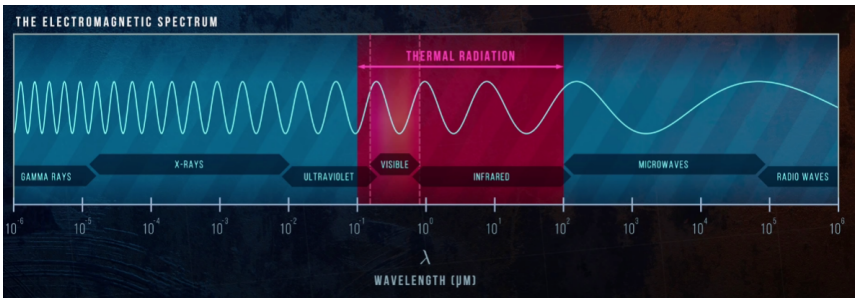
Parameter	Enhet
$\dot{Q}$ = raten til netto varmeoverføring	$W = J/s$
k = termisk konduktivitet	$W/(m \cdot K)$
A = tversnittsareal	m^2
T_{varm} = temperatur til varmtobjekt	K
T_{kald} = temperatur til kaldtobjekt	K
d = tykkelse	m

6.2.3 Konveksjon

Konveksjon er den andre måten varmeoverføring kan skje på. Akkurat som konduksjon, baserer konveksjon seg på varmeoverføring via fysiske medier. Hovedforskjellen mellom konduksjon og konveksjon er at konduksjon foregår mellom faste stoffer, mens konveksjon skyldes et fluid (væske eller gass) i bevegelse. I denne bacheloroppgaven er varmeoverføring via konveksjon ikke relevant. All varmeoverføring foregår jo i et vakuumkammer, der befinner det seg ikke noe fluid som kan transportere varme. Det er dermed ikke noe poeng å se videre på dette fenomenet i denne bacheloroppgaven.

6.2.4 Varmestråling

Den siste formen for varmeoverføring er stråling. Imotsetning til konduksjon og konveksjon kan varmestråling forplante seg i vakuum, og er dermed helt sentral for denne bacheloroppgaven. Både faste stoffer og fluider kan radiere varme. Når et stoff radierer varme omhandler dette elektromagnetiske bølger som sprer seg med lysets hastighet. Man karakteriserer elektromagnetisk stråling utifra bølgelengde. Termisk stråling innehar bølgelengder som strekker seg fra deler av det ultrafiolettet spekteret, helt til det infrarøde.



Figur 29: Figuren viser den delen av det elektromagnetiske spekteret varmestråling opptrer i. (The Efficient Engineer, 2021, 1:49 [30])

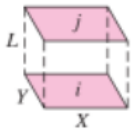
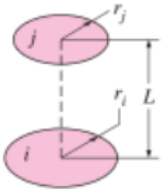
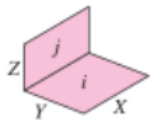
Når en jobber med varmestråling er det nyttig å definere en perfekt emmiter, altså et stoff som radierer makismalt gitt en bestemt temperatur. Denne perfekte emitteren omtales ofte som et svart legeme. Et slikt stoff finnes ikke ute i den virkelig verden, men når det kommer til fysikken er den allikevel nyttig å ta i bruk. Stefan-Boltzmann's lov gir sammenhengen mellom netto varmeoverføring av varmestråling:

$$\dot{Q} = A\sigma\epsilon T^4$$

Parameter	Enhet
$\dot{Q}$ = raten til netto varmeoverføring	W
A = areal	m^2
σ = Stefan-Boltzmanns konstant	W $m^{-2}K^{-4}$
T = temperatur	K
ϵ = emisitivitet [0,1]	enhetsløs

Emisiviteten som inngår i formelen ovenfor er en overflateegenskap. Verdien til den egenskapen ligger i intervallet mellom 0 og 1. Hvis emmisiviteten er lik 1 så har man et svart legeme, og hvis den er 0 så har man en perfekt reflektor.

Et annet viktig konsept ved varmeoverføring via stråling er "view factor". Når det er varmeveksling mellom to objekter på grunn av stråling skyldes ikke dette bare strålingsegenskapene til objektene, men også deres relative orientering og plassering. "View factor" er parameteren som innføres for å ta hensyn til dette. Denne parameteren er kun geometrisk. Parameteren oppgis slik: F_{ij}. "View factor" i dette tilfelle er fra objekt i til j, og er delen av strålingen som radieres fra i som treffer j. Det finnes ulike måter å beregne denne parameteren på basert på oppstillingen man er interessert i. Nedenfor er det en tabell som inneholder tre eksempler på hvordan man beregner "view factor" for tre ulike oppsett.

Geometry	**Relation**
Aligned parallel rectangles 	$\overline{X} = X/L$ and $\overline{Y} = Y/L$ $F_{i\to j} = \frac{2}{\pi \overline{X}\overline{Y}} \left\{ \ln \left[\frac{(1+\overline{X}^2)(1+\overline{Y}^2)}{1+\overline{X}^2+\overline{Y}^2} \right]^{1/2} + \overline{X}(1+\overline{Y}^2)^{1/2} \tan^{-1} \frac{\overline{X}}{(1+\overline{Y}^2)^{1/2}} + \overline{Y}(1+\overline{X}^2)^{1/2} \tan^{-1} \frac{\overline{Y}}{(1+\overline{X}^2)^{1/2}} - \overline{X} \tan^{-1} \overline{X} - \overline{Y} \tan^{-1} \overline{Y} \right\}$
Coaxial parallel disks 	$R_i = r_i/L$ and $R_j = r_j/L$ $S = 1 + \frac{1+R_j^2}{R_i^2}$ $F_{i\to j} = \frac{1}{2} \left\{ S - \left[S^2 - 4 \left(\frac{r_j}{r_i} \right)^2 \right]^{1/2} \right\}$
Perpendicular rectangles with a common edge 	$H = Z/X$ and $W = Y/X$ $F_{i\to j} = \frac{1}{\pi W} \left(W \tan^{-1} \frac{1}{W} + H \tan^{-1} \frac{1}{H} - (H^2+W^2)^{1/2} \tan^{-1} \frac{1}{(H^2+W^2)^{1/2}} + \frac{1}{4} \ln \left\{ \frac{(1+W^2)(1+H^2)}{1+W^2+H^2} \times \left[\frac{W^2(1+W^2+H^2)}{(1+W^2)(W^2+H^2)} \right]^{W^2} \times \left[\frac{H^2(1+H^2+W^2)}{(1+H^2)(H^2+W^2)} \right]^{H^2} \right\} \right)$

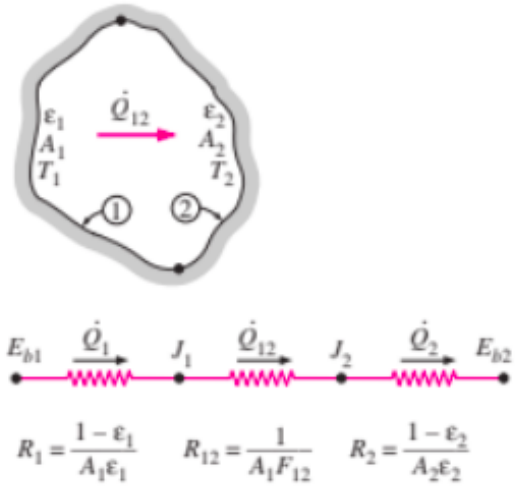
Figur 30: Figuren viser hvordan man beregner view factor parameteren for tre ulike oppsett. (Cengel, 2002, s. 609 [31])

Orienteringen og plasseringen til det fysiske oppsettet kan sammenlignes med det første oppsettet i figur 30. Dermed kan "View factor" for det fysiske oppsettet beregnes med hjelp av første relasjon i figuren. Antagelsen innebærer også at "View factor" kun beregnes i området hvor overflate 1 og overflate 2 overlapper hverandre. Ifølge utregningen i appendiks vil "View factor" av det fysiske oppsettet være $F_{ij} = 0.74$.

Netto varmeoverføringsrate mellom to grå legemer med forskjellige emissivitet, temperatur og areal i en innkapsling kan utledes slik

$$\dot{Q}_{12} = \dot{Q}_1 = -\dot{Q}_2$$

Her sier ligningen at netto varmeoverføringsrate fra overflate 1 til 2 må være lik netto varmeoverføringsrate fra overflate 1 og netto varmeoverføringsrate til overflate 2.



Figur 31: Figuren viser et strålingsnettverk mellom to objekter i en innkapsling. (Cengel, 2002, s. 627 [31])

I figuren 31 representerer R_1 og R_2 overflateresistansene i et strålingsnettverk, mens R_{12} representerer "space"-resistans. Strømmen gjennom resistansene i en elektrisk krets kan bestemmes ved å dividere differansen av potensialet mellom punkt a og b på totalresistansen mellom de samme punktene. Ved å ta utgangspunkt i teorien over kan netto varmeoverføringsrate i et strålingsnettverk utledes slik:

$$\dot{Q}_{12} = \frac{E_{b1} - E_{b2}}{R_1 + R_{12} + R_2} = \dot{Q}_1 = -\dot{Q}_2$$

eller

$$\dot{Q}_{12} = \frac{\sigma(T_1^4 - T_2^4)}{\frac{1-\epsilon_1}{A_1 \cdot \epsilon_1} + \frac{1}{A_1 \cdot F_{12}} + \frac{1-\epsilon_2}{A_2 \cdot \epsilon_2}}$$

Parameter	Enhet
σ = Stefan-Boltzmanns konstant	$W/m^2 \cdot K^4$
T_1	K
T_2	K
Emissivitet av overflate 1	enhetsløs
Emissivitet av overflate 2	enhetsløs
Overflateareal av overflate 1	m^2
Overflateareal av overflate 2	m^2
F_{12} = "View factor" mellom oveflate 1 og 2	enhetsløs

6.2.5 Joule heating

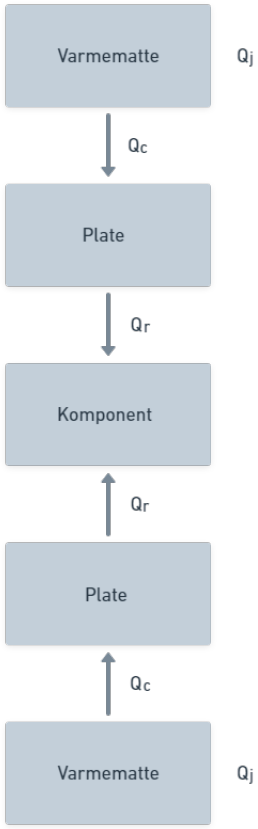
"Joule heating" er en måte å produsere varme på [32]. Denne varmeproduksjonen skjer via at man påtrykker en resistiv last en elektrisk effekt. Dette er relevant for oppgaven ettersom varmematten er å anse som en ren resistiv last. Formelen for elektrisk effekt er gjengitt under:

$$P = UI = I^2R = U^2/R$$

Parameter	Enhet
P = effekt	W
U = spenning	V
I = strøm	A
R = resistans	Ω

6.3 Modellering

Målet med å utlede en fysikkbasert modell er å kunne bruke denne informasjonen til å implementere en modellbasert-regulering. Rene fysikkbaserte modeller stemmer som regel ikke helt overens med det virkelige systemet. Det er dermed viktig å sammenligne hvor godt den faktisk stemmer med virkelig data hentet fra den faktiske prosessen. For denne modelleringen antas det at varme som blir overført i løpet av tastetiden er jevn. Varmeoverføringen i systemet skjer hovedsakelig gjennom selvoppvarming, konduksjon og varmestråling med retningen vist i figur 32. For å gjøre det enklere for beregningene under settes tastetiden til ett sekund.



Figur 32: En enkel framstilling av varmeflyt mellom varmematte, plate og komponent i vakuumkammeret.

Selvoppvarmingen av varmematten er beskrevet av formelen for Joule heating:

$$Q_j = P \cdot \Delta t = I^2 \cdot R \cdot \Delta t$$

videre er varmeoverføringen fra varmematten til platen beskrevet av formelen for konduksjon:

$$Q_c = \frac{k \cdot A_{varmematte \to Plate}(T_{varmematte} - T_{Plate})}{d_{varmematte \to Plate}} \cdot \Delta t$$

og til slutt er varmeoverføringen fra platen til komponenten beskrevet av formelen for varmestråling:

$$Q_r = \frac{\sigma(T_{Plate}^4 - T_{Komponent}^4)}{\frac{1-\epsilon_{Plate}}{A_{Plate} \cdot \epsilon_{Plate}} + \frac{1}{A_{Plate} \cdot F_{Plate \to Komponent}} + \frac{1-\epsilon_{Komponent}}{A_{Komponent} \cdot \epsilon_{Komponent}}} \cdot \Delta t$$

der nevneren kan forenkles til:

$$\lambda = \frac{1 - \epsilon_{Plate}}{A_{Plate} \cdot \epsilon_{Plate}} + \frac{1}{A_{Plate} \cdot F_{Plate \to Komponent}} + \frac{1 - \epsilon_{Komponent}}{A_{Komponent} \cdot \epsilon_{Komponent}}$$

Siden varmeoverføring ikke kan måles direkte må det relateres til temperatur for sammenligning. Differensiallikningene for temperatur av de ulike gjenstandene kan utledes gjennom å sette opp en energibalanse med hensyn på varmeflyten i figur 32, og varmekapasiteten av de ulike gjenstadene som henvist i ligning 6. Varmetap ut av gjenstandene er ikke blitt tatt hensyn i energibalansene, siden det virkelige systemet er langt mer kompleks. Det samme gjelder varmetap gjennom ledningene og komponentene i den elektriske kretsen.

$$f_1 = \dot{T}_{varmematte} = \frac{Q_j - Q_c}{C_{varmematte}}$$

$$f_2 = \dot{T}_{Plate} = \frac{Q_c - Q_r}{C_{Plate}}$$

$$f_3 = \dot{T}_{Komponent} = \frac{2 \cdot Q_r}{C_{Komponent}}$$

Differensiallikningene kan videre brukes til å utvikle en tilstandsrommodell for å beskrive temperaturresponsen, men siden de er svært ulineære må de først lineariseres rundt arbeidspunktene x_{ref} og u_{ref} med hjelp av de to første leddene i Taylorrekken [33] som henvist under.

$$\dot{x} = f(x, u) \approx f(x_{ref}, u_{ref}) + \frac{\partial f}{\partial x}\Big|_{x_{ref}, u_{ref}} (x - x_{ref}) + \frac{\partial f}{\partial u}\Big|_{x_{ref}, u_{ref}} (u - u_{ref})$$

Hvis x_{ref} og u_{ref} velges ved stabil tilstand vil $f(x_{ref}, u_{ref}) = 0$. Videre kan avviksvariablene introduseres som følge, $\tilde{x} = x - x_{ref}$ og $\tilde{u} = u - u_{ref}$. Til slutt kan de lineariserte differensiallikningene beskrives på formen:

$$\dot{\tilde{x}} = A\tilde{x} + B\tilde{u}$$

eller:

$$\begin{bmatrix} \dot{\tilde{T}}_{varmematte} \\ \dot{\tilde{T}}_{Plate} \\ \dot{\tilde{T}}_{Komponent} \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \cdot \begin{bmatrix} \tilde{T}_{varmematte} \\ \tilde{T}_{Plate} \\ \tilde{T}_{Komponent} \end{bmatrix} + \begin{bmatrix} b_{1,1} \\ b_{2,1} \\ b_{3,1} \end{bmatrix} \cdot \tilde{P}$$

Her representerer $a_{1,1}$ de individuelle elementene i matrise A som finnes ved å partiellderivere differensiallikningene ved stabil tilstand.

$$a_{1,1} = \frac{\partial f_1}{\partial T_{varmematte}}\Big|_{T_{ref},P_{ref}} = -\frac{k \cdot A_{varmematte \to Plate}}{C_{varmematte} \cdot d_{varmematte \to Plate}}$$

$$a_{2,2} = \frac{\partial f_2}{\partial T_{Plate}}\Big|_{T_{ref},P_{ref}} = -(\frac{k \cdot A_{varmematte \to Plate}}{C_{Plate} \cdot d_{varmematte \to Plate}} + \frac{4\sigma T_{ref}^3}{C_{Plate} \cdot \lambda})$$

$$b_{1,1} = \frac{\partial f_1}{\partial P}\Big|_{T_{ref},P_{ref}} = \frac{1}{C_{varmematte}}$$

Resultatet av matrise A og B er presentert under og kan videre brukes i Matlab for simulering med verdier gitt i tabell 1.

$$A = \begin{bmatrix} -\frac{k \cdot A_{varmematte \to Plate}}{C_{varmematte} \cdot d_{varmematte \to Plate}} & \frac{k \cdot A_{varmematte \to Plate}}{C_{varmematte} \cdot d_{varmematte \to Plate}} & 0 \\ \frac{k \cdot A_{varmematte \to Plate}}{C_{Plate} \cdot d_{varmematte \to Plate}} & -(\frac{k \cdot A_{varmematte \to Plate}}{C_{Plate} \cdot d_{varmematte \to Plate}} + \frac{4\sigma T_{ref}^3}{C_{Plate} \cdot \lambda}) & \frac{4\sigma T_{ref}^3}{C_{Plate} \cdot \lambda}) \\ 0 & \frac{8\sigma T_{ref}^3}{C_{Komponent} \cdot \lambda} & -\frac{8\sigma T_{ref}^3}{C_{Komponent} \cdot \lambda} \end{bmatrix}$$

$$B = \begin{bmatrix} \frac{1}{C_{varmematte}} \\ 0 \\ 0 \end{bmatrix}$$

Siden det er ønskelig å måle alle tilstandene vil C matrisen her representere identitetsmatrisen. For målingene antas det at D matrisene ikke er med på å forstyrre, og er dermed er lik null.

$$y(t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \tilde{T}_{varmematte} \\ \tilde{T}_{Plate} \\ \tilde{T}_{Komponent} \end{bmatrix}$$

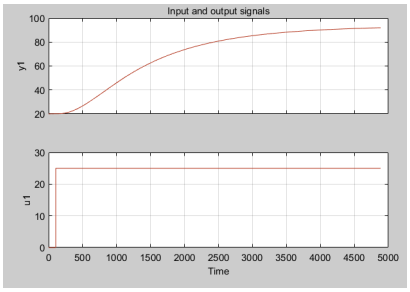
Tabell 1: Tabell for systemmatrisene

Definisjon	Parameter	Verdi	Enhet
Arealet mellom varmematte og aluminiumsplate	$A_{varmematte \to Plate}$	$7.6 \cdot 10^{-3}$	m^2
Arealet av aluminiumsplate	$A_{Aluminium}$	0.0176	m^2
Arealet av komponent	$A_{Komponent}$	0.008604	m^2
Emissivitet av varmematte	$\epsilon_{varmematte}$	0.91	enhetsløs
Emissivitet av aluminiumsplate	$\epsilon_{Aluminium}$	0.09	enhetsløs
Emissivitet av komponent	$\epsilon_{Komponent}$	0.8	enhetsløs
Spesifikk varmekapasitet av varmematte	$c_{Varmematte}$	1175	$J/Kg \cdot K$
Spesifikk varmekapasitet av aluminiumsplate	$c_{Aluminium}$	896	$J/Kg \cdot K$
Spesifikk varmekapasitet av komponent	$c_{Komponent}$	$1.15 \cdot 10^3$	$J/Kg \cdot K$
Masse av varmematte	$m_{Varmematte}$	0.0212	Kg
Masse av aluminiumsplate	$m_{Aluminiumsplate}$	0.0868	Kg
Masse av komponent	$m_{Komponent}$	0.018	Kg
Distansen mellom varmematte og aluminiumsplate	d	0.01	m
Varmeledningskoeffisienten av aluminiumsplate	k	230	$W/m \cdot K$
Stefan-Boltzmanns konstant	σ	$5.67 \cdot 10^{-8}$	$W/m^2 \cdot K^4$
View factor mellom aluminiumsplate og komponent	$F_{Plate \to Komponent}$	0.74	enhetsløs

6.4 Systemidentifikasjon

Systemidentifikasjon er en slags verktøykasse innenfor reguleringsteknikken som en ingeniør bruker til å lage matematiske modelleler av ulike prosesser basert på målt sensordata. Av og til er det vanskelig å få rene matematiske modeller til å stemme godt overens med den faktiske prosessen, da kan det var aktuelt å ta i bruk systemidentifikasjon. Bachelorgruppa er relativt ferske når det kommer til modellering av termiske systemer, så det er ingen garanti for at modellen som har blitt utviklet stemmer 100% med prosessen. Systemidentifikasjon blir dermed svært nyttig for å verifisere denne.

Den enkleste formen for systemidentifikasjon er å påføre systemet et sprang på inngangen og se hvordan utgangen reagerer. Basert på innhentet sensordata kan man tilpasse prosessen til en modell. Denne metoden er godt beskrevet i boka "Reguleringsteknikk" som er skrevet av Kåre Bjørvik og Per Hveem [34]. Bachelorgruppa gjennomførte en slik identifikasjon ved å starte prosessen i initiell tilstand (romtemperatur), for å så sette det manuelle pådraget til 25%. Slik ble følgende resultat:



Figur 33: Figuren viser sprangresponsen til TVC. U1 er inngangen og Y1 er utgangen.

Basert på "Reguleringsteknikk"-boka, kan denne responsen tilpasses en 2.ordens prosess med reele poler. Denne prosessen kan bli beskrevet slik matematisk:

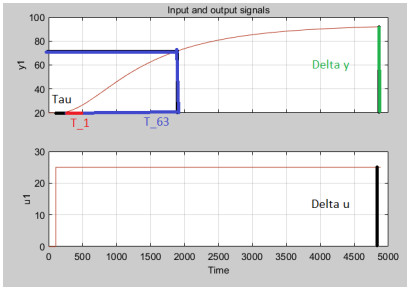
$$h_{uy}(s) = \frac{y(s)}{u(s)} = \frac{K}{(1 + T_1 \cdot s)(1 + T_2 \cdot s)} \cdot e^{-\tau \cdot s}$$

Overføringsfunksjonen til prosessen kan bli funnet via disse formlene:

$$K = \frac{\Delta y}{\Delta u}$$

$$T_2 = T_{63} - T_1$$

$$T_1 < T_2$$



Figur 34: Figuren viser hvordan man finner de ulike parameterne for prosessmodellen.

$$K = \frac{92 - 20}{25 - 0} = 2.88$$

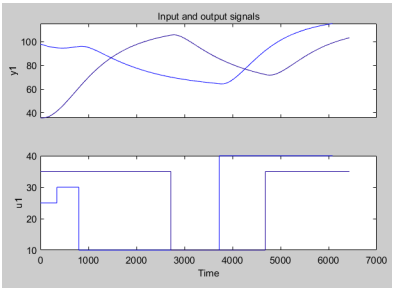
$$\tau = 7sek \cdot 8 = 56sek$$

$$T_1 = 7sek \cdot 15 = 105sek$$

$$T_2 = (142 - 24) \cdot 7sek - 105sek = 721sek$$

$$h_{uy}(s) = \frac{y(s)}{u(s)} = \frac{2.88}{(1 + 105 \cdot s)(1 + 721 \cdot s)} \cdot e^{-56 \cdot s}$$

En mer avansert form for systemidentifikasjon kan gjennomføres ved å påtrykke flere ulike pådrag, og se hvordan prosessen endrer seg. Denne prosessdataen må deretter analyseres. I denne bacheloroppgaven benyttet gruppa seg av Matlab sin "System Identification Toolbox" [35]. Denne databaserte verktøykassa kan generere modeller som overføringsfunksjoner eller på tilstandsromform. Utover sprangtesten ble det utført to andre systemtester, der operatør endret pådrag flere ganger til ulike verdier. Slik var de to følgende systemtestene:



Figur 35: Figuren viser de to systemtestene som ble brukt for å identifisere systemet.

For å få analysert denne dataen, må den importeres inn i "System Identification Toolbox". Her kan dataen videre prosesseres og estimeres. Siden en PID-regulator skal innstilles og en LQR skal simuleres trengs det å estimere en prosessmodell og en tilstandsrommodell. De modellene som ble estimert ble validert med de tre ulike systemresponsene, slik at estimert modell stemmer godt overens med alle datasettene.

Følgende modeller ble estimert med "System Identification Toolbox":

$$h_p(s) = \frac{y(s)}{u(s)} = \frac{3.9933}{(1 + 2142.8 \cdot s)(1 + 49.983 \cdot s)} \cdot e^{-68.663 \cdot s}$$

Denne modellen kan brukes til å finne riktige parametre for PID-regulatoren.

$$\dot{x} = A \cdot x + B \cdot u$$

$$\mathrm{A} = \begin{bmatrix} -0.0003828 & -0.0002208 \\ 0.002247 & -0.003235 \end{bmatrix}$$

$$\mathrm{B} = \begin{bmatrix} -3.143e-08 \\ -2.666e-05 \end{bmatrix}$$

Denne modellen kan brukes til å implementere LQR.

6.5 Simulering

Systemmatrisene hentet fra modelleringen og systemidentifikasjonen er grunnleggende for å beregne og simulere innsvingningsforløpet til prosessen med LQR algoritmen. Hovedsaklig bygger algoritmen på teorien presentert i kapittel 6.1.3, men den omformuleres litt annerledes for å ta hensyn til ikke-null likevektspunkt. Målet med algoritmen er å beregne et pådrag u_{ref} slik at systemet oppnår ønsket tilstand x_{ref} fra initialbetingelsen $x_0 \neq 0$. Fremgangsmåten for algoritmen under er hentet fra [36].

Tilstandsrommodellen for et generelt system er beskrevet på formen:

$$\dot{x} = Ax + Bu \tag{7}$$

Ved å substituere x_{ref} med x og u_{ref} med u inn i (7) utledes ligningen under som følge:

$$0 = Ax_{ref} + Bu_{ref} \tag{8}$$

Begrunnelsen for dette er at i stabil tilstand så er x_{ref} konstant og dermed blir $\dot{x}_{ref} = 0$. Videre introduseres avviksparameterne:

$$\begin{aligned} \tilde{x} &= x - x_{ref} \\ \tilde{u} &= u - u_{ref} \end{aligned} \tag{9}$$

Avviksparameterne i (9) kan løses for x og u og innholdet kan videre substitueres inn i (7) som vist under:

$$\dot{x} = \dot{\tilde{x}} + \dot{x}_{ref} = A(\tilde{x} + x_{ref}) + B(\tilde{u} + u_{ref}) \tag{10}$$

Fra tidligere er $\dot{x}_{ref} = 0$ og ved å substituere det inn i (10) utledes ligningen under:

$$\dot{x} = \dot{\tilde{x}} = A\tilde{x} + B\tilde{u} + Ax_{ref} + Bu_{ref} \tag{11}$$

Videre substitueres (8) inn i (11) for å utlede ligningen:

$$\dot{\tilde{x}} = A\tilde{x} + B\tilde{u} \tag{12}$$

Med systemet over kan LQR algortimen følge samme framgangsmåte som i kapittel 6.1.3. Målet er å optimalisere kostfunksjonen under:

$$J(\tilde{u}) = \int_0^\infty (\tilde{x}^T(t)Q\tilde{x}(t) + \tilde{u}^T(t)R\tilde{u}(t))dt \tag{13}$$

Fra teorien i kapittel 6.1.3 er løsningen på optimaliseringen av kostfunksjonen i (13) tilbakekonlingsloven på følgende form:

$$\tilde{u} = -K\tilde{x} \tag{14}$$

Ved å substituere (14) inn i (12) utledes det lukkede sløyfesystemet under:

$$\dot{\tilde{x}} = (A - BK)\tilde{x} \tag{15}$$

LQR algoritmen vil gjøre matrisen $(A-BK)$ asymptotisk stabil. Dette betyr at avviksparameteren $\tilde{x}$ vil asymptotisk gå mot null, mens tilstanden x vil bli referansen x_{ref}. Til slutt kan $\dot{x} = \dot{\tilde{x}}$ fra (11) og $\tilde{x} = x - x_{ref}$ fra (9) substitueres inn i (15) som vist under.

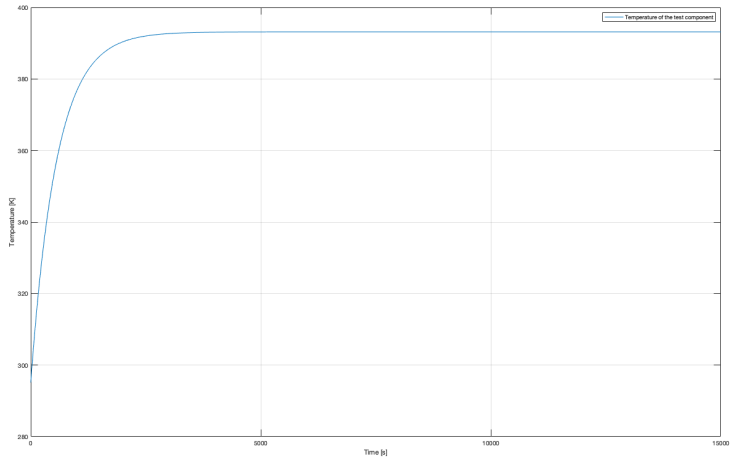
$$\dot{x} = (A - BK)x - (A - BK)x_{ref} \tag{16}$$

For å simulere det lukkede sløyfesystemet med LQR-algoritmen i Matlab må systemmatrisene fra modelleringen og systemidentifikasjonen defineres som henvist i appendiks E. Tilstandene relatert

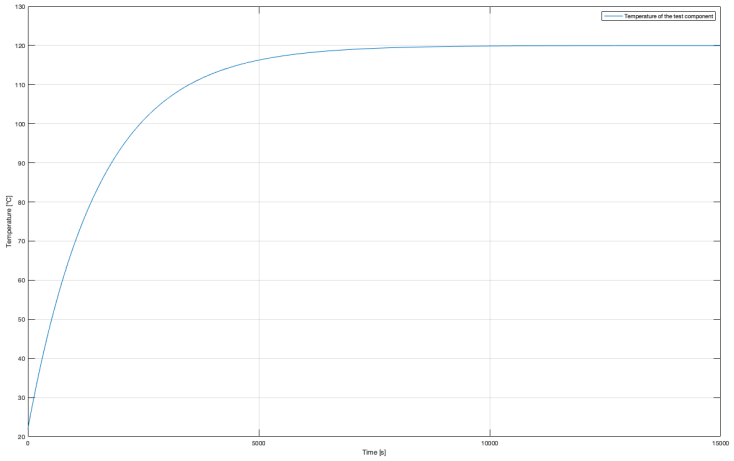
til systemmatrisene prioriteres overfor pådraget, og dermed settes det en høyere verdi for vektningsmatrisen Q enn R. Tilstanden for testkomponenten prioriteres igjen overfor de andre tilstandene. Simuleringen starter fra initialbetingelsen $x_0 = 22°C = 295.15K$ og oppnår ønsket tilstand x_{ref} ved $120°C = 393.15K$.

Ved å ta hensyn til et praktisk perspektiv som referert i kapittel 6.1.3, beregnes forsterkingsmatrisen K ved hjelp av den innebygde funksjonen *lqr*. Forsterkningsmatrise K brukes videre i det lukkede sløyfesystemet (16), og for å utvikle en tilstandsrommodell av det lukkede sløyfesystemet brukes *ss* funksjonen i matlab. Banene for tilstandene i det lukkede sløyfesystemet simuleres med *lsim*. Her tar funksjonen til seg tilstandsrommmodellen for det lukkede sløyfesystemet fra (16), ønsket tilstand x_{ref} som konstant pådrag, totaltiden for simuleringen og initialbetingelsen x_0 for tilstandene. Til slutt plottes simuleringen med *plot* funksjonen.

Simuleringen av modelleringen og systemidentifikasjonen er vist under.



Figur 36: Simulering av responsen fra modellering.



Figur 37: Simulering av responsen fra systemidentifikasjon.

7 Regulering

7.1 Implementering av PID-regulator

PID-regulatoren som var forklart i teori-delen av oppgaven er en kontinuerlig regulator. Gruppa ønsker å implementere PID-regulatoren i kode på en Raspberry Pi, som er en digital enhet. For å gjøre dette må regulatoren diskretiseres. Dessuten er den kontinuerlige PID-en en ideel regulator, når man skal implementere en praktisk regulator er det visse problemer man må ta hensyn til. Mye av denne fremgangsmåten er hentet fra forlesningsnotatene til Torleif Anstensrud i emnet digitale reguleringssystemer (IELET2102) [37].

7.1.1 Diskretisering

For å gå fra en kontinuerlig PID-regulator til en diskret, er man nødt til å diskretisere. Dette er en prosess som går ut på å endre strukturen til PID-en, slik at den tar i mot tall med et fast tidsskritt (tastetid) istedenfor en jevn strøm. Diskretiseringen skjer ved å først Laplace-transformere PID-regulatoren fra tids-domenet til S-domenet. Følgende transformasjon vil se slik ut:

$$U(s) = K_p \cdot (1 + \frac{1}{T_i \cdot s} + T_D \cdot s) \cdot E(s)$$

Deretter er man nødt til å transformere dette utrykket til z-domenet. Z-domenet er en slags versjon av s-domenet på diskret form. Transformasjonen fra s til z-domenet kan gjøres med ulike avbildninger. Bakoverdifferanse er en av disse avbildningsmetodene. Den utrykkes slik:

$$s = \frac{1 - z^{-1}}{T}$$

T = tastetid

Denne avbildningstypen vil alltid være stabil så lenge prosessen er stabil. Tastetiden som sier noe om hvor ofte data hentes fra prosessen vil ikke påvirke denne stabiliteten. Dette skyldes at prosessen vil være stabil hvis den kun har poler i venstre halvplan av s-domenet. Bakoverdifferanse vil alltid få trykt inn disse polene innenfor enhetssirkelen i z-domenet, som er stabilitetskriteriumet for z-domenet.

Etter at PID-en omgjort til z-domenet kan den løses som en differenslikning. En differenslikning uttrykkes via måling og tastesteg. Hvis en differenslikning refererer til y[n] så omhandler dette prosessverdi fra nåværende tasting, y[n-1] betyr derimot at prosessverdien skal hentes fra forrige tast (måling). Differensversjonen av PID-regulatoren vil se slik ut:

$$u[n] = u_p[n] + u_i[n] + u_d[n]$$

P-leddet til PID-regulatoren vil være følgende på differensform:

$$u_p[n] = K_p \cdot e[n]$$

I-leddet til PID-regulatoren utledes slik til differensform:

$$U_i(s) = \frac{K_p}{T_i \cdot s} \cdot E(s)$$

$$U_i(z) = \frac{K_p}{T_i \cdot \frac{1-z^{-1}}{T}} \cdot E(z)$$

$$U_i(z) \cdot (T_i - T_i \cdot z^{-1}) = K_p \cdot E(z) \cdot T$$

$$U_i(z) = K_p \cdot \frac{T}{T_i} \cdot E(z) + U_i(z) \cdot z^{-1}$$

$$u_i(n) = K_p \cdot \frac{T}{T_i} \cdot e(n) + U_i(n-1)$$

D-leddet til PID-regulatoren i s-domenet lar seg derimot ikke implementere. Dette skyldes at den er uproper. Det vil si at overføringsfunksjonen har en høyere grad i teller enn i nevner, noe som fører til utrykket krever fremtidig kunnskap om prosessen. Dette er ikke kausalt og dermed ikke mulig å få til.

7.1.2 Derivatfilter- og spark

For å kunne implementere D-leddet i en praktisk regulator er man nødt til å approksimere derivatet. Derivatet kan approksimeres som et første ordens høypassfilter, ettersom dem fungerer på ganske lik måte. En ren derivator vil se på endringer. Hvis man deriverer en stasjonær prosess vil derivatet bli null. Hvis prosessen derimot stadig endrer på seg vil derivatet bli et positivt eller negativt tall avhengig av hvordan den utvikler seg. Et høypassfilter demper lavfrekvente signaler og tillater høyfrekvente å passere. Lavfrekvente signaler endrer seg lite og vil da være sammenlignbare med stasjonære tilfeller. Høyfrekvente signaler derimot endrer seg en del og kan sammelignes med en prosess under endring. Ut ifra denne kunnskapen kan man konkludere med at et høypassfilter vil være en grei approksimasjon for en derivator. Derivatoren blir tilpasset et høypassfilter slikt:

$$U(s) = K_p \cdot T_d \cdot s \cdot E(s)$$

$$U(s) = K_p \cdot \frac{T_d \cdot s}{1 + T_f \cdot s} \cdot E(s)$$

T_f = filterkonstanten for høypassfilteret

Denne overføringsfunksjonen er proper siden teller og nevner har samme grad. Dette fører til at utrykket er kausalt, og dermed implementerbart. Filterkonstanten i overføringsfunksjonen er dessuten med på å redusere forsterkning av høyfrekvent støy. Denne filterkonstanten utrykkes slik:

$$T_f = \frac{T_d}{N}$$

N settes ofte til et tall mellom 5 og 10.

Et annet problem med derivatleddet er at det kan forårsake et såkalt derivatspark ved endring av settpunkt (ønsket temperatur). Derivatspark er et svært høyt uønsket regulatorpådrag, som er forårsaket av at spranget til settpunktverdien blir derivert i det regulatoren beregner avviket. For unngå å derivere et sprang kan man rett og slett fjerne det fra leddet slik at man sitter igjen med dette:

$$e_d(t) = -y(t)$$

Det går greit å fjerne settpunktet i denne sammenhengen siden den hovedsaklig vil være konstant. Når en deriver en konstant vil den være null, så den vil ikke bidra med noe verdi annet enn i spranget da dette bidraget er uønsket. Siden denne bacheloroppgaven dreier seg om prosessregulering er dette en grei måte å løse dette problemet på. Hadde oppgaven derimot omhandlet regulering av et system der settpunktet stadig var i endring, som for eksempel banefølging med en robotmanipulator hadde dette vært en dårlig idé. Ved å ta hensyn til disse to problemene vil det endelige D-leddet se slik ut:

$$U_d(s) = K_p \cdot \frac{T_d \cdot s}{1 + T_f \cdot s} \cdot -Y(s)$$

$$U_d(z) + T_f \cdot \frac{1 - z^{-1}}{T} \cdot U_d(z) = -K_p \cdot T_d \cdot \frac{1 - z^{-1}}{T} \cdot Y(z)$$

$$U_d(z) \cdot (T + T_f) = U_d(z) \cdot z^{-1} \cdot T_f - K_p \cdot T_d \cdot (Y(z) - Y(z) \cdot z^{-1})$$

$$U_d(n) = \frac{T_f}{T + T_f} \cdot U_d(n-1) - K_p \cdot \frac{T_d}{T + T_f} \cdot (y(n) - y(n-1))$$

$$U_d(n) = \frac{T_d}{N \cdot T + T_d} \cdot U_d(n-1) - K_p \cdot \frac{N \cdot T_d}{N \cdot T + T_d} \cdot (y(n) - y(n-1))$$

$$U_d(n) = \beta \cdot U_d(n-1) - K_p \cdot \frac{T_d}{T} \cdot (1 - \beta) \cdot (y(n) - y(n-1))$$

$$\beta = \frac{T_d}{T_d + T \cdot N}$$

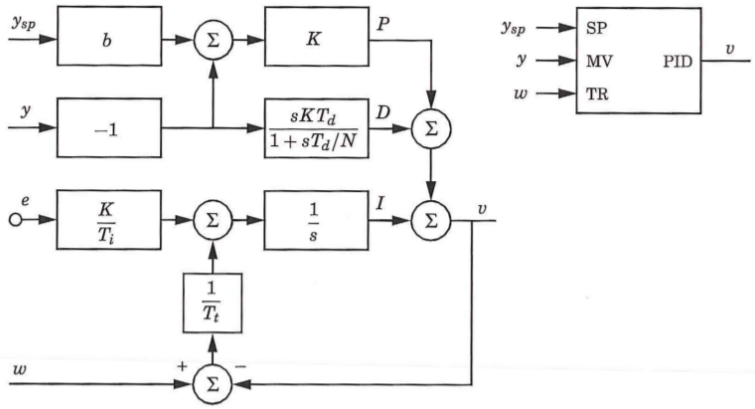
7.1.3 Windup

En annen utfording som dukker opp når en skal implementere en PID-regulator er hvordan man skal løse problemet med integrator "windup". Integrator "windup" er et fenomen som ofte oppstår på grunn av I-leddet i PID-regulatoren, og skyldes at et pådragsorgan har en maksimal og minimal grense for hvor stort pådrag de kan yte (f.eks. en ventil vil ikke kunne åpne seg mer enn den er konstruert for). Ettersom pådragsorganet kun har et vist intervall de kan yte innenfor vil dette føre til at pådragsorganet ikke kan bidra med like mye pådrag som regulatoren muligens har beregnet. Hvis et slikt tilfelle skulle oppstå vil ikke prosessverdien legge seg stabilt på referansen. Det vil derimot oppstå et stasjonært avvik selv om man har med en integralvirkning i regulatoren. Videre vil denne tilstanden føre til at integralvirkningen vil vokse og bygge seg opp, selv om prosessverdien ikke endrer på seg. Dette er et problem, fordi reguleringssystemet vil bruke unødvendig mye tid på å reagere ved en referanseendring. Dette skyldes at integratoren må fjerne pådragsoverskuddet den bygde opp i "windup" perioden, før den vil operere som normalt. Problemet med "windup" vil kun oppstå med regulatorer som tar i bruk en integralvirkning (P og PD regulatorer kan ikke oppleve windup).

I denne bacheloroppgaven vil dette problemet oppstå med at det er PWM-signalet fra Raspberry Pi-en som er pådraget. Dette PWM-signalet styres ved å sette "duty cycle" til signalet lik det PID-regulatoren beregner. "Duty cycle" er nødt til å være et tall mellom null og hundre ettersom dette forteller noe om hvor stor prosentandel av signalet er høyt. Dette tallet kan ikke være negativt eller over hundre, selv om PID-regulator vil beregne det. Derfor er det nødvendig å innføre en løsning for "windup" og begrensning av pådrag.

Det finnes flere måter og løse problemet med "windup" på. I denne bacheloroppgaven har det blitt implementert to ulike metoder: "clamping" og "back-calculation". "Clamping" løser dette

problemet med å kode inn en egen logikk som stopper integratoren ved metning. Dette er en svært enkel løsning på problemet, og denne koden ligger i appendiks. "Back-calculation" derimot løser problemet med "windup" ved å innføre en egen tilbakekobling fra pådraget som fôres inn i integralvirkningen. Denne tilbakekoblingen sørger for at pådraget fra regulatoren ikke overskrider metningen. Selve implementeringen av denne løsningen er beskrevet i "Advanced PID control" av K.J. Åström og T. Hägglund. Implementeringsmetoden er lagt til nedenfor:



Figur 38: Figuren viser hvordan K.J. Åström og T. Hägglund har implementert "anti- windup" med en egen "tracking"-inngang.(Åström & Hägglund,1995,s. 86 [38])

For å implementere denne løsningen i I-leddet tas det utgangspunkt i figuren ovenfor. I følge figuren kan en bare bygge videre på det I-leddet man allerede har utviklet og bare summere det med "Tracking"-inngangen. Det nye utrykket for I-leddet vi bli slik:

$$u_i(n) = K_p \cdot \frac{T}{T_i} \cdot e(n) + \frac{T}{T_t} \cdot (w(n) - v(n)) + U_i(n-1)$$

w = "tracking"-inngangen

v = totalt pådrag fra PID-regulator

7.1.4 Implementasjon av PID i Python

Siden PID-regulatoren implementeres på en Raspberry Pi, ble det bestemt at den skulle kodes i python. Python er et relativt enkelt og oversiktlig programmeringspråk, som prosjektgruppa har god erfaring med. Språket er også veldig godt integrert med ulike Raspberry Pi-enheter. Det kommer blant annet installert med operativsystemet. Selve PID-programmet som ble skrevet består av en rekke seksjoner: initialisering, PID-struktur, manuell styring, avlesning av parametre, og PWM.

For å kjøre koden må alle de ulike variablene bli initialisert og de nødvendige bibliotekene må importeres. Tabellen under er en forklaring på hva de ulike variabelnavnene i koden betyr. Flere av variablene der er satt som "array" i koden. Dette skyldes at disse variablene er nødt til å huske tidligere verdier for å kunne beregne pådraget ut fra PID-regulatoren.

Variabelnavn:	Betydning:
Ts	Tastetid (hvor ofte ny måling hentes)
SP	Settpunkt (Ønsket temperatur for regulering)
K_p	Parameter for P-leddet
T_i	Parameter for I-leddet
T_d	Parameter for D-leddet
N	Tall mellom 5 og 10
ManVal	Manuell pådragsverdi (0-100)
U_{total}	Totalt pådrag fra PID-regulator
U_p	Pådrag fra P-leddet
U_i	Pådrag fra I-leddet
U_d	Pådrag fra D-leddet
PV	Prosessverdi (Målt temperatur)
e	Avvik
Auto	Modus: 1 (PID) og 0 (Manuell)

I tillegg til å deklarere variabler, blir også PWM-utgangspinnen deklarert. Det er en fysisk utgangspinne som står på Raspberry Pi-kortet, som er tilkoblet transistorkretsen. Det er kun noen få av pinnene på Raspberry Pi-kortet som støtter PWM. Pinne 13 er en av disse.

PID-koden benytter seg av følgende biblioteker: RPi.GPIO, time, os.path og csv. RPi.GPIO biblioteket brukes i denne koden for å iverksette den spesifiserte PWM-pinnen, endre på "duty cycle" til pådragssignalet og avslutte. Biblioteket os.path og csv brukes til å sjekke om filene koden trenger for å kjøre eksisterer. Om de ikke gjør det, oppretter koden en ny fil den kan avlese. Dessuten brukes dem til å skrive aktuell data til fil.

Programmet er skrevet og designet slik at det alltid vil starte i manuell modus med et manuelt pådrag på null (ingen effekt). I manuell modus kan operatør velge pådraget helt selv. Operatøren kan hele tiden velge og endre modus. Hvis operatør endrer til auto-modus vil PID-regulatoren ta over og bestemme pådraget basert på dens beregninger. Operatør kan i auto-modus endre på referanse og PID-parametre. Selve PID-strukturen var relativ enkel å implementere, ettersom hvert av PID-leddene er utledet ovenfor i form av differenslikninger. Disse differenslikningene ble satt direkte inn i koden. For å unngå feilsituasjoner er koden nødt til kunne håndtere divisjon med null, slik at parameterene kan settes til hva enn brukeren skulle ønske uten at det skal oppstå problemer. PID-en inneholder også sensoravlesnings-kode som henter inn sensordata fra temperatursensor, slik at den kan beregne avvik basert på dette. Regulatoren trenger også å oppdatere variabelverdiene inne i arrayene. Det er også lagt inn "clamping" av det totale pådraget slik at denne verdien aldri vil overskride hundre eller havne under null. Ettersom PID-regulatoren skal kjøre kontinuerlig må denne koden settes inn i en evig løkke, som skal kunne bli brutt av brukeren enten ved å bytte modus eller ved avslutning.

Det var ønskelig at PID-regulatoren skulle kunne motta eksterne endringer uten å stanse koden, slik som endring av referanse og parametre. For å få til dette eksperimenterte gruppa med flere ulike løsninger. En løsning som ble arbeidet med lenge var å ta i bruk multitråd-programmering.

Gruppa fikk laget en prototype av denne sorten, men denne ble ikke implementert i det endelige systemet på grunn av sin kompleksitet. Istedenfor gikk gruppa for en enklere løsning med skriving og avlesing av en config-fil. Ved oppstart av programmet vil det bli opprettet en fil som inneholder de initielle parameterne. Denne config-filen, blir lest av PID-koden kontinuerlig for å se etter parameterendringer. Parameterendringene kommer av at operatøren kan bruke det grafiske brukergrensesnittet til å skrive til config-fila, og dermed oppdatere de ulike verdiene som mottas av PID-koden. Hele PID-koden er lagt til i appendiks.

7.1.5 Valg av tastetid

Det er ikke alltid like lett å velge en god tastetid, siden det er flere aspekter en må ta hensyn til. En altfor treg tastetid kan føre til at det rekonstruerte signalet vil være ganske annerledes, såkalte aliaser. Dette er en ting som bør unngås ved valg av tastetid. Dette skyldes at informasjon mellom tastingene vil gå tapt. Det er derfor nødvendig at denne informasjonen ikke er viktig, eller at det er mulig å rekonstruere informasjonen basert på kjennskap om systemet. En rask tastetid derimot vil taste mer enn nødvendig, noe som vil tære unødvendig mye på lagringsplassen til systemet. Det er dermed viktig å finne en tastetid som ikke er altfor rask eller treg.

Når en velger tastetid er det viktig at tastefrekvensen er større enn maksfrekvensen til signalet som skal rekonstrueres. Nyquist-Shannon tasteteoremet brukes derfor for å velge en tastetid rask nok til å unngå tap av nødvendig informasjon, mens det samtidig begrenser tastetiden slik at en ikke bruker altfor mye unødvendig lagringsplass. Tastetiden $T_s = 1$s ble valgt.

7.1.6 Valg av parameterverdier

Det finnes mange ulike måter å finne gode parameterverdier for PID-regulatoren slik at innsvingningsløpet blir som ønsket. Orbit NTNU ønsker å oppnå en temperatur på 120 grader så raskt som mulig med minimalt oversving. Regulatoren er nødt til å holde temperaturen innenfor 115 og 125 grader.

En av metodene for å finne brukbare parameterverdier er Ziegler-Nichols. Ziegler-Nichols går ut på å kjøre regulatoren med bare P-leddet for å så øke denne parameteren til det oppstår stående svingninger. Forsterkningsverdien til P-leddet som forårsaker disse stående svigningene kalles for kritisk forsterkning, og perioden på oscillasjonene kalles kritisk periodetid. Parameterne finnes så ved denne tabellen:

Regulatortype	K_p	T_i	T_d
P	$0.5 \cdot K_k$	0	0
PD	$0.65 \cdot K_k$	0	$0.12 \cdot T_k$
PI	$0.45 \cdot K_k$	$0.85 \cdot T_k$	0
PID	$0.65 \cdot K_k$	$0.5 \cdot T_k$	$0.12 \cdot T_k$

Ved å ta i bruk de to prosessmodellene fra kapittelet om systemidentifikasjon kan man simulere systemet med en P-regulator. En kan dermed stille K_p helt til det oppstår stående svingninger. Dette skjer ved en K_p på 6 for sprangresponsen og 10 for den datadrevne. Perioden på svingningene til sprangresponsen er 375 og den til den datadrevne er 433.3 sekunder. Basert på dette finnes følgende parameterverdier:

$K_p = 0.65 \cdot 6 = 3.9$ eller $K_p = 0.65 \cdot 10 = 6.5$

$T_i = 0.5 \cdot 375 = 187.5$ eller $T_i = 0.5 \cdot 433.3 = 216.65$

$T_d = 0.12 \cdot 375 = 45$ eller $T_d = 0.12 \cdot 433.3 = 51.996$

SIMC eller Skogestads metode er en annen måte å finne parameterverdier på. Denne metoden baserer seg på direkte syntese. Mer informasjon om denne metoden finnes i artikkelen “The SIMC

method for smooth PID controller tuning" skrevet av Sigurd Skogestad og Chriss Grimholt [39]. Deres metode tar utgangspunkt i en 1. eller 2.ordens prosessmodell med tidsforsinkelse. Parameterne finnes slik:

$K_p = \frac{1}{k} \cdot \frac{T_1}{\tau_c + \tau}$

$T_i = min(T_1, 4 \cdot (\tau_c + \tau))$

$T_d = T_2$

Det er bare en justeringsparameter en kan endre på, τ_c. Verdien til τ_c avhenger av hva slags ytelse man ønsker. Man kan velge å vektlegge hurtighet eller glatthet.

$\tau_c = 0.5 \cdot \tau$ (aggresivt)
$\tau_c = 3 \cdot \tau$ (robust)

Ved å ta i bruk de to prosessmodellene fra kapitellet om systemidentifikasjon kan man beregne ulike verdier for PID-parameterne:

$K_p = \frac{1}{2.88} \cdot \frac{721}{\tau_c + 56} = (1.1, 2.98)$ eller $\frac{1}{3.9933} \cdot \frac{2142.8}{\tau_c + 68.663} = (1.95, 5.2)$

$T_i = min(721, 4 \cdot (\tau_c + 56)) = (336, 896)$ eller $min(2142.8, 4 \cdot (\tau_c + 68.663)) = (411.92, 2142.8)$

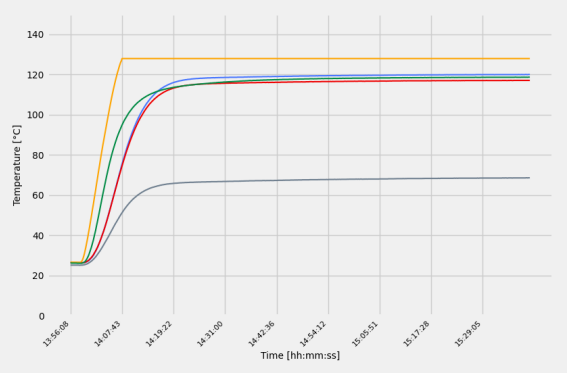
$T_d = 105$ eller 49.983

Ut ifra beregningene som er gjort via de to metodene med to ulike prosessmodeller bør parameterne til systemet være innenfor disse intervallene:

$K_p = (1.1, 6.5)$
$T_i = (187.5, 2142.8)$
$T_d = (45, 105)$

Metodene som ble beskrevet ovenfor baserer seg på prosessmodeller. Disse modellene stemmer aldri 100% med den faktiske prosessen. For å finne gode parametere til den faktiske prosessen blir man nødt til å drive med noe manuell etterjustering. Når en driver med dette er det lurt å følgende i bakhodet: en økning av P-forsterkingen gir som regel en raskere og mer urolig prosess, en reduksjon av integraltiden fjerner stasjonært avvik raskere men fører samtidig til at prosessen blir mer urolig og en økning av derivasjonstiden vil opp til et visst nivå føre til en raskere og mer stabil prosess. Disse reglene er hentet fra forlesningen om manuell etterjustering i emnet "Reguleringsteknikk 2" som ble avholdt av Fredrik Dessen [40]. Etter mange tester med ulike parametere kom gruppa frem til følgende verdier:

$K_p = 1.1$
$T_i = 180$
$T_d = 13.5$

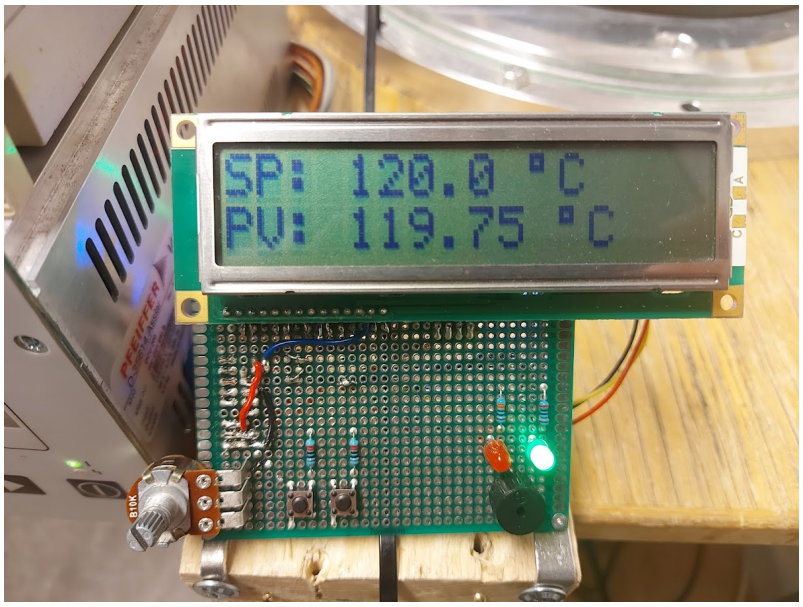


Figur 39: Figuren viser innsvingsforløpet til prosessen med PID-parametere funnet med manuell etterjustering. Den blå grafen viser temperaturen til testkomponenten.

8 Brukergrensesnitt

8.1 Det fysiske brukergrensesnittet

Det som gjør versjon 4 av FTTR komplett er et fysisk brukergrensesnitt i tillegg til det grafiske, som visualiserer prosessverdien i sanntid og viser settpunkt og manuell verdi. Det ferdigstilte systemet kan også endre mellom de to modusene auto og manuell, samt at det tillater justering av settpunkt og manuell verdi. Dette gjør at en ikke nødvendigvis har behov for det grafiske aspektet ved brukergrensesnittet, dersom regulatorverdiene er satt på forhånd.



Figur 40: Bildet viser det fysiske brukergrensesnittet.

8.1.1 Oppkobling

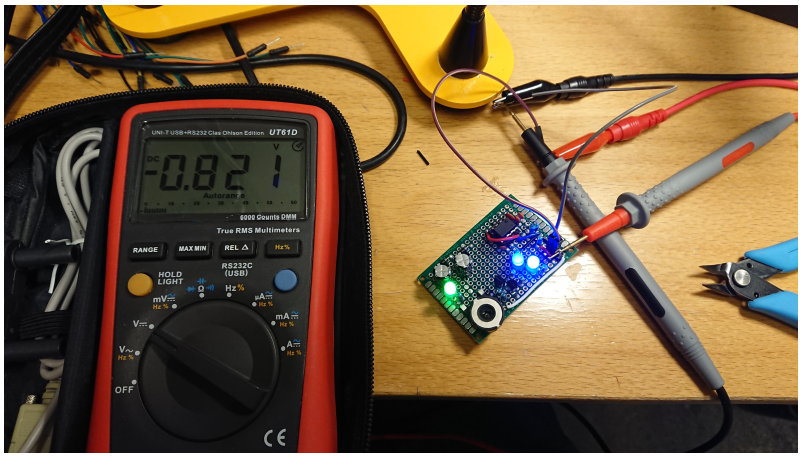
Alle komponentene som skulle brukes hadde blitt kjøpt inn tidlig i prosjektperioden. Det var snakk om en HD44780-LCD-modul, en I^2C-utvidelse basert på PCF8574, to trykknapper, en rød og en grønn lysdiode, en piezospeaker, et potensiometer, og til slutt et perfboard som alt skal loddes fast på. Disse skal gjøre det mulig å få input fra brukeren og gi informasjon om prosessen til brukeren. I^2C-utvidelsen gjør det mulig å styre LCD-modulen over I^2C. LCD-modulen og I^2C-utvidelsen har header-pinner som tilkobling. For å gjøre montering og demontering enkelt ble det bestemt at disse skulle monteres i passende sokler som igjen skal loddes fast i perfboardet. Da vil også feilsøking bli mindre arbeidskrevende. Sidetilkoblingene på I^2C-utvidelsen ble også tilkoblet med en sokkel loddet på fire ledninger som igjen er loddet fast i perfboardet. Det ble montert flere grupper med header-pins på kortet. Tre pinner ble brukt til jord, $3,3V$ og $5V$ inn fra Pi-en. Fem pinner ble brukt til signaler fra Pi-en. Dette var to lysdioder, en piezospeaker, og SDA og SCL som er til I^2C-kommunikasjon med I^2C-utvidelsen. Tre pinner overfører signaler tilbake til Pi-en. Disse brukes til to trykknapper og analog spenning fra potensiometeret. Tre pinner går også videre til kontrastspenningskretsen og disse er forklart i kapittelet om kontrastspenningskretsen. (8.1.2) Det ble laget en ledningsbunt med alle de nødvendige koblingene mellom Pi-en og brukergrensesnittet. Denne plugges direkte inn i brukergrensesnittet og den andre enden kobles til Pi-en med jumper-ledninger. Kortet ble montert fast til en plankebit med to metallfester som gruppa lagde selv. Strips ble brukt for å hindre at plankebiten sklir utfor kanten på tralla. Til slutt ble ledningsbunten teipet sammen og teipet fast til tralla flere steder.

8.1.2 Kontrastspenningskrets

En av tilkoblingspinnene på LCD-modulen, pinne 3, styrer kontrasten. Det går an å styre kontrasten på slike LCD-er med et potensiometer. Kontrastpinnen kobles til sleperen, og de to endene av potensiometeret kobles til jord og spenning. Lavere spenning betyr mer kontrast. Det er også slik potensiometeret på I^2C-utvidelsen er koblet. For denne LCD-modulen viste det seg at $0V$ på pinne 3 ikke var lavt nok. Det var mulig å se hva som sto på skjermen, men det var ikke med stor margin. Det ville vært frustrerende å bruke brukergrensesnittet hvis skjermen skulle forblitt så utydelig. Derfor bestemte gruppa seg for å prøve å koble en negativ spenning til potensiometeret i stedet for en positiv. Da ble det mulig å stille inn en optimal kontrast, og det ble mye enklere å se hva som sto på skjermen, selv på avstand. Når gruppa hadde stilt inn kontrasten slik at kontrasten ble optimal, ble spenningen og strømmen på pinne tre målt. Spenningen var $-820mV$ og strømmen var $-680\mu A$ Dermed ble det bestemt at det skulle bygges en egen krets som kan omforme $5V$ fra Pi-en til en negativ spenning som deretter kan justeres med et potensiometer og kobles til pinne 3. Potensiometeret skal kunne brukes til å stille inn kontrasten.

Det viste seg at den enkleste måten å lage en slik negativ spenning på, var ved hjelp av en "charge pump" krets. Dette er en DC-DC omformer som bruker kondensatorer til å lagre energi i stedet for spoler som er mer vanlig [41]. Disse må kobles frem og tilbake på en slik måte, at det blir en annen spenning på utgangskondensatoren enn den som er på inngangen. Det kan også gjøres slik at spenningen på utgangskondensatoren blir negativ, noe gruppa skal gjøre. I gruppas krets fungerer det slik: Først lades C5 opp til $4,3V$ via Q1 og D3. Deretter vil C5 lade opp C6 til $(U_{C5} - 0,7V = 3,6V)$ via Q2 og D4. Spenningen på C6 vil bare bli circa $3,6V$ fordi $0,7V$ blir tapt over hver av diodene. Denne spenningen er likevel mer enn nok til formålet. Disse to stegene må gjentas med en fast frekvens. Det vil ta noen sykluser før denne spenningen er oppnådd på utgangen, fordi C5 mister spenning samtidig som den lader opp C6.

Fordi den positive tilkoblingen til C5 må kobles frem og tilbake mellom $5V$ og jord, trenger kretsen et firkantsignal med fast frekvens, og circa 50% arbeidssyklus. Dette signalet kan lages på mange måter, men en enkel og tradisjonell måte å gjøre dette på, er å bruke en NE555-timer. Denne kan kobles som en astabil vippe. Dette blir da en oscillator som gir et firkantsignal ut. Frekvensen kan justeres ved å forandre på resistansen til R1, R2 og R3 og kapasitansen C2 og C3. I dette tilfellet er det ikke nødvendig å finjustere frekvensen. Det holder at den er omtrent riktig.



Figur 41: Bildet viser det det ferdig loddede kortet med Kontrastspenningskretsen. Her foregår en test der potensiometeret har blitt stilt inn til den spenningen som LCD-Modulen skal ha.

For å finne komponentverdiene prøvde gruppa seg frem med ulike verdier. Ønsket var at frekvensen skulle bli mellom 50Hz og 100Hz. Kondensatoren skulle være liten nok til at en keramisk en kunne brukes. Gruppa lyktes med denne strategien. Formelen for frekvens og utregning med gruppas komponentverdier:

$$f = \frac{1}{t_{high} + t_{low}} = \frac{1}{ln(2) * (R1 + 2 * (R2||R3)) * C} = 69,56Hz$$

For at arbeidssyklusen skal bli nær 50% må $R1$ være mye mindre enn $R2||R3$. Her er utregningen av arbeidssyklus for gruppas krets:

$$Arbeidssyklus = \frac{t_{high}}{t_{high} + t_{low}} * 100\% = \frac{R1 + (R2||R3)}{R1 + 2 * (R2||R3)} * 100\%$$

[42]

Utgangen fra NE555 ble først koblet direkte til "charge pump" kretsen. Dette fungerte delvis, men det viste seg at utgangsimpedansen fra NE555 var for høy til å kunne levere full amplitude til "charge pump" kretsen. Når gruppa kortsluttet utgangskondensatoren med et amperemeter ble det kun målt $-2mA$. Dette gir lite margin, fordi både potmeteret, LCD-modulen og en indikator-LED skal trekke strøm fra utgangen. Da ble det bestemt at en driverkrets skulle designes. Denne skal stå mellom NE555P og "charge pump" kretsen. Den må ha høy inngangsimpedans og lav utgangsimpedans. Da vil den ikke belaste NE555P for mye og den vil kunne levere full spenning til "charge pump" kretsen. Dette vil gi den høyeste mulige negative spenningen ut fra "charge pump" delen av kretsen.

BJT-småsignaltransistorer ble brukt i driverkretsen fordi disse var lett tilgjengelige. For å få minst mulig spenningsfall over transistorene når de er på, ble det valgt å bruke PNP koblet til $5V$ og NPN koblet til jord. For å unngå at begge transistorene er på samtidig og kortslutter $5V$ fra Pi-en, ble det lagt inn to blå lysdioder. Disse skal i utgangspunktet kun fungere som en diode med høyt spenningsfall, men de har også den bieffekten at de gir litt informasjon om signalet som kommer inn til kretsen. Ved 50% skal de lyse omtrent like sterkt. Diodene gjør at inngangsspenningen må over en terskelverdi for å koble utgangen til jord, og under en annen terskelverdi for å koble den til $5V$. Dette betyr også at driveren inverterer signalet, men siden arbeidssyklusen er cirka 50% vil det ikke ha noe å si i denne kretsen. Inngangen er slik at når den ikke er koblet til, hviler spenningen på $2,5V$. Da vil utgangen ha høy impedans, og begge lysdiodene vil lyse svakt.

Gruppa prøvde seg frem og fant ut at $10k\Omega$ for R4 og R5 fungerte bra. Dette ga $110mV$ over R4 og det samme over R5. Dette er lite nok til at begge transistorene skal være helt av. Spenningen over D1 ble $2,40V$ og det samme over D2. $R8$ og $R9$ ble satt inn for å hindre at strømmen inn og ut av utgangen skulle kunne overstige $I_C(max)$ som er $100mA$.

$$R8||R9 = \frac{5V}{100mA} = 50\Omega = 100\Omega||100\Omega$$

For å være helt sikker på at transistorene går i metning brukte gruppa $I_C < 1000mA$. Utregning for begge transistorene:

$$\beta > 200,\ \frac{1000mA}{200} = 5mA$$

Det er ikke nødvendig å ta hensyn til verdien av R4 og R5 når R6 og R7 skal regnes ut. Dette fordi spenningen over lysdiodene forandres relativt lite, selv om strømmen gjennom forandres mye.

$$U_{R6/R7} = 5V - 2,7V - 0,7V = 1,6V,\ \frac{1,6V}{5mA} = 320\Omega,\ R6/R7 = 330\Omega$$

En grønn LED drevet av den negative utgangsspenningen skal indikere at kretsen virker. Gruppa ønsker circa $1mA$ gjennom dioden.

$$U_{R10} = 3,6V - 2,3V = 1,3V,\ \frac{1,3V}{1mA} = 1300\Omega,\ R10 = 1000\Omega$$

Til slutt ble den negative spenningen og jord koblet til potensiometeret. Fordi strømmen til pinne 3 vil være ca $-680\mu A$ er det lurt at potmeteret leder en god del mer kontinuerlig. Dette for å unngå at sammenhengen mellom potensiometerets vinkel og kontrasten blir for ulineær. Gruppa valgte å bruke et potensiometer på $1k\Omega$. Da vil strømmen bli:

$$\frac{-3,6V}{1000\Omega} = -3,6mA$$

Alt dette ble loddet fast på et lite perfboard. NE555 ble montert i en IC-sokkel, noe som gjør det enkelt å skifte den ut dersom dette skulle bli nødvendig. Jord og $5V$ kobles inn med jumperledninger på perfboardet. Det samme gjelder negativ kontrastspenning ut fra potensiometeret.

8.1.3 Bibliotek

Biblioteker som er benyttet i koden til det fysiske brukergrensesnittet er RPi.GPIO, RPLCD, time, board, busio, adafruit_ads1x15.ads1115, Mode fra sist nevnte bibliotek og AnalogIn fra adafruit_ads1x15.analog_in. RPi.GPIO brukes for å initialisere knapper og led-dioder og gi disse ønsket funksjonalitet. RPLCD-biblioteket tillater skriving til LCD-skjermen og egner seg godt til Hitachi HD44780-kontrolleren, samt I^2C-utvidelsen PCF8574, som blir brukt i denne kretsen. RPLCD er inspirert av Adafruit Industries sitt CharLCD-bibliotek og Arduino sitt LiquidCrystal-bibliotek. Time-biblioteket brukes til tidsforsinkelser i koden. Busio og board brukes for initialisering av I^2C til ADS1115 og adafruit_ads1x15.ads1115 er biblioteket til ADC-modulen ADS1115. Fra adafruit_ads1x15.analog_in importeres AnalogIn, som brukes til å lese analoge verdier fra ADC-en. En oversikt over de ulike bibliotekene finnes i appendiks G.

8.1.4 Meny-system

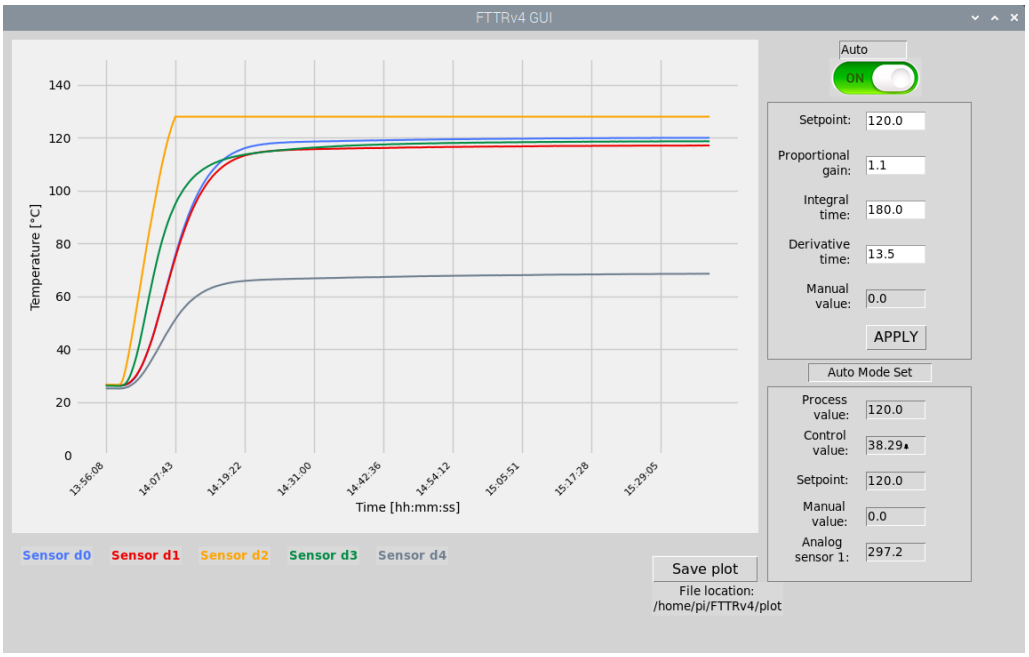
Når koden til det fysiske brukergrensesnittet starter er den i kun-lese-modus. Det som vises på LCD-en er prosessverdi og avhengig om systemet er i auto eller manuell modus, vises settpunkt eller manuell verdi. Når den venstre knappen trykkes i denne modusen, vil man komme inn i modusen for å justere på enten manuell verdi eller settpunkt. Potensiometeret brukes for å justere og den høyre knappen for å sette verdien. En bekreftende melding kommer opp rett etter at verdien er satt, samt at en av LED-ene blinker for å indikere at den er lagret. Etter dette går den tilbake til kun-lese-modus. I kun-lese-modus kan brukeren trykke inn den høyre knappen for å endre mellom auto og manuell modus. Igjen kommer det opp en bekreftende melding og blinkende LED for å indikere at modus er endret.

8.1.5 Programvare

Etter at bibliotekene er importert blir I^2C og ADS1115 initialisert. For å senere kunne både lese og skrive til pid.conf må regulatorverdiene få en initiell verdi. Knappene er satt opp som GPIO.IN, siden de mottar et knappetrykk (inngangssignal). Siden de er koblet opp med "pull-up"-konfigurasjon, er dette også definert i oppsettet til knappene i koden. Knappene er også satt opp med en "event detect"-funksjon slik at de registrerer trykk på stigende flanke. LCD-en blir satt opp i I^2C-modus med definert antall kolonner og rader, samt et spesifikt karakter-sett og I^2C-utvidelse, adresse og port. Koden starter i en "try-except"-blokk, som starter med en velkomst-melding. Den går deretter inn i en "while"-løkke der den først kjører funksjonen for lesing av pid.conf. En "if"-løkke sjekker om auto er satt til 1 eller 0 og går inn i de respektive funksjonene for enten auto eller manuell. Funksjonene for auto og manuell (showAll_A og showAll_M) fungerer på samme måte. Forskjellene er hvilken verdi som skal settes, hvilken informasjon som skrives til skjermen og hvilken LED som lyser/blinker. I auto-modus er det settpunkt og i manuell-modus er det manuell verdi. De to funksjonene **auto_mode** og **manual_mode** brukes til å justere og skrive verdiene til pid.conf. For å avslutte koden trykker man Ctrl + C. Da havner man i "except KeyboardInterrupt" der en farvel-melding kommer opp på skjermen, samt at LCD-en og GPIO-portene ryddes.

8.2 Det grafiske brukergrensesnittet

Den forrige versjonen av FTTR baserte seg kun på styring av systemet fra terminalvinduet på en ekstern maskin. Den hadde den mest grunnleggende funksjonaliteten, som blant annet styring av PPS og logging av temperaturdata. I den nye versjonen, FTTRv4 var planen å lage et grafisk kontrollpanel der en kan endre på regulatorparametere og lese av ulike verdier. Logging av data skulle skje i bakgrunnen og disse skulle plottes i etterkant. Det ble også lagt inn et grafvindu i brukergrensesnittet som oppdaterer seg i sanntid. Med dette kan brukeren følge med på om systemet oppfører seg som forventet, og har i tillegg muligheten til å endre på parameterverdiene under testing.



Figur 42: Skjermbildet viser det grafiske brukergrensesnittet.

8.2.1 Bibliotek

Det er benyttet flere av de samme bibliotekene i koden for det grafiske brukergrensesnittet som både i PID-koden og LCD-koden. De bibliotekene som er unike for denne koden er blant annet tkinter, FigureCanvasTkAgg, FuncAnimation og Pyplot fra Matplotlib. De grafiske delene av brukergrensesnittet er basert på tkinter fra Python. FigureCanvasTkAgg er en modul fra Matplotlib som brukes til å opprette et grafvindu som blir innebygd i brukergrensesnittet. Pandas brukes til å lese av filen for temperaturdata og numpy til tilordning av elementene på x-aksen i grafen. En oversikt over de ulike bibliotekene finnes i appendiks G.

8.2.2 Logger

FTTRv4_temp er programmet som henter temperaturdata fra de digitale sensorene og behandler de. Det blir dokumentert i appendiks G hvordan "One-wire"-grensesnittet blir satt opp på Raspberry Pi. De ulike sensorene befinner seg i mappen **/sys/bus/w1/devices**, og denne blir definert som *base_dir* i koden. For å kunne lagre temperaturdata til en ny fil hver gang koden kjører, blir det først definert en spesifikk mappe der disse blir lagret. Funksjonen **init_time** sørger for at det blir opprettet en ny fil for temperaturdata hver gang og benytter seg av biblioteket "datetime" for å gi navn til filene. Filnavnet er dato og klokkeslett ved opprettelse av filen. På denne måten blir det enklere å holde orden på filene. Funksjonen **read_temp_raw** leser rådata fra *base_dir*-mappen og returnerer disse, slik at de kan konverteres til temperatur i Celsius i funksjonen **convert_temp**.

De konverterte temperaturene blir deretter lest i **read_temp**-funksjonen og lagt i en liste. Her kan en også endre på hvor mange sensorer en vil lese i for-løkken ved å endre på området. **read_temp0** henter ut data fra den første sensoren i listen *temps*, som brukes i PID-koden til å regulere etter. Det opprettes både en tekstfil til bruk i live-plotteren og en ny fil som blir opprettet hver gang koden kjører. Filen til live-plotteren blir overskrevet hver gang koden kjører, og er nødvendig for å kunne både plotte live og lagre den samme temperaturdataen samtidig. Den siste funksjonen skriver både tidspunkt og temperaturdata til disse to filene.

8.2.3 Plotter

Når temperaturdata er skrevet til fil, kan den plottes. Plottinga skjer inni funksjonen kalt **animate** i **FTTRv4_GUI**. Først leses fila for sanntidsplotting, *temp_read.csv*. De ulike elementene i fila blir ekstrahert og får egne navn. *Pyplot* fra Matplotlib blir brukt til å plotte dataen. På x-aksen er det klokkeslett og på y-aksen de ulike temperatursensorene. Plottet blir konfigurert med ulike innstillinger for blant annet x- og y-akse, begrensninger på x- og y-akse og navn på aksene. Navn på de ulike sensorene er plassert under grafvinduet for å holde det ryddig. Plottevinduet blir opprettet med FigureCanvasTkAgg fra Matplotlib og tkinter. FuncAnimation animerer funksjonen **animate** med et tastetid på 1 sekund.

8.2.4 Inntastingsfelter

Alle inntastingsfeltene er fra biblioteket tkinter. De blir opprettet når koden starter og får en initiell verdi fra pid.conf-filen. Noen inntastingsfelter er mulig å skrive i, mens andre brukes som indikatorer for ulike verdier. Disse er i kun-lese-modus. Det var nødvendig å bruke inntastingsfelter for indikatorer, fordi vanlig tekstbokser gjorde at tallverdiene havnet oppå hverandre når de oppdaterte seg og gjorde det uryddig. Inntastingsfeltene er gruppert inni hver sin boks for å skille på funksjonaliteten og informasjonen til de.

8.2.5 Programvare

For å kunne oppdatere verdiene i inntastingsfeltene, blir de også opprettet i **animate**-funksjonen. Først blir det lest data enten fra temperaturkoden eller fra pid.conf-filen. Denne dataen blir så satt inn i inntastingsfeltene og blir oppdatert med samme intervall som plottet. I PID-koden blir pådragsverdien skrevet til terminalvinduet, men det var ønskelig å få denne visualisert også i brukergrensesnittet. Måten dette gjøres på er at det blir lest av en fil kalt "u_total.csv", som blir opprettet i PID-koden. Den inneholder kun pådragsverdien, som blir ekstrahert og oppdateres i inntastingsfeltet "Control Value". Filen "pid.conf" blir også lest, og verdiene for settpunkt, auto og manuelt pådrag blir ekstrahert og oppdateres i respektive inntastingsfelter. Det blir også opprettet inntastingsfelter for indikasjon av hvilken modus systemet er i. Det sjekkes om auto er 0 eller 1 og teksten i inntastingsfeltet endrer seg i henhold til dette. Det finnes tre knapper i brukergrensesnittet. En av disse fungerer som en slags bryter, som endrer mellom auto og manuell modus. De andre to er for å lagre parameterverdiene og lagre plottet. Bryteren fungerer som en helt vanlig knapp, bare at den har to moduser: på eller av. Knappen bruker to bilder som den bytter mellom ved trykk: et på-bilde og et av-bilde. I "på-modus" blir auto satt til 1 og inntastingsfeltet for manuelt pådrag settes i "kun-lese-modus", mens i "av-modus" settes auto til 0 og resten av inntastingsfeltene settes i "kun-lese-modus". Parameterverdiene blir også satt når knappen trykkes. Knappen "APPLY" skriver de nye parameterverdiene til "pid.conf". For å lagre sanntidsplottet benyttes knappen "Save Plot". Den kaller på en funksjon kalt **savePlot**. Pyplot fra Matplotlib benyttes for å lagre plottet og funksjonen **init_time_plot** sørger for at det blir opprettet en ny fil hver gang. Filnavnet blir dato og klokkeslett ved trykking av knappen. Under knappen vil en melding dukke opp som forteller brukeren filplasseringen til figuren.

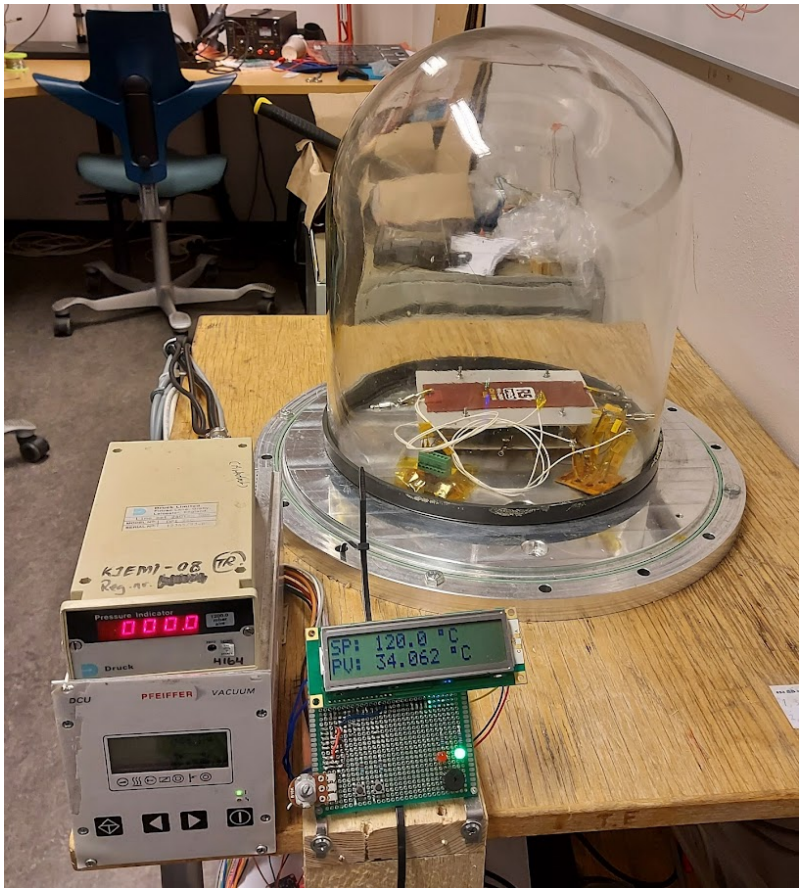
9 Resultat

Resultatet av alt arbeidet er et velfungerende reguleringssystem som fungerer bedre enn den tidligere versjonen. Systemet bruker omtrent 20 minutter på å oppnå 120 grader i vakuum, noe som er betydelig bedre enn det tidligere oppsettet. Innsvingningsforløpet til systemet unngår oversving og holder seg stabilt innenfor det spesifiserte intervallet på ± 5°C. De to nye pådragsorganene er dessuten svært mye bedre enn de gamle. De er mye enklere å montere og de har dessuten bedre varmeutvikling. Det tar kortere tid å varme opp testobjektet med de nye pådragsorganene i forhold til de gamle.

Det ble desverre ikke nok tid til å implementere LQR på det endelige systemet. Alt av grunnarbeid med modellering og simulering er gjort, og resultatet av dette virker svært lovende. De to modellene som ble brukt i simuleringen er noe forskjellige, men de har tilnærmet lik oppførsel og bruker omtrent samme tid som det fysiske oppsettet. Simuleringene med LQR ble godkjent av universitetslektor Torleif Anstensrud.

Transistorkretsen som har blitt utviklet fungerte akkurat slik som planlagt, og gjorde det mulig å effektstyre pådragsorganene svært nøyaktig. Selve BJT-transistoren ble også svært lite varm under bruk.

De to nye brukergrensesnittene som har blitt utviklet har gjort operasjonen av vakuumkammeret mye mer brukervennlig. Det fysiske brukergrensesnittet gjør det mulig for operatør å bytte modus, avlese temperatur og endre verdier for settpunkt og manuelt pådrag rett ved siden av systemet. Det grafiske brukergrensesnittet derimot gir operatør tilgang til litt mer avanserte innstillinger. Dessuten gir sanntidsplottet mer innsikt i systemoppførselen og en visuell indikasjon på ytelse og oppnåelse. Designet av dette brukergrensesnittet ble godkjent som brukervennlig av universitetslektor Cevdet Islek.



Figur 43: Bildet viser det ferdige oppsettet.

10 Diskusjon

Arbeidet som har blitt gjennomført har holdt en god standard og alle problemstillingene Orbit fremmet har blitt adressert, men allikevel kan systemet forbedres. De er dessuten meget fornøyde med arbeidet bachelorgruppa har gjort.

10.1 Videre arbeid

Å innføre en foroverkobling var opprinnelig en arbeidspakke. Denne arbeidspakken ble skrinlagt for å fokusere mer på LQR. Implementasjon av en slik foroverkobling kan være med på å forbedre reguleringssystemet, og vil fungere som et supplement til PID-regulatoren. En har mulighet til å lage foroverkobling enten fra referanse eller fra forstyrrelse. I tillfellet med TVC vil det antageligvis holde med å innføre en foroverkobling fra referanse. Forstyrrelsene til TVC vil være neglisjerbare, ettersom romtemperaturen på taket holder seg relativt konstant. En foroverkobling fra referanse vil være med på å redusere innvirkningen av referanseendring på systemet.

Det ble nevnt tidligere i oppgaven at det ikke ble nok tid til å få implementert LQR. Orbit har dermed fått en god mulighet til å ta med seg denne kunnskapen videre og implementere dette basert på grunnarbeidet som har blitt gjennomført av bachelorgruppa. LQR vil føre til en optimalregulering av prosessen. Professor Damiano Varagnolo mente at den utarbeidede LQR-en som baserer seg på linearisering bør komme som et tillegg til PID-regulatoren. Ettersom modellen har blitt linearisert vil den ikke stemme like godt når man er et stykke unna arbeidspunktet. Tenkt strategi vil dermed være å starte med en PID-regulator for å så få LQR-en til å ta over når prosessverdien er nærmere arbeidspunktet.

For at innsvingningsforløpet fra modelleringen skal stemme bedre med det fysiske oppsettet kan varmetap ut av de ulike gjenstandene tas med i energibalansene. Framgangsmåten for resten av modelleringen følger samme prosedyre.

På det fysiske brukergrensesnittet er det installert en buzzer. Bachelorgruppa nedprioriterte arbeidet på denne. Om Orbit ønsker er det mulig å lage en alarm som går av dersom temperaturen på komponenten blir altfor høy, eller dersom det oppstår en annen feilsituasjon.

Bachelorgruppa gikk også til innkjøp av to analoge temperatursensorer ettersom temperaturen på pådragsorganene overskred avlesningsgrensen til de digitale temperatursensorene. Disse to temperatursensorene ble testet og er implementert i det grafiske brukergrensesnittet. Problemet med disse sensorene var at de ikke var ferdigkalibrert. Gruppa hadde ikke nok tid til å utføre dette.

10.2 Gruppas refleksjon

Gjennomføringen av bachleoroppgaven har gitt gruppa verdifull erfaring i hvordan man går frem for å gjennomføre et større prosjekt. Dette omfatter informasjoninnsamling av relevant teori, design, programmering og uttesting. Gruppa føler at prosessmålene som ble satt ved prosjektstart har blitt nådd. Ved prosjektstart ble arbeidsoppgaver fordelt mellom gruppemedlemmene, men oppgavene endret seg underveis. Sverre endte opp med hovedansvar for maskinvare (oppkobling og lodding). Martin fikk hovedansvaret for implementering av PID-regulator og parameterinnstillingen av denne. Håkon ble brukergrensenitt-ansvarlig og kodet for det meste på disse. Sacit endte opp med å se på mer avanserte reguleringssystemer etter en oppfordring av professor Damiano Varagnolo, og ble dermed ansvarlig for LQR. Selv om alle fikk sitt hovedområde har det vært en del overlapp i arbeidet, og det har vært mye fokus på at bacheloroppgaven er en samarbeidsoppgave der alle skal bidra omtrent like mye.

Orbit NTNU har vært en veldig ålreit organisasjon å skrive for. Dem har verdt veldig imøtekommende og hyggelige. Hvis gruppa hadde spørsmål var de veldig behjelpelige med dette. Organisasjon har også mange medlemmer som smått har bidratt med litt tips og triks.

11 Konklusjon

Bacheloroppgaven vil nå bli oppsummert og det vil bli trekt noen sentrale konklusjoner. Arbeidet som ble gjennomført ble slik som tiltenkt. Det nye fysiske oppsettet fungerer meget bra. Det er mindre kronglete og vaglete enn det tidligere. Dessuten var innføringen av varmematter som pådragsorgan er stor suksess. Det å varme opp testkomponenten til 120°C, noe som før tok rundt 1,5 time tar nå omtrent 20 minutter.

Elektronikken som ble utviklet fungerte også som planlagt. Transistorkretsen takler lasten svært godt, og gjør det mulig å styre effekten ganske nøyaktig. Den vil også kunne takle en utvidelse av antall varmematter. Oppkoblingen av det fysiske brukergrensesnittet fungerer også tilfredstillende godt.

Programvaren utviklet fungerer også ganske bra. PID-regulatoren regulerer temperaturen temmelig godt, og parameterene til den er valgt slik at innsvingningsforløpet ikke vil ha noe oversving. Kommunikasjon mellom PID-en og brukergrensesnittene skjer via en CSV-fil og en CONFIG-fil. Dette er en brukbar løsning som gjør det mulig å både styre og overvåke systemet fra det fysiske og det grafiske brukergrensesnittet. Utformingen av brukergrensenittene, særlig det grafiske er svært brukervennlig.

Gruppa fikk ikke tid til å implementere LQR på det endelige oppsettet, men en grundig modellering og simulering av systemet har blitt gjennomført. Det ble heller ikke nok tid til å få lagt inn en foroverkobling. Dette kan være en mulig fremtidig forbedring Orbit kan undersøke om de ønsker det.

Bachelorgruppa er ganske fornøyd med arbeidet som de har gjort, og mener at alle problemstillingene Orbit NTNU fremmet ved prosjektstart har blitt besvart. Arbeidet innad i gruppa har også fungert svært godt. Gruppa holdt seg rimelig godt innenfor budsjettet. Gruppa er også veldig tilfreds med Orbit NTNU som oppdragsgiver, og mener at de ga gruppa en relevant teknisk utfording som var fornøyelig å løse.

12 Referanser

[1] Pihl, Roger (13.07.21). SNL Selfie.
https://snl.no/selfie

[2] Pedersen, Bjørn (08.11.21). SNL Vakuum.
https://snl.no/vakuum

[3] Wikipedia. (19.05.2022). BJT
https://en.wikipedia.org/wiki/Bipolar_junction_transistor

[4] Wikipedia. (24.01.2022). Pulse-width modulation
https://en.wikipedia.org/wiki/Pulse-width_modulation

[5] Gravdahl, Jan Tommy (12.03.18). SNL PID-regulator.
https://snl.no/PID-regulator

[6] Wikipedia. (19.05.2022). Raspberry Pi.
https://no.wikipedia.org/wiki/Raspberry_Pi

[7] Wikipedia. (18.05.2022). LCD.
https://no.wikipedia.org/wiki/LCD

[8] Wikipedia. (19.05.2022). Analog-til-digital omformer.
https://no.wikipedia.org/wiki/Analog-til-digital_omformer

[9] Wikipedia. (19.05.2022). I^2C.
https://no.wikipedia.org/wiki/I%C2%B2C

[10] Wikipedia. (19.05.2022). Read-only memory.
https://no.wikipedia.org/wiki/Read-only_memory

[11] Birkeland, Roger (27.01.22). SNL CubeSat.
https://snl.no/CubeSat

[12] Wikipedia. (19.05.2022). Orbit NTNU
https://no.wikipedia.org/wiki/Orbit_NTNU

[13] Orbit NTNU. hjemmeside Orbit NTNU.
Hentet 03.02.2022 fra https://www.orbitntnu.com/

[14] Orbit NTNU. (18.01.2022). OV vacuum chamber Guide.
Følgende dokument er ikke offentligjort.

[15] Orbit NTNU. (18.01.2022). FTTRv3 Guide.
Følgende dokument er ikke offentligjort.

[16] RS PRO Silicone Heater Mat
https://no.rs-online.com/web/p/heater-pads/1812061

[17] deGroh III, H. C., Daniels, C. C., Dever, J. A., Miller S. K., Waters, D. L., Finkbeiner, J. R., Dunlap, P. H. and Steinetz, B. M. (2010) Space Environment Effects on Silicone Seal Materials
https://ntrs.nasa.gov/citations/20100029591

[18] Wikipedia. (19.05.2022). Sziklai pair
https://en.wikipedia.org/wiki/Sziklai_pair

[19] Electronic Tutorials
https://www.electronics-tutorials.ws/transistor/tran_2.html

[20] Wikipedia. (19.05.2022). MOSFET
https://en.wikipedia.org/wiki/Power_MOSFET#On-state_resistance

[21] Wikipedia.(19.05.2022). Persistence of vision
https://en.wikipedia.org/wiki/Persistence_of_vision

[22] ElectroPeak. (19.05.2022). DS18B20 digital temperature sensor.
https://electropeak.com/sensor-ds18b20-1-1

[23] Elprocus
https://www.elprocus.com/what-is-a-closed-loop-control-system-its-working/

[24] Wikimedia. (19.05.2022) PID
https://commons.wikimedia.org/wiki/File:PID_en.svg

[25] Mathworks. Optimal Control
https://es.mathworks.com/discovery/optimal-control.html

[26] A Comparative Study Between a Classical and Optimal Controller for a Quadrotor
https://arxiv.org/ftp/arxiv/papers/2009/2009.13175.pdf

[27] Wikipedia. (19.05.2022) Linear–quadratic regulator
https://en.wikipedia.org/wiki/Linear%E2%80%93quadratic_regulator

[28] Store Norske Leksikon. (19.05.2022). Termodynamikk.
https://snl.no/termodynamikk

[29] Store Norske Leksikon. (19.05.2022). Varmekapasitet.
https://snl.no/varmekapasitet

[30] The Efficient Engineer, 2021, 1:49.
https://www.youtube.com/watch?v=FDmYCI_xYlA&list=LL&index=29

[31] Cengel, Y. A. (2002). "Heat transfer: A Practical Approach".
McGraw-Hill.

[32] Wikipedia. (19.05.2022) Joule heating
https://en.wikipedia.org/wiki/Joule_heating

[33] APMonitor. State Space Model.
https://apmonitor.com/pdc/index.php/Main/StateSpaceModel

[34] Bjørvik, K. & Hveem, P. (2014). "Reguleringsteknikk".
Kybernetes forlag.

[35] MathWorks (2022). "System Identification Toolbox".
https://se.mathworks.com/products/sysid.html

[36] Aleksandar Haber. Compute and Simulate Linear Quadratic Regulator (LQR) in MATLAB for Non-zero Set Points
https://aleksandarhaber.com/compute-and-simulate-linear-quadratic-regulator-lqr-in-matlab-for-set-point-tracking/

[37] Anstensrud, T. (2021). "Digitale reguleringssystemer".
Forelesningsnotater

[38] Åström, K.J. og Hägglund,T. PID Controllers: Theory, Design and Tuning 2nd Edition.
https://aiecp.files.wordpress.com/2012/07/1-0-1-k-j-astrom-pid-controllers-theory-design-and-tuning-2ed.pdf

[39] Skogestad, S. & Grimholt, C. (2011).
"The SIMC method for smooth PID controller tuning".
https://folk.ntnu.no/skoge/publications/2012/skogestad-improved-simc-pid/old-submitted/simcpid.pdf

[40] Dessen, F. (2021). "Reguleringsteknikk 2".
Forelesningsnotater

[41] Wikipedia. (18.05.2022). Charge Pump
https://en.wikipedia.org/wiki/Charge_pump

[42] Wikipedia. (18.05.2022). 555 Timer.
https://en.wikipedia.org/wiki/555_timer_IC

13 Appendiks

13.1 A: Oppgavetekst

Oppgaveforslag bacheloroppgave elektroingeniør (BIELEKTRO) i Trondheim, vårsemester 2022

Navn bedrift: **Orbit NTNU**	**Kontaktperson: Mari Linnerud** **Epost: cto@orbitntnu.com** **Telefon/mobil: 90034988**	
Tittel på oppgave: ***Varmereguleringssystem for testing av komponenter i termisk vakuum***		
Hvilke studieretninger passer oppgaven for? (kryss av for alle aktuelle retninger; flervalg er mulig):	**Automatisering og robotikk**	X
	Elektronikk og sensorsystemer	X
	Elkraft og bærekraftig energi	
Er oppgaven reservert for noen bestemte studenter? I så fall skriv navnene på studentene til høyre.		
Er dette en lukket oppgave? Dvs. at sluttrapporten ikke kan publiseres senere fordi den inneholder sensitiv informasjon.	[ ] ja [X] nei [ ] ikke enda bestemt	

Kort beskrivelse av oppgaven med problemstilling.

Orbit NTNU er en teknisk studentorganisasjon som bygger småsatellitter av typen kubesatellitt. Kubesatellitter er små standardiserte satellitter, der den minste størrelsen, 1U, er 10cmx10cmx10cm. Disse kan settes sammen til 2U, 3U og hele veien til 24U. Utviklingstiden til en satellitt blir derfor markant kortere enn konvensjonelle store satellitter fordi systemene på satellitten i større grad er commercial off the shelf components (COTS). Det er derimot ikke sikkert alle komponentene er designet for å brukes i verdensrommet. Man må derfor sørge for at de tåler det miljøet som satellitten vil opplevet i bane rundt jorden.

I hovedsak er det to forhold man ønsker å undersøke når man tester en komponent som skal brukes i en satellitt. Det første man må sørge for er at komponenten ikke utgasser når den blir utsatt for vakuum. Utgassing kan sammenlignes med fordamping. Enkelte plasttyper gasser ut molekyler når det blir utsatt for vakuum, noe som resulterer i at komponenten mister masse. Denne massen kan legge seg på elektronikk, optiske linser og lignende, og potensielt ødelegge annet utstyr på satellitten. Man må derfor utføre utgassingstester for å sikre at den følger gitte standarder. Dette gjøres ved å legge komponenten i et termisk vakuum ved 125 grader celsius i 24 timer.

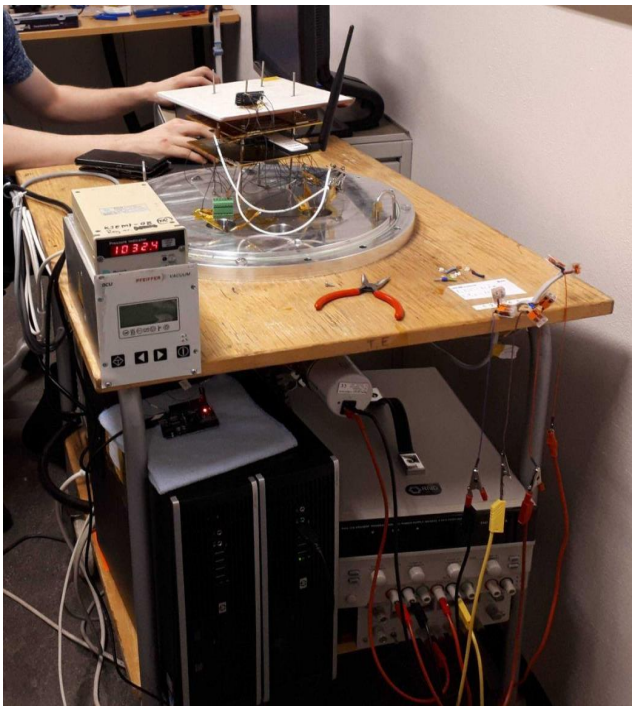
Det andre man ønsker å teste er at hele satellitten er operativ i de temperaturene man kan forvente at den vil oppleve i bane rundt jorden. Det kan variere alt fra -40 grader i skyggesiden av jorden, til +90 grader i solen. Dette testes i et TVAC (thermal vacuum) kammer der formålet er å teste hvordan varmen fordeles i satellitten mens ulike subsystemer er skrudd på.

Orbit NTNU utfører i dag utgassingstester i et egenprodusert vakuumkammer. TVAC utføres i hovedsak hos industrisponsorer, men vi ønsker å kunne reprodusere testene i vårt eget vakuumkammer på kontoret dersom vi finner feil eller svakheter ved testene som vi utførte hos sponsoren.

Det hjemmelagde vakuumkammer oppnår 3e-6 mBar. Vi har et system som vi kaller FTTR (Fancy thermal test rig) som består av to kobberplater, en Raspberry Pi, en arduino og varmetråder. For å varme opp systemet til ønsket temperatur reguleres effekten som blir ført gjennom varmetrådene manuelt. Systemet er tregt, tungvint å operere,

og krever at en person følger nøye med for å passe på at systemet oppnår ønsket temperatur. I bacheloroppgaven ønsker vi gruppen skal se på muligheter for å automatisere reguleringsprosessen. Det er mulig å ta utgangspunkt i eksisterende oppsett, eller de kan utvikle sitt eget. På grunn av økonomiske begrensninger er det ikke mulig å innføre et system for avkjøling. Utstyret som benyttes må være kompatibelt med vakuum dersom det skal være på innsiden av vakuumkammeret. I tillegg må det ha fysisk plass i vakuumkammeret.

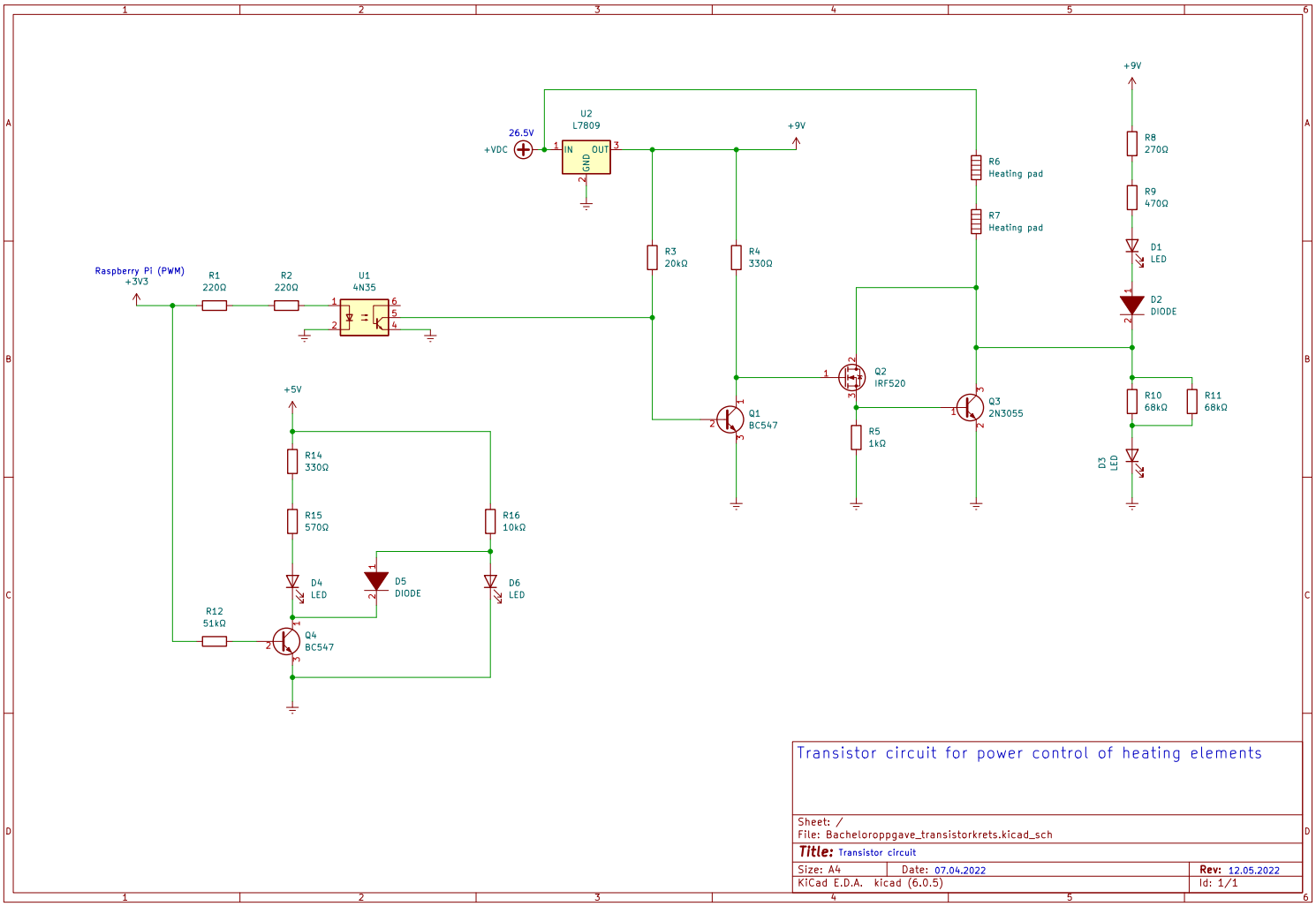
Under er to bilder av vakuumkammeret og et eksempel på en test som ble gjort påsken 2021.

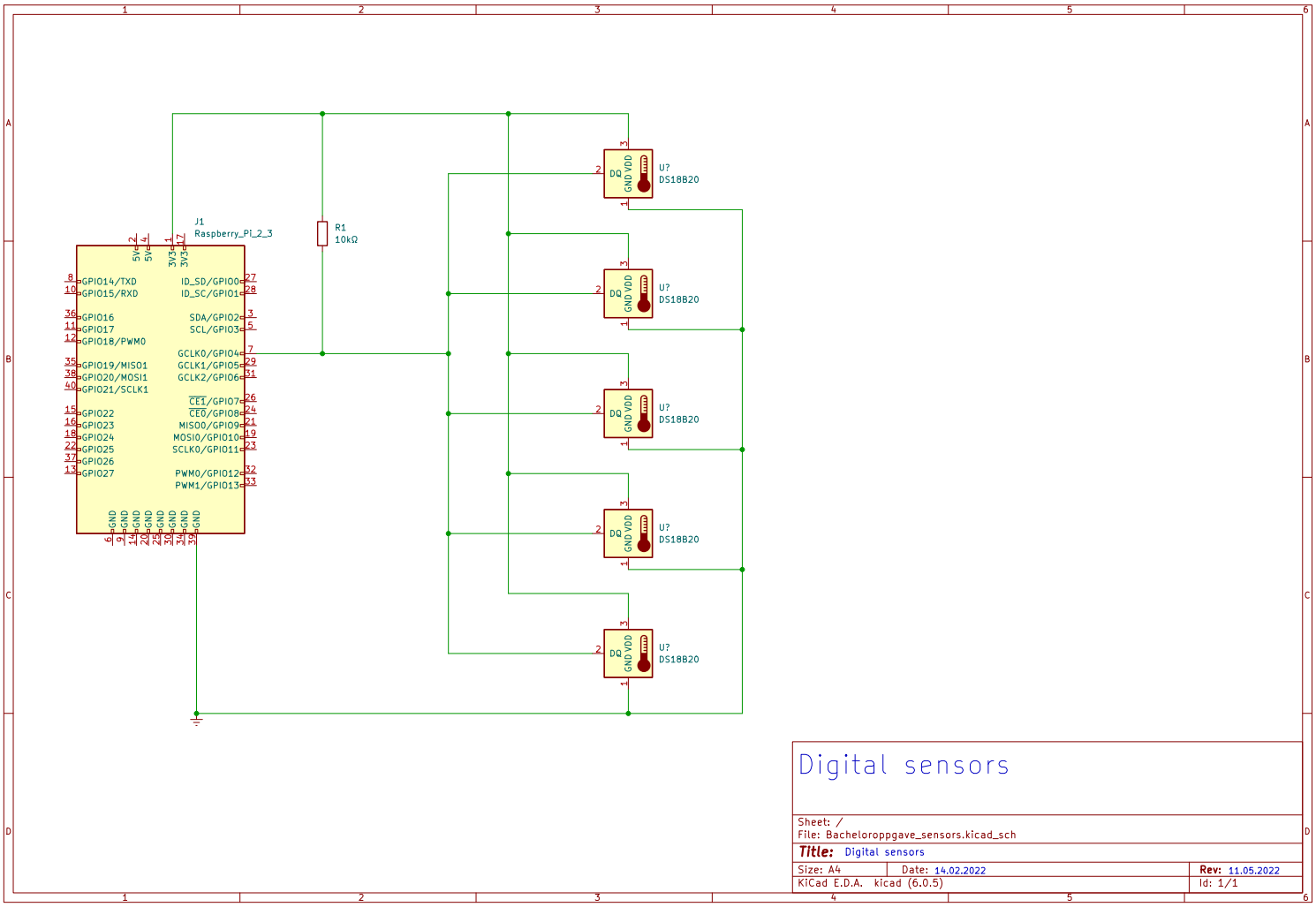


13.2 B: Material- og komponentliste

Material/komponent	Antall	Datablad
Aluminiumsplate	2	Ingen
Gjengestang	4	Ingen
M3 skive	16	Ingen
M3 mutter	24	Ingen
Avstandstykke	4	Ingen
Varmematte 12V 60W	2	RS Pro Silicone Heater Mat datasheet
Raspberry Pi 3A+	1	Raspberry Pi 3 Model A+ documentation
RND 320-KD3305P (PPS)	1	Programmable DC Power Supply documentation
HDMI kabel 10m	1	Ingen
DS18B20 (digital temperatursensor)	5	DS18B20 datasheet
LM35 (analog temperatursensor)	2	LM35 datasheet
4N35 (optocoupler)	1	4N35 datasheet
L7809 (spenningsregulator)	1	L7809 datasheet
Knapp	2	Push-Button datasheet
Buzzer	1	Buzzer datasheet
ADS1115 (ADC)	1	ADS1115 datasheet
PCF8574 (I2C utvidelse)	1	PCF8574 datasheet
LCD-skjerm	1	LCD datasheet
BSS138 (4-kanal-logisk-nivåomformer)	1	BSS138 datasheet
NE555P (555 timer)	1	NE555P datasheet
BC547 (BJT)	3	BC547 datasheet
IRF520 (MOSFET)	1	IRF520 datasheet
2N3055 (BJT)	1	2N3055 datasheet
BC557 (BJT)	2	BC557 datasheet
Potensiometer	1	Potentiometer datasheet
Diode	4	Ukjent
Motstander (ulike verdier)	31	Ukjent
LED	9	Ukjent

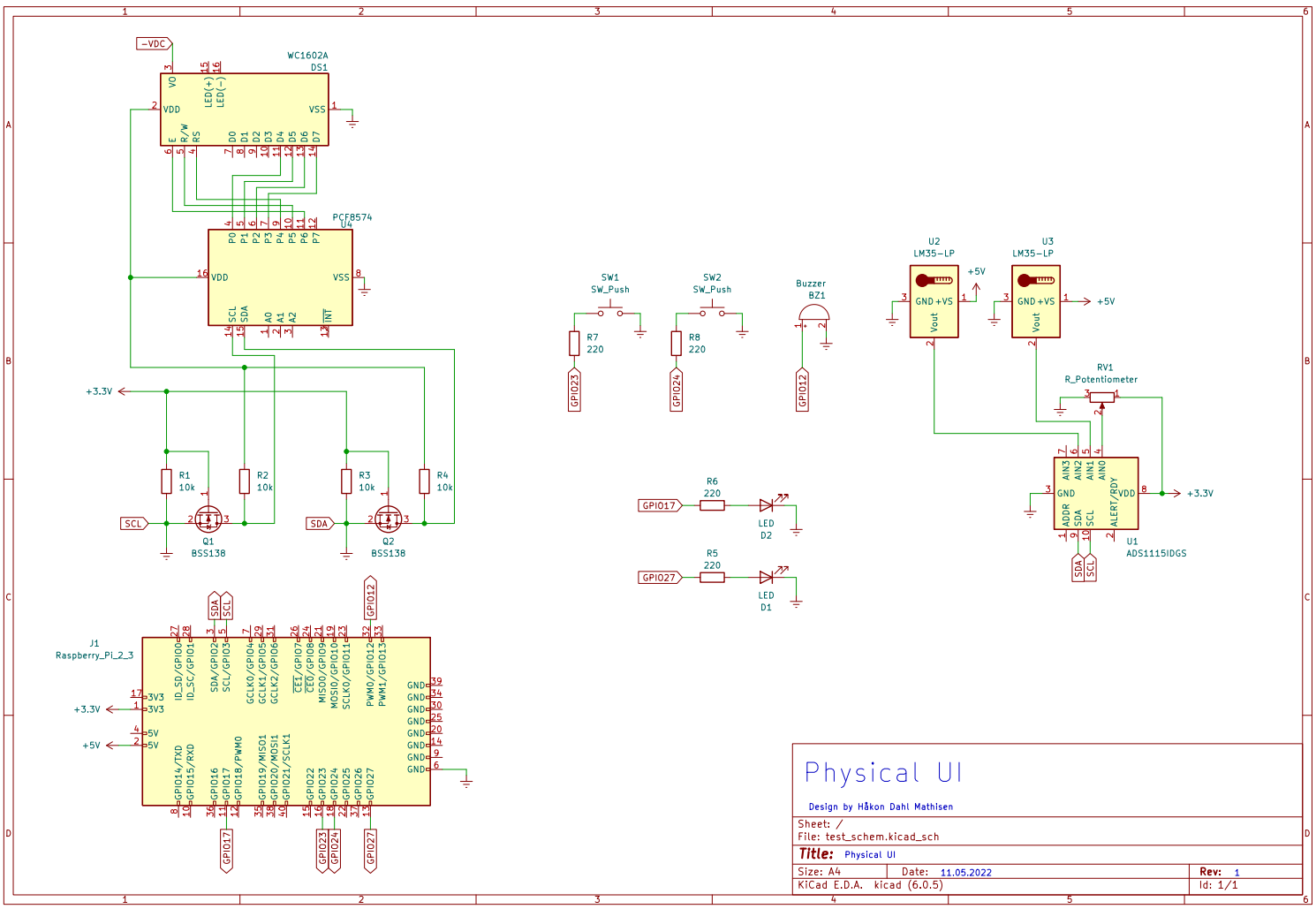
13.3 C: Kretstegninger





Digital sensors

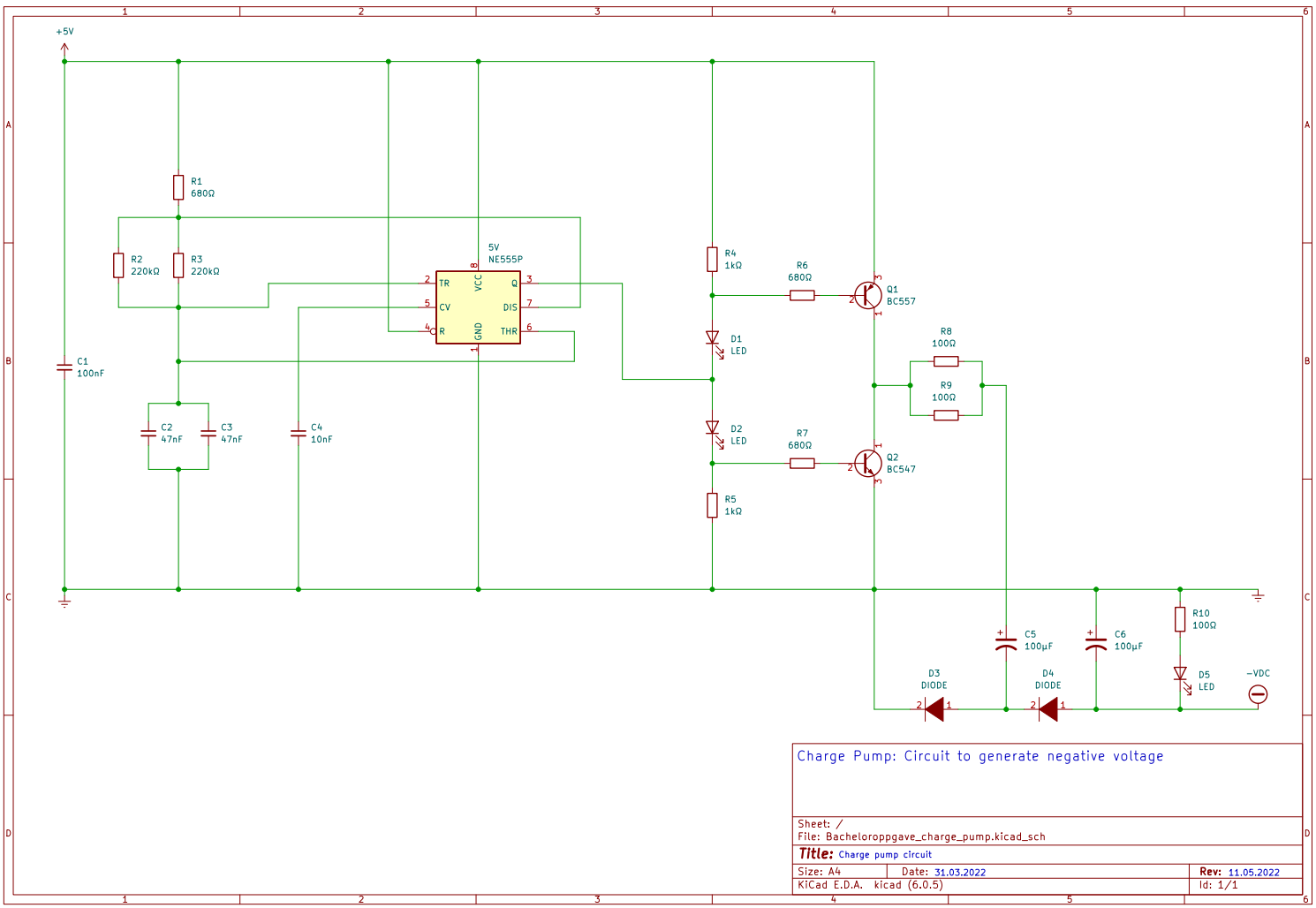
Sheet: /		
File: Bacheloroppgave_sensors.kicad_sch		
Title: Digital sensors		
Size: A4	Date: 14.02.2022	**Rev:** 11.05.2022
KiCad E.D.A. kicad (6.0.5)		Id: 1/1



Physical UI

Design by Håkon Dahl Mathisen

Sheet: /	
File: test_schem.kicad_sch	
Title: Physical UI	
Size: A4	Date: 11.05.2022
KiCad E.D.A. kicad (6.0.5)	**Rev:** 1
	Id: 1/1


+5V
R1
680Ω
R2
220kΩ
R3
220kΩ
5V
NE555P
TR
CV
R
VCC
GND
Q
DIS
THR
C1
100nF
C2
47nF
C3
47nF
C4
10nF
R4
1kΩ
D1
LED
D2
LED
R5
1kΩ
R6
680Ω
R7
680Ω
Q1
BC557
Q2
BC547
R8
100Ω
R9
100Ω
C5
100µF
C6
100µF
D3
DIODE
D4
DIODE
R10
100Ω
D5
LED
-VDC
Charge Pump: Circuit to generate negative voltage
Sheet: /
File: Bacheloroppgave_charge_pump.kicad_sch
Title: Charge pump circuit
Size: A4
Date: 31.03.2022
Rev: 11.05.2022
KiCad E.D.A. kicad (6.0.5)
Id: 1/1

13.4 D: Python kode

Main for PID-regulator og sensoravlesning:

```
import FTTRv4_PID as PID
import FTTRv4_temp as tmp

def main():

    tmp.init_time()
    PID.PID_main()

main()
```

Listing 1: main (PID and sensors)

PID-regulator:

```
import RPi.GPIO as GPIO
import time
import os.path
import FTTRv4_temp as tmp
import csv
import pandas as pd

fieldnames = ["U_total"]

# Parameters
Ts = 1
SP = 120
K_p = 1.1
T_i = 180
T_d = 13.5
N = 10

T_t = 0
ManVal = 0
Tr_gain = 0
U_total = 0
PV = [0,0]
e = [0, 0]
U_i = [0, 0]
U_d = [0, 0]

# user input
#Auto = int(input("Enter Auto (1) or Manual (0): "))
Auto = 0
PWM_pin = 13 # PWM pin on Raspberry Pi


with open('u_total.csv', 'w') as p:
    csv_writer = csv.DictWriter(p, fieldnames=fieldnames)
    csv_writer.writeheader()

# Setup of the PWM pin on the Raspberry Pi
def setup():
    global pwm
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(PWM_pin, GPIO.OUT)
    GPIO.output(PWM_pin, GPIO.LOW)
    pwm = GPIO.PWM(PWM_pin, 100) # Set Frequency to 100 Hz
    pwm.start(0) # Set the starting Duty Cycle

# Destroy PWM pin
def destroy():
    pwm.stop()
    GPIO.output(PWM_pin, GPIO.LOW)
    GPIO.cleanup()


def createConfig():
    file_exists = os.path.exists('pid.conf')
    if file_exists == True:
```

```
56          os.remove("pid.conf")
57      else:
58          pass
59      with open ('pid.conf', 'w') as f:
60          f.write('%s,%s,%s,%s,%s,%s'%(SP,K_p,T_i,T_d,Auto,ManVal))
61
62  def readConfig():
63      global SP, K_p, T_i, T_d, Auto, ManVal
64      with open ('pid.conf', 'r+') as f:
65          config = f.readline().split(',')
66          SP = float(config[0])
67          K_p = float(config[1])
68          T_i = float(config[2])
69          T_d = float(config[3])
70          Auto = int(config[4])
71          ManVal = float(config[5])
72
73  # PID-controller
74  def FTTR_PID(Ts, SP, PV, K_p, T_i, T_d, T_t, Tr_gain, U_total):
75      # Check for zero division
76      if(T_i > 0):
77          alpha = Ts/T_i
78      else:
79          alpha = 0
80
81      if(T_t > 0):
82          gamma = Ts/T_t
83      else:
84          gamma = 0
85
86      if((T_d + Ts*N)>0):
87          beta = T_d/(T_d+Ts*N)
88      else:
89          beta = 0
90
91      PV[0] = tmp.read_temp0()
92      print("Sensor0: " + str(PV[0]))
93
94      # Calculate error from setpoint
95      e[0] = SP - PV[0]
96
97      # Proportional control
98      U_p = K_p * e[0]
99
100     # Integral control with anti-windup (back calculation)
101     if T_i == 0:
102         U_i[0] = 0
103     else:
104         U_i[0] = U_i[1] + (K_p * alpha * e[0]) + gamma*(Tr_gain - U_total)
105
106     # Clamp I-term
107     if(U_i[0]>100):
108         U_i[0] = 100
109     elif(U_i[0]<0):
110         U_i[0] = 0
111
112     # Derivative control
113     U_d[0] = beta*U_d[1] - K_p*(T_d/Ts)*(1-beta)*(PV[0]-PV[1])
114
115     # Total control
116     U_total = U_p + U_i[0] + U_d[0]
117
118     # Clamp total control
119     if(U_total>100):
120         U_total = 100
121     elif(U_total<0):
122         U_total = 0
123
124     # Update values
125     e[1] = e[0]
126     PV[1] = PV[0]
127     U_i[1] = U_i[0]
128     U_d[1] = U_d[0]
```

```
129
130     print(round(U_total,1))
131
132     pwm.ChangeDutyCycle(U_total)
133
134     # Samplingtime
135     time.sleep(Ts)
136
137     with open('u_total.csv', 'a') as p:
138         csv_writer = csv.DictWriter(p, fieldnames=fieldnames)
139         U_total = str(round(U_total, 2))
140         info = {
141             "U_total": U_total
142         }
143         csv_writer.writerow(info)
144         p.close()
145
146
147     return U_total
148
149
150 def PID_loop():
151     readConfig()
152     FTTR_PID(Ts, SP, PV, K_p, T_i, T_d, T_t, Tr_gain, U_total)
153
154 def ManVal_loop():
155     readConfig()
156     man_output = ManVal
157     print(man_output)
158     temp_read = tmp.read_temp0()
159     print("Sensor0: " + str(temp_read))
160     pwm.ChangeDutyCycle(man_output)
161     time.sleep(Ts)
162
163 def PID_main():
164     createConfig()
165     tmp.create_tmpFile()
166     tmp.create_tmpFile_live()
167     setup()
168     try:
169         while True:
170             #tmp.read_temp0()
171             tmp.write_tmp()
172             if Auto == 1:
173                 PID_loop()
174             elif Auto == 0:
175                 ManVal_loop()
176     except KeyboardInterrupt:
177         destroy()
```

Listing 2: PID-kode

Sensoravlesning:

```
1  import os
2  import glob
3  from time import sleep
4  import csv
5  import datetime as dt
6
7
8  os.system('modprobe w1-gpio')
9  os.system('modprobe w1-therm')
10
11 base_dir = '/sys/bus/w1/devices/'
12
13 temp_folder = "temp/"
14 temp_filename = ""
15 temp_filepath = ""
16
17
18 fieldnames = ["x", "dtemp0", "dtemp1", "dtemp2", "dtemp3", "dtemp4"]
19
```

```
20
21 def init_time():
22     global ref_time
23     global temp_filename
24     global temp_folder
25     global temp_filepath
26
27
28     now = dt.datetime.now()
29     t = now.strftime("%H:%M:%S")
30     (h, m, s) = t.split(':')
31     ref_time = int(h) * 3600 + int(m) * 60 + int(s)
32
33     temp_filename = now.strftime("%m_%d_%Y-%H:%M")
34     temp_filepath = temp_folder + temp_filename
35
36
37
38 def read_temp_raw(n):
39     device_folder = glob.glob(base_dir + '28*')[n]
40     device_file = device_folder + '/w1_slave'
41     f = open(device_file, 'r')
42     lines = f.readlines()
43     f.close()
44     return lines
45
46
47 def convert_temp(n):
48     lines = read_temp_raw(n)
49
50     while lines[0].strip()[-3:] != 'YES':
51         lines = read_temp_raw(n)
52
53     equals_pos = lines[1].find('t=')
54
55     if equals_pos != -1:
56         temp_string = lines[1][equals_pos+2:]
57         temp_c = float(temp_string) / 1000.0
58
59         if temp_c > 2048:
60             temp_c = temp_c - 4096
61
62         return temp_c
63
64
65 def read_temp():
66     temps = []
67
68     for i in range (0, 5):
69         temp = convert_temp(i)
70         temps.append(temp)
71
72     return temps
73
74 def read_temp0():
75     temps0 = read_temp()
76
77     return temps0[0]
78
79
80 def create_tmpFile_live():
81     with open('temp_read.csv', 'w') as live_csv:
82         csv_writer = csv.DictWriter(live_csv, fieldnames=fieldnames)
83         csv_writer.writeheader()
84
85 def create_tmpFile():
86     with open(f'{temp_filepath}.csv', 'w') as data_csv:
87         csv_writer = csv.DictWriter(data_csv, fieldnames=fieldnames)
88         csv_writer.writeheader()
89
90 def write_tmp():
91     x = dt.datetime.now().strftime('%H:%M:%S')
92
```

```
    temps = read_temp()

    with open(f'{temp_filepath}.csv', 'a') as data_csv:
        csv_writer = csv.DictWriter(data_csv, fieldnames=fieldnames)

        info = {
            "x": x,
            "dtemp0": temps[0],
            "dtemp1": temps[1],
            "dtemp2": temps[2],
            "dtemp3": temps[3],
            "dtemp4": temps[4]
        }
        csv_writer.writerow(info)
        data_csv.close()

        x = dt.datetime.now().strftime('%H:%M:%S')

    with open('temp_read.csv', 'a') as live_csv:
        csv_writer = csv.DictWriter(live_csv, fieldnames=fieldnames)

        info = {
            "x": x,
            "dtemp0": temps[0],
            "dtemp1": temps[1],
            "dtemp2": temps[2],
            "dtemp3": temps[3],
            "dtemp4": temps[4]
        }
        csv_writer.writerow(info)
        data_csv.close()

        x = dt.datetime.now().strftime('%H:%M:%S')


    sleep(0.1)
```

Listing 3: Sensoravlesning-kode

Det grafiske brukergrensesnittet:

```
import csv
import os
import time
import datetime as dt
import tkinter as tk
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.animation import FuncAnimation
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

#---Import FTTRv4 files---
import FTTRv4_temp as tmp
import FTTRv4_PID as PID

#------------ADC libraries-------------
import board
import busio
import adafruit_ads1x15.ads1115 as ADS
from adafruit_ads1x15.analog_in import AnalogIn
from adafruit_ads1x15.ads1115 import Mode


#-----------Initialize ADC------------
i2c = busio.I2C(board.SCL, board.SDA)
ads = ADS.ADS1115(i2c)
ads.mode = Mode.CONTINUOUS

#---------Degree symbol----------
degree_sign = u'\N{DEGREE SIGN}'

#---Initial values---
```

```
sp = 0
kp = 0
ti = 0
td = 0
auto = 0
man = 0
is_on = True


plot_folder = "plot/"
plot_filename = ""
plot_filepath = ""


#-----Set up main GUI code-----
root = tk.Tk() #Initialize tkinter
root.title("FTTRv4 GUI") #Title
root.configure(background = 'light grey') #Window background
root.geometry("1150x700") # Window size
plt.style.use('fivethirtyeight')

#------Function for plot filelocation------
def init_time_plot():
    global ref_time
    global plot_folder
    global plot_filename
    global plot_filepath

    now = dt.datetime.now()
    t = now.strftime("%H:%M:%S")
    (h, m, s) = t.split(':')
    ref_time = int(h) * 3600 + int(m) * 60 + int(s)

    plot_filename = now.strftime("%m_%d_%Y-%H:%M")
    plot_filepath = plot_folder + plot_filename


#--------Save plot function--------
def savePlot():
    init_time_plot()
    plt.savefig(f'{plot_filepath}.png')
    saved = tk.Label(root, text='File location:\n/home/pi/FTTRv4/plot', font = ('
    calibre', 10))
    saved.place(x=730, y=620)


#---Plot function to animate---
def animate(i):

    #-----Reads csv file & collecting data-----
    data = pd.read_csv('temp_read.csv')
    x = data["x"]
    dtemp0 = data["dtemp0"]
    dtemp1 = data["dtemp1"]
    dtemp2 = data["dtemp2"]
    dtemp3 = data["dtemp3"]
    dtemp4 = data["dtemp4"]

    plt.cla()

    plt.plot(x, dtemp0, linewidth = 1.5, label='Sensor d0', color = '#4876FF')
    plt.plot(x, dtemp1, linewidth = 1.5, label='Sensor d1', color = '#EE0000')
    plt.plot(x, dtemp2, linewidth = 1.5, label='Sensor d2', color = 'orange')
    plt.plot(x, dtemp3, linewidth = 1.5, label='Sensor d3', color = '#008B45')
    plt.plot(x, dtemp4, linewidth = 1.5, label='Sensor d4', color = '#708090')

    plt.ylim([0, 150])
    plt.xlabel("Time [hh:mm:ss]", fontsize=10)
    plt.ylabel("Temperature " + "[" + degree_sign + "C]", fontsize=10)
    plt.xticks(rotation=45, ha='right', fontsize=8)
    plt.xticks(np.arange(0, len(x)+1, 70))
    plt.yticks(fontsize=10)
    plt.tight_layout()
```

```
105
106
107
108
109     #--------ADC reading---------
110     #--Optional for analog temp reading--
111     #chan0 = AnalogIn(ads, ADS.P0)
112     #chan1 = AnalogIn(ads, ADS.P1)
113     #S1 = chan0.value
114     #V1 = chan0.voltage
115     #atemp0 = V1 / (11/1000)
116     #atemp0 = float(round(atemp0, 1))
117     #S2 = chan1.value
118     #V2 = chan1.voltage
119     #atemp1 = V2 / (11/1000)
120     #atemp1 = float(round(atemp1, 1))
121
122
123     #----Read-only entry for control value updating----
124     temp0 = tmp.read_temp0()
125     temp0 = str(round(temp0, 2))
126     temp = tk.Entry(root, width = 7)
127     temp.insert(0, temp0)
128     temp.config(state='readonly')
129     temp.place(x = 970, y = 415)
130
131
132     #--------Reads u_total file---------
133     with open('u_total.csv', 'r') as p:
134         U_total = p.readlines()[-1]
135     #---Read only entry for U_total updating---
136     control = tk.Entry(root, width = 7)
137     control.insert(0, U_total)
138     control.config(state='readonly')
139     control.place(x = 970, y = 455)
140
141     #----Read-only entry for SP updating----
142     with open ('pid.conf', 'r+') as g:
143         conf = g.readline().split(',')
144         SP = float(conf[0])
145         auto = int(conf[4])
146         man = float(conf[5])
147     S_P_ = tk.Entry(root, width=7)
148     S_P_.insert(0, SP)
149     S_P_.config(state='readonly')
150     S_P_.place(x = 970, y = 495)
151
152     #-----Read-only entry for A0 and A1 updating-----
153     manVal = tk.Entry(root, width = 7)
154     manVal.insert(0, man)
155     manVal.config(state='readonly')
156     manVal.place(x = 970, y = 535)
157
158     A1 = tk.Entry(root, width = 7)
159     A1.insert(0, atemp1)
160     A1.config(state='readonly')
161     A1.place(x = 970, y = 575)
162
163
164     if(auto == 1):
165         modeI_ = tk.Entry(root, width = 15)
166         modeI_.insert(0, "     Auto Mode Set")
167         modeI_.config(state='readonly')
168         modeI_.place(x = 905, y = 373)
169
170     else:
171         modeI_ = tk.Entry(root, width = 15)
172         modeI_.insert(0, "   Manual Mode Set")
173         modeI_.config(state='readonly')
174         modeI_.place(x = 905, y = 373)
175
176
177 #----------------Plot window in GUI----------------
```

```
canvas = FigureCanvasTkAgg(plt.gcf(), master=root)
canvas.get_tk_widget().place(x = 10, y = 10, width = 840, height = 555)
canvas.draw()

#------------------Animate function------------------
ani = FuncAnimation(plt.gcf(), animate, interval=1000)

#----Auto/Manual switch function-----
def switch():
    global sp, kp, ti, td, auto, man, is_on
    if  is_on:
        auto = 0
        MA.config(image = off)
        SP_ent.config(state='readonly')
        kp_ent.config(state='readonly')
        ti_ent.config(state='readonly')
        td_ent.config(state='readonly')
        man_ent.config(state='normal')
        modeA_ = tk.Entry(root, width=8)
        modeA_.insert(0, "Manual")
        modeA_.config(state='readonly')
        modeA_.place(x = 940, y = 10)

        is_on = False
    else:
        auto = 1
        MA.config(image = on)
        SP_ent.config(state='normal')
        kp_ent.config(state='normal')
        ti_ent.config(state='normal')
        td_ent.config(state='normal')
        man_ent.config(state='readonly')
        modeM_ = tk.Entry(root, width=8)
        modeM_.insert(0, "Auto")
        modeM_.config(state='readonly')
        modeM_.place(x = 940, y = 10)

        is_on = True

    #-----Gets values from input fields-----
    sp = SP_ent.get()
    sp = float(sp)
    kp = kp_ent.get()
    kp = float(kp)
    ti = ti_ent.get()
    ti = float(ti)
    td = td_ent.get()
    td = float(td)

    #-----Gets value from input field-----
    man = man_ent.get()
    man = float(man)

#-----------Writes the regulator values to file-------------
    with open ('pid.conf', 'w') as f:
        f.write('%s,%s,%s,%s,%s,%s'%(sp,kp,ti,td,auto,man))



#------Setting regulator values------
def SetRegVals():
    global sp, kp, ti, td, auto, man
    #-----Gets values from input fields-----
    sp = SP_ent.get()
    sp = float(sp)
    kp = kp_ent.get()
    kp = float(kp)
    ti = ti_ent.get()
    ti = float(ti)
    td = td_ent.get()
    td = float(td)
    man = man_ent.get()
    man = float(man)
```

```
251
252 #----------Writes the regulator values to file-------------
253     with open ('pid.conf', 'w') as f:
254         f.write('%s,%s,%s,%s,%s,%s'%(sp,kp,ti,td,auto,man))
255
256     #-----Entry for SP updating-----
257     S_P_ = tk.Entry(root, width=7)
258     S_P_.insert(0, sp)
259     S_P_.config(state='readonly')
260     S_P_.place(x = 970, y = 495)
261
262
263 #-------Reads config file--------
264 with open ('pid.conf', 'r+') as g:
265     conf = g.readline().split(',')
266     SP = float(conf[0])
267     KP = float(conf[1])
268     TI = float(conf[2])
269     TD = float(conf[3])
270     man = float(conf[5])
271
272 #---------Create frames----------
273 frame1 = tk.Frame(root, width=230, height=290, highlightbackground='grey',
        highlightthickness=1)
274 frame1.place(x=860, y=80)
275 frame2 = tk.Frame(root, width=230, height=220, highlightbackground='grey',
        highlightthickness=1)
276 frame2.place(x=860, y=400)
277
278 #----on/off image files for switch----
279 on = tk.PhotoImage(file = "on.png")
280 off = tk.PhotoImage(file = "off.png")
281
282 #-------Create buttons-------
283 root.update()
284 MA = tk.Button(root, image = off, bd = 0, command = lambda: switch())
285 MA.place(x = 930, y = 30)
286
287 root.update()
288 S = tk.Button(root, text = "Save plot", font = ('calibri', 12), command = lambda:
        savePlot())
289 S.place(x = 730, y = 590, width=120, heigh=31)
290
291 root.update()
292 SV = tk.Button(root, text = "APPLY", font = ('calibri', 12), command = lambda:
        SetRegVals())
293 SV.place(x = 970, y = 330, width=70, height=30)
294
295 #-------Create input fields--------
296 root.update()
297 modeI_ = tk.Entry(root, width = 15)
298 modeI_.config(state='readonly')
299 modeI_.place(x = 905, y = 373)
300
301 root.update()
302 SP_label = tk.Label(root, text = 'Setpoint:', font = ('calibre', 10))
303 SP_label.place(x = 894, y = 90)
304 SP_ent = tk.Entry(root, width=7)
305 SP_ent.insert(0, SP)
306 SP_ent.place(x = 970, y = 90)
307
308 root.update()
309 kp_label = tk.Label(root, text = 'Proportional\n             gain:', font = ('
        calibre', 10))
310 kp_label.place(x = 868, y = 130)
311 kp_ent = tk.Entry(root, width=7)
312 kp_ent.insert(0, KP)
313 kp_ent.place(x = 970, y = 140)
314
315 root.update()
316 ti_label = tk.Label(root, text = 'Integral\n     time:', font = ('calibre', 10))
317 ti_label.place(x = 897, y = 180)
318 ti_ent = tk.Entry(root, width=7)
```

```
ti_ent.insert(0, TI)
ti_ent.place(x = 970, y = 190)

root.update()
td_label = tk.Label(root, text = 'Derivative\n         time:', font = ('calibre', 10))
td_label.place(x = 881, y = 230)
td_ent = tk.Entry(root, width=7)
td_ent.insert(0, TD)
td_ent.place(x = 970, y = 240)

root.update()
man_label = tk.Label(root, text = 'Manual\n   value:', font = ('calibre', 10))
man_label.place(x = 900, y = 280)
man_ent = tk.Entry(root, width=7)
man_ent.insert(0, man)
man_ent.place(x = 970, y = 290)

#-------Process value entry--------
root.update()
temp_label = tk.Label(root, text = 'Process\n     value: ', font = ('calibre', 10))
temp_label.place(x = 890, y = 405)
temp = tk.Entry(root, width = 7)
temp.config(state='readonly')
temp.place(x = 970, y = 415)

#-------Control value entry---------
root.update()
control_label = tk.Label(root, text = 'Control\n    value: ', font = ('calibre', 10))
control_label.place(x = 893, y = 445)
control = tk.Entry(root, width = 7)
control.config(state='readonly')
control.place(x = 970, y = 455)

#-------Creates SP entry---------
root.update()
S_P_label = tk.Label(root, text = 'Setpoint:', font = ('calibre', 10))
S_P_label.place(x = 891, y = 495)
S_P_ = tk.Entry(root, width=7)
S_P_.insert(0, SP)
S_P_.config(state='readonly')
S_P_.place(x = 970, y = 495)

#------Read-only manual value entry-------
root.update()
manVal_label = tk.Label(root, text = 'Manual\n    value: ', font = ('calibre', 10))
manVal_label.place(x = 892, y = 525)
manVal = tk.Entry(root, width = 7)
manVal.insert(0, man)
manVal.config(state='readonly')
manVal.place(x = 970, y = 535)

#-------Analog sensor entry--------
root.update()
A1_label = tk.Label(root, text = 'Analog\nsensor 1:', font = ('calibre', 10))
A1_label.place(x = 889, y = 565)
A1 = tk.Entry(root, width = 7)
A1.config(state='readonly')
A1.place(x = 970, y = 575)

#--------Labels for graph lines--------
sensord0_c = tk.Label(root, text = 'Sensor d0', font = ('calibre', 10, 'bold'), fg = '#4876FF')
sensord0_c.place(x = 20, y = 580)
sensord1_c = tk.Label(root, text = 'Sensor d1', font = ('calibre', 10, 'bold'), fg = '#EE0000')
sensord1_c.place(x = 120, y = 580)
sensord2_c = tk.Label(root, text = 'Sensor d2', font = ('calibre', 10, 'bold'), fg = 'orange')
sensord2_c.place(x = 220, y = 580)
sensord3_c = tk.Label(root, text = 'Sensor d3', font = ('calibre', 10, 'bold'), fg = '#008B45')
```

```
sensord3_c.place(x = 320, y = 580)
sensord4_c = tk.Label(root, text = 'Sensor d4', font = ('calibre', 10, 'bold'), fg = '#708090')
sensord4_c.place(x = 420, y = 580)

#---Mainloop---
root.mainloop()
```

Listing 4: Det grafiske brukergrensenittet (GUI)

Det fysiske brukergrensesnittet:

```
import time
import RPi.GPIO as GPIO
from RPLCD import i2c

#---ADC-libraries---
import board
import busio
import adafruit_ads1x15.ads1115 as ADS
from adafruit_ads1x15.analog_in import AnalogIn
from adafruit_ads1x15.ads1115 import Mode

#----Import _temp code----
import FTTRv4_temp as tmp


#-----------Initialize I2C-----------
I2C = busio.I2C(board.SCL, board.SDA)
ads = ADS.ADS1115(I2C)
ads.mode = Mode.CONTINUOUS

#--Initial values--
SP = 0
Kp = 0
Ti = 0
Td = 0
auto = 0
man = 0
ManVal = 0

#---------Degree symbol---------
degree_sign = u'\N{DEGREE SIGN}'

#---Setup buttons and leds---
GPIO.setmode(GPIO.BCM)
GPIO.setup(23, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.add_event_detect(23, GPIO.RISING, bouncetime=150)
GPIO.setup(24, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.add_event_detect(24, GPIO.RISING, bouncetime=150)
GPIO.setup(17, GPIO.OUT)
GPIO.setup(27, GPIO.OUT)
GPIO.setup(12, GPIO.OUT)

#---Constants to initialize LCD---
lcdmode = 'i2c'
cols = 20
rows = 4
charmap = 'A00'
i2c_expander = 'PCF8574'

#--LCD I2C address and port--
address = 0x27
port = 1

#--------Initialize the LCD--------
lcd = i2c.CharLCD(i2c_expander, address, port=port, charmap=charmap,
                  cols=cols, rows=rows)

#---The first line on LCD in long string-mode---
framebuffer = [
    'Orbit NTNU',
    '',
```

```
]

#------Function for writing long string to LCD------
def write_to_lcd(lcd, framebuffer, num_cols):
    lcd.home()
    for row in framebuffer:
        lcd.write_string(row.ljust(num_cols)[:num_cols])
        lcd.write_string('\r\n')

#-------Function for looping string on LCD---------
def loop_string(string, lcd, framebuffer, row, num_cols, delay=0.1):
    padding = ' ' * num_cols
    s = padding + string + padding
    for i in range(len(s)- num_cols + 1):
        framebuffer[row] = s[i:i+num_cols]
        write_to_lcd(lcd, framebuffer, num_cols)
        time.sleep(delay)

#---The long string---
long_string = 'Like and subscribe or I will delete your Minecraft account'

#-----Reads the config file-----
def readConfig():
    global SP, Kp, Ti, Td, auto, man
    with open ('pid.conf', 'r+') as g:
        conf = g.readline().split(',')
        SP = float(conf[0])
        Kp = float(conf[1])
        Ti = float(conf[2])
        Td = float(conf[3])
        auto = int(conf[4])
        man = float(conf[5])

#---Auto mode---
def auto_mode():
    global SP, Kp, Ti, Td, auto, man
    isPressed1 = False
    while(isPressed1 == False):
        chan1 = AnalogIn(ads, ADS.P1)
        V1 = chan1.voltage
        sp = (V1*121.5)/3.3
        sp = str(round(sp, 0))
        lcd.clear()
        lcd.write_string("SP: " + sp + " " + degree_sign + "C")
        time.sleep(0.1)
        if(GPIO.event_detected(24)):
            isPressed1 = True
            GPIO.output(17, False)
            time.sleep(0.1)
            GPIO.output(17, True)
            time.sleep(0.1)
            GPIO.output(17, False)
            time.sleep(0.1)
            GPIO.output(17, True)
            time.sleep(0.1)
            GPIO.output(17, False)
            time.sleep(0.1)
            readConfig()
            with open ('pid.conf', 'w') as f:
                f.write('%s,%s,%s,%s,%s,%s'%(sp,Kp,Ti,Td,auto,man))
            lcd.clear()
            lcd.cursor_pos = (0, 0)
            lcd.write_string("SP set")
            time.sleep(0.5)
            showAll_A()

        else:
            isPressed1 = False

#---Read-only auto mode---
def showAll_A():
    global SP, Kp, Ti, Td, auto, man
    isPressed1 = False
```

```
135     isPressed2 = False
136     while (isPressed1 == False):
137         readConfig()
138         if(auto == 0):
139             showAll_M()
140         else:
141             pass
142         GPIO.output(17, True)
143         GPIO.output(27, False)
144         temp0 = tmp.read_temp0()
145         temp0 = str(temp0)
146         SP = str(SP)
147         lcd.clear()
148         lcd.cursor_pos = (0, 0)
149         lcd.write_string("SP: " + SP + " " + degree_sign + "C")
150         lcd.cursor_pos = (1, 0)
151         lcd.write_string("PV: " + temp0 + " " + degree_sign + "C")
152         time.sleep(0.5)
153 
154         if(GPIO.event_detected(23)):
155             isPressed1 = True
156             time.sleep(0.5)
157             auto_mode()
158         else:
159             isPressed1 = False
160 
161         if(GPIO.event_detected(24)):
162             isPressed2 = True
163             GPIO.output(17, False)
164             GPIO.output(27, False)
165             time.sleep(0.2)
166             GPIO.output(27, True)
167             time.sleep(0.2)
168             GPIO.output(27, False)
169             time.sleep(0.2)
170             GPIO.output(27, True)
171             time.sleep(0.2)
172             GPIO.output(27, False)
173             readConfig()
174             auto = 0
175             with open ('pid.conf', 'w') as f:
176                 f.write('%s,%s,%s,%s,%s,%s'%(SP,Kp,Ti,Td,auto,man))
177             lcd.clear()
178             lcd.cursor_pos = (0, 0)
179             lcd.write_string("Manual mode set")
180             time.sleep(0.5)
181             showAll_M()
182         else:
183             isPressed2 = False
184 
185 #---Manual mode---
186 def man_mode():
187     global SP, Kp, Ti, Td, auto, man
188     isPressed3 = False
189     while(isPressed3 == False):
190         chan1 = AnalogIn(ads, ADS.P1)
191         V1 = chan1.voltage
192         ManVal = (V1*101.5)/3.3
193         ManVal = str(round(ManVal, 0))
194         lcd.clear()
195         lcd.write_string("ManVal: " + ManVal + "%")
196         time.sleep(0.1)
197         if(GPIO.event_detected(24)):
198             isPressed3 = True
199             GPIO.output(27, False)
200             time.sleep(0.1)
201             GPIO.output(27, True)
202             time.sleep(0.1)
203             GPIO.output(27, False)
204             time.sleep(0.1)
205             GPIO.output(27, True)
206             time.sleep(0.1)
207             GPIO.output(27, False)
```

```
208             readConfig()
209             with open ('pid.conf', 'w') as f:
210                 f.write('%s,%s,%s,%s,%s,%s'%(SP,Kp,Ti,Td,auto,ManVal))
211             lcd.clear()
212             lcd.cursor_pos = (0, 0)
213             lcd.write_string("Manual value set")
214             time.sleep(0.5)
215             showAll_M()
216         else:
217             isPressed3 = False
218 
219 #---Read-only manual mode---
220 def showAll_M():
221     global SP, Kp, Ti, Td, auto, man
222     isPressed4 = False
223     isPressed5 = False
224     while isPressed4 == False:
225         readConfig()
226         if(auto == 1):
227             showAll_A()
228         else:
229             pass
230         GPIO.output(27, True)
231         GPIO.output(17, False)
232         temp0 = tmp.read_temp0()
233         temp0 = str(temp0)
234         man = str(man)
235         lcd.clear()
236         lcd.cursor_pos = (0, 0)
237         lcd.write_string("ManVal: " + man + "%")
238         lcd.cursor_pos = (1, 0)
239         lcd.write_string("PV: " + temp0 + " " + degree_sign + "C")
240         time.sleep(0.5)
241 
242         if(GPIO.event_detected(23)):
243             isPressed4 = True
244             time.sleep(0.5)
245             man_mode()
246         else:
247             isPressed4 = False
248 
249         if(GPIO.event_detected(24)):
250             isPressed5 = True
251             GPIO.output(27, False)
252             GPIO.output(17, False)
253             time.sleep(0.2)
254             GPIO.output(17, True)
255             time.sleep(0.2)
256             GPIO.output(17, False)
257             time.sleep(0.2)
258             GPIO.output(17, True)
259             time.sleep(0.2)
260             GPIO.output(17, False)
261             readConfig()
262             auto = 1
263             with open ('pid.conf', 'w') as f:
264                 f.write('%s,%s,%s,%s,%s,%s'%(SP,Kp,Ti,Td,auto,man))
265             lcd.clear()
266             lcd.cursor_pos = (0, 0)
267             lcd.write_string("Auto mode set")
268             time.sleep(0.5)
269             showAll_A()
270         else:
271             isPressed5 = False
272 
273 #---Try/except to run from start---
274 try:
275     lcd.clear()
276     lcd.write_string("Welcome!")
277     time.sleep(2)
278     lcd.clear()
279     while True:
280         readConfig()
```

```
281         if(auto == 1):
282             GPIO.output(17, True)
283             GPIO.output(27, False)
284             showAll_A()
285 
286         elif(auto == 0):
287             GPIO.output(27, True)
288             GPIO.output(17, False)
289             showAll_M()
290 
291 except KeyboardInterrupt:
292     lcd.clear()
293 #   loop_string(long_string, lcd, framebuffer, 1, 16)    #Uncomment for long string
294     lcd.write_string("Goodbye")                          #Comment for long string
295     time.sleep(2)
296     lcd.close(clear = True)
297     GPIO.cleanup()
```

Listing 5: Det fysiske brukergrensenittet (LCD)

Main for plotting av logget data:

```
1 import PLT_plot as plot
2 import PLT_config as config
3 
4 def main():
5     plot.plot(config.filename)
6 
7 main()
```

Listing 6: main (plotting logged data)

Konfigurasjon av plott:

```
1 filename = '05_02_2022-17:40'
2 
3 sensors = {
4     'sensor_0' : 'sensor_0',
5     'sensor_1' : 'sensor_1',
6     'sensor_2' : 'sensor_2',
7     'sensor_3' : 'sensor_3',
8     'sensor_4' : 'sensor_4'
9 }
```

Listing 7: configuration of plott

Plotter av loggdata:

```
1 #------Import _config------
2 import PLT_config as config
3 
4 #----------Libraries-----------
5 import matplotlib.pyplot as plt
6 import numpy as np
7 import pandas as pd
8 
9 #--------Degree symbol---------
10 degree_sign = u'\N{DEGREE SIGN}'
11 
12 #--Plot function--
13 def plot(filename):
14 
15     path_to_file = 'temp/' + filename + '.csv'
16 
17     #--------Reads temp file--------
18     data = pd.read_csv(path_to_file)
19     x = data["x"]
20     temp0 = data["dtemp0"]
21     temp1 = data["dtemp1"]
22     temp2 = data["dtemp2"]
23     temp3 = data["dtemp3"]
24     temp4 = data["dtemp4"]
25 
```

```
    #----Plot all temperature readings----
    plt.plot(x, temp0, label = config.sensors['sensor_0'], linewidth = 1.5, color =
     '#4876FF')
    plt.plot(x, temp1, label = config.sensors['sensor_1'], linewidth = 1.5, color =
     '#EE0000')
    plt.plot(x, temp2, label = config.sensors['sensor_2'], linewidth = 1.5, color =
     'orange')
    plt.plot(x, temp3, label = config.sensors['sensor_3'], linewidth = 1.5, color =
     '#008B45')
    plt.plot(x, temp4, label = config.sensors['sensor_4'], linewidth = 1.5, color =
     '#708090')

    #-------Plot settings--------
    plt.xlabel('Time [hh:mm:ss]')
    plt.ylabel('Temperature' + '[' + degree_sign + 'C]', fontsize=10)
    plt.title(str(filename), fontsize = 15)
    plt.ylim([0, 150])
    plt.xticks(rotation=45, ha='right', fontsize=8)
    plt.xticks(np.arange(0, len(x), 80))
    plt.legend(loc='lower right', prop={'size':10})
    plt.grid()
    plt.tight_layout()
    plt.show()
```

Listing 8: Plotter of logged data

13.5 E: Matlab kode

Simulering av matematisk modell:

```
clear
clc
close all

% System matrices from physics-based model
A = [-7.017 7.017 0; 2.248 -2.248+3.32*10^(-5) 3.32*10^(-5); 0 2.496*10^(-4)
    -2.496*10^(-4)];
B = [24.938; 0; 0];
C = eye(3);
D = 0;

% Weighting factors
Q = diag([0.1 0.1 10]);
R = (0.01);

%Desired equilibrium states for the system
xd = [393.15; 393.15; 393.15];

%Initial conditions
x0 = [295.15; 295.15; 295.15];

%LQR solution
[K, S, E] = lqr(A, B, Q, R);

%Close loop system
sys = ss(A-B*K, -(A-B*K), C, D);

%Final simulation time
tfinal = 15000;
time_total = 0:0.1:tfinal;

%Close loop input
closed_loop_input = [xd(1)*ones(size(time_total)); xd(2)*ones(size(time_total)); xd
    (3)*ones(size(time_total))];

%Simulation of the temperature response
[output_closed_loop,time_closed_loop,state_closed_loop] = lsim(sys,
    closed_loop_input,time_total, x0);

%Plot of the temperature response
figure(1);
plot(time_total,state_closed_loop(:,1))
xlabel('Time [s]')
ylabel('Temperature [K]')
legend('Temperature of the heating mat')
grid on;

figure(2);
plot(time_total,state_closed_loop(:,2))
xlabel('Time [s]')
ylabel('Temperature [K]')
legend('Temperature of the plate')
grid on;

figure(3);
plot(time_total,state_closed_loop(:,3))
xlabel('Time [s]')
ylabel('Temperature [K]')
legend('Temperature of the test component')
grid on;
```

Listing 9: Kode av simuleringen med systemmatrisene fra modellering

Simulering av modell fra systemident:

```
clear
clc
close all

% System matrices from system identification
A = [-0.000636];
B = [4.738e-06];
C = 1;
D = 0;

% Weighting factors
Q = diag([10]);
R = [0.01];

%Desired equilibrium state for the system
xd = [120];

%Initial condition
x0 = [22];

%LQR solution
[K, S, E] = lqr(A, B, Q, R);

%Close loop system
sys = ss(A-B*K, -(A-B*K), C, D);

%Final simulation time
tfinal = 15000;
time_total = 0:0.1:tfinal;

%Close loop input
closed_loop_input = [xd*ones(size(time_total))];

%Simulation of the temperature response
[output_closed_loop,time_closed_loop,state_closed_loop] = lsim(sys,
    closed_loop_input,time_total, x0);

%Plot of the temperature response
figure(1);
plot(time_total, state_closed_loop)
xlabel('Time [s]')
ylabel('Temperature [°C]')
legend('Temperature of the test component')
grid on;
```

Listing 10: Kode av simuleringen med systemmatrisene fra systemidentifikasjon

13.6 F: Regnskap

Økonomiske midler	8 000kr			
Utgifter:	Pris:	Antall:	Frakt:	Beløp:
Varmepære	173	2	294	640
Varmematte	486,07	2	0	972,14
Div. komponenter OV	333	1	0	333
ADC	131	1	0	131
Div. komponenter OV	13	1	0	13
Div. komponenter OV	48	1	0	48
Div. komponenter OV	34	1	0	34
Div. komponenter OV	40	1	0	40
Div. komponenter OV	18	1	0	18
Div. komponenter Clas	233	1	0	233
Kjølepasta Kjell	50	1	0	50
Div komponenter OV	58	1	0	58
Div. komponenter OV	32	1	0	32
HDMI kabel 10 m	129	1	0	129
Biltema	70	1	0	70
Div. komponenter OV	21	1	0	21
Klut og brødposer	50	1	0	50
		Totalt:		2872,14
		Rest:		5127,86

Figur 44: Figuren viser regnskapet til bachelorgruppa.

13.7 G: Timeregnskap

Gruppemedlem	Timer totalt dokumentasjon	Timer totalt praktisk	Timer totalt	Planlagte timer
HDM	135,5	223,5	359	473
SG	137	208	345	473
SAS	241	166	407	473
MKG	204	214,5	418,5	473
Totalt gruppen	717,5	812	1529,5	1892

Figur 45: Figuren viser det totale timeregnskapet til bachelorgruppa.

Det individuelle timeregnskapet er lagt til i ZIP-mappa.

13.8 H: Brukermanual FTTRv4 (User Manual FTTRv4)

13.8.1 Introduction

This user manual provides a detailed description of how to set up and operate FTTRv4. FTTRv4 is the newest version of Orbit's thermal vacuum chamber. This version of the system includes automatic control (PID-controller), a GUI (real time plotting and settings), and a physical interface (LCD-screen). Make sure to follow this guide before turning on the vacuum chamber!

13.8.2 Equipment

Name/components
Raspberry Pi 3A+
Power supply unit (RND 320-KD3305P)
Rig with heating pads
Transistor circuit
LCD circuit
HDMI cable
PC monitor

13.8.3 Necessary packages for Raspberry Pi

Most of the packages can be installed with **sudo apt-get install**. Some of them can be installed with **pip install** or **pip3 install** if pip is installed on Pi.

Necessary packages for Raspberry Pi
Pandas
Numpy
tkinter
Matplotlib
smbus
I2C Tools
setuptools
board (comes with setuptools)
busio (comes with setuptools)
ADS1x15 library from CircuitPython
RPLCD from PyPI
RPi.GPIO from PyPI

13.8.4 Setup

1. Plug the power cord into a wall outlet.
2. Connect the Raspberry Pi to power.
3. Connect the Transistor- and LCD circuits to its appropriate pins on the Raspberry Pi. For more information, see the schematic in chapter 5.5: Figure (25)
4. Connect the heating pads in series, and fasten them.
5. Connect one end of the HDMI cable to the Raspberry Pi and the other to the computer monitor to control og surveil the system.
6. Set the power supply unit to 26,5 V.

13.8.5 Setting up DS18B20 sensors on Pi

- Open the command window and run:

```
$ sudo nano /boot/config.txt
```

  and insert the line **dtoverlay=w1-gpio** at the bottom of the file.

- Restart the Pi
- Again open the command window and run:

```
$ sudo modprobe w1-gpio
$ sudo modprobe w1-therm
```

- Open the file location for the sensors with:

```
$ cd /sys/bus/w1/devices
```

  If the sensors are connected correctly they will show here with an address in the form: **28-XXXXXXXXXXXX**.

- Test the sensors by going into their location by cd and device address and run:

```
$ cat w1_slave
```

  The temperature will be shown as **t=xxxxx**. In room temperature the two first numbers represent the temperature and the rest are decimals. Ex.: $t = 23456$ equals to $t = 23.456°C$.

13.8.6 Configuring I^2C on Pi

- Open command window and run:

```
$ sudo raspi-config
```

- Under "Interfacing Options" - "Advanced Options" - Option nr. 7 I2C: turn on.
- After reboot of Pi I^2C should work.
- Install smbus and i2c-tools:

```
$ sudo apt-get install python3-smbus python3-dev i2c-tools
```

- To test I^2C use the command:

```
$ sudo i2cdetect -y 1
```

  You will get a list of I^2C-addresses. The standard for ADS1115 is 0x48 and for LCD-backpack 0x27.

13.8.7 Running the system

- Make sure the FTTRv4-folder is up to date. Use *git pull* for this.
- On the Raspberry Pi open the terminal window and cd to **FTTRv4**. For all the functionality you need to open three more terminal windows. To do this press Ctrl + Shift + N.
- In the first window you need to run the **__main__.py** code with

```
$ sudo python3 __main__.py
```

This is the code for temperature measurements and PID-controller.

- In the second window you need to grant permission to the pid.conf file. To do this run

```
$ sudo chown pi:pi pid.conf
```

This needs to be done every time the main code starts, or else the GUI-code will not run.

- In the third window you can run the GUI-code with

```
$ /bin/python /FTTRv4/FTTRv4_GUI
```

If you choose to run the code with

```
$ sudo python3 FTTRv4_GUI
```

some of the functionality may not work, such as the live plotter or the input fields.

- In auto mode it's important to grant permission to the u_total.csv file, or else the control value indicator will not show. This can be done in the same way as the pid.conf file:

```
$ sudo chown pi:pi u_total.csv
```

- In the last window you may start the code for the physical UI with

```
$ sudo python3 FTTRv4_LCD
```

13.8.8 Physical UI menu guide

- At start the read-only mode is active. Here you see setpoint/manual value and process value. The green led indicates auto mode and the red manual mode.
bilde?
- Press the left button to go into the mode for adjusting setpoint or manual value. Press the right button to set the value.
bilde?
- In read-only mode press the right button to change mode (auto/manual).
bilde?

13.8.9 Troubleshooting

- Go through the system components. Any loose connections?
- Check the indication LEDs on the circuitry.
  - If the indication LEDs are faulty use a multimeter.
- If the LCD screen is not showing anything after you run the code, you can try running it again (Ctrl + C, wait, arrow up then enter).

Varmereguleringssystem for testing av komponenter i termisk vakuum

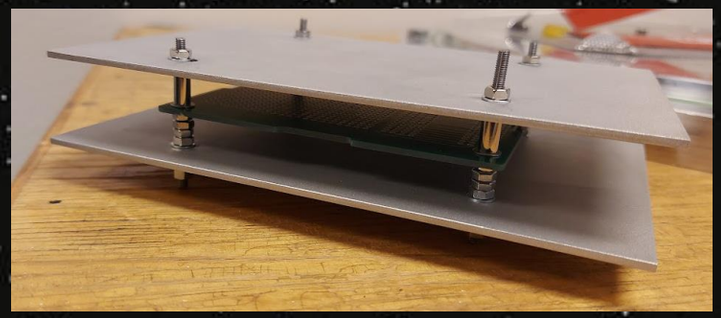
Bakgrunn:

Orbit NTNU er en teknisk organisasjon som konstruerer kubesatellitter. Før en slik kubesatellitt skytes ut i verdensrommet er det viktig at den blir testet grundig, for å sjekke at den tåler vakuum-tilstanden i rommet og varmestrålingen som befinner seg der. For å teste at en satellitt tåler disse forholdene brukes det et termisk vakuumkammer for å undersøke dette.

Orbit NTNU har et eget termisk vakuumkammer som de ønsker å forbedre og automatisere. I denne bacheloroppgaven skal prosjektgruppen se på mulige forbedringer til dette kammeret, utvikle et reguleringssystem som gjør at testobjektet i kammeret oppnår ønsket temperatur uten manuell styring og gjøre oppsettet mer brukervennlig

Oppsett:

FTTRv4 (Fancy Thermal Testing Rig) som er navnet på systemet som har blitt utviklet består av Raspberry Pi, spenningsforsyning, fem temperatursensorer, to varmematter og et fysisk brukergrensesnitt. Komponenten som skal testes festes mellom to plater med hver sine påklistrete varmematter. Når disse platene varmes opp vil platene radiere varme på testobjektet.



Regulering:

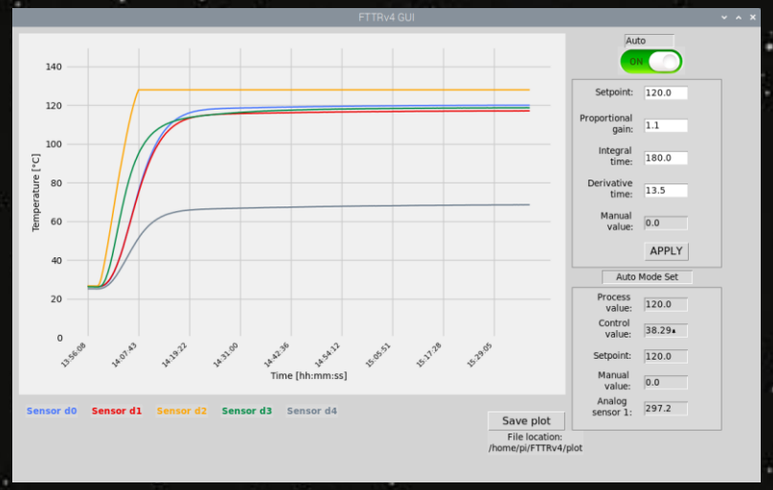
Varmereguleringen av testobjektet skjer via en PID-regulator. Denne regulatoren har blitt diskretisert og implementert i kode på Raspberry Pi-en. For å få til en god regulering ble det gjennomført systemidentifikasjon, slik at man fikk parameterinnstilt regulatoren på en bra måte. En del arbeid ble gjort for å implementere LQR, men det ble for lite tid til å teste den ordentlig ut.

Transistorkrets:

For å kunne styre effekten som suppleres til varmemattene fra spenningsforsyningen, ble det utviklet en egen transistorkrets. Denne transistorkretsen skrur av og på varmemattene i takt med et PWM-signal fra regulatoren, og gjør det dermed mulig å styre effekten ganske nøyaktig.

Brukergrensesnitt:

Det ble laget to brukergrensesnitt til systemet; et fysisk og et grafisk brukergrensesnitt. Det fysiske brukergrensesnittet består av en LCD-skjerm med to knapper og et potmeter for bruker input. Det grafiske derimot visualiseres via en PC-skjerm. Dette brukergrensesnittet innehar sanntidsplotting, logging og gir brukeren mulighet til å endre på regulatorparametere mens systemet er i drift.

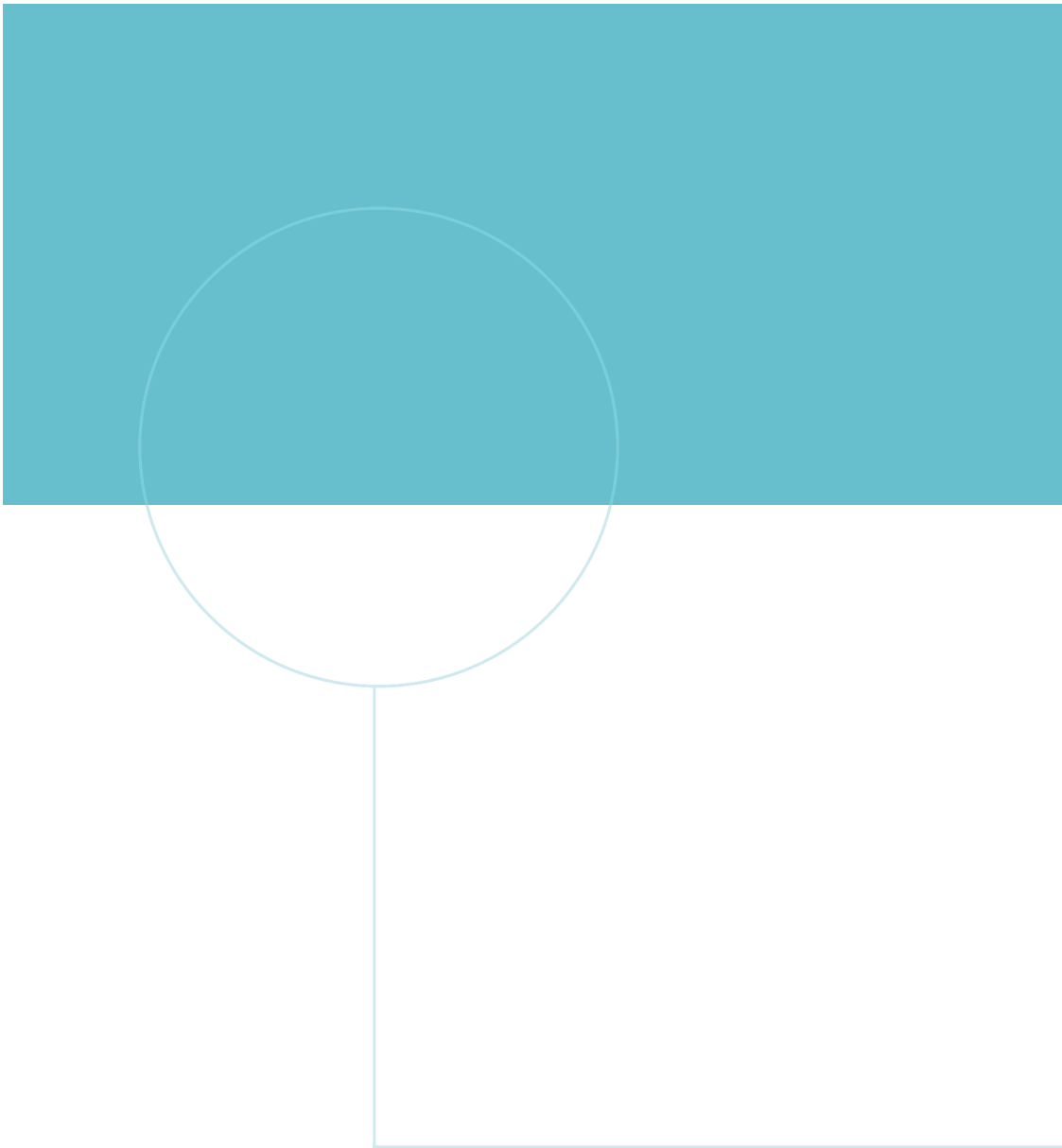


Resultat:

Resultatet av arbeidet er et velfungerende reguleringssystem som er bedre enn den tidligere versjonen. Systemet bruker omtrent 20 minutter på å nå 120 grader i vakuum, noe som er betydelig bedre enn tidligere oppsett. Dessuten har de to nye brukergrensesnittene som har blitt utviklet gjort operasjonen av vakuumkammeret mye mer brukervennlig. Grunnarbeidet med modellering av systemet og simulering med LQR gjort, så Orbit kan ta med denne kunnskapen videre for å forbedre systemet enda mer.

Forfattere: Martin K. Gløsmyr, Sverre Graffer, Håkon D. Mathisen og Sacit A. Senkaya

Orbit NTNU



NTNU

Kunnskap for en bedre verden