NTNU

Tittelside Bacheloroppgave BIELEKTRO

Oppgavetittel (norsk og engelsk):

Robotisert tilberedning av kaffe og kaker

Robotized preparation of coffee and cakes

Forfattere:
Karoline Margrethe Mørkeng
Renate Heierstad Klemetsdal
Astrid Meen Wold

Prosjektnummer: 2314
Innleveringsdato: 29.05.2023
Gradering: [ ] åpen
[x] lukket

Studium: **Bachelor Elektroingeniør (BIELEKTRO)**

Studieretning: **Automasjon og Robotikk,**
Elektronikk og Sensorsystemer

Veileder internt: Sigurd Gosse'
Institutt: Instituttet for Teknisk Kybernetikk og Instituttet for Elektroniske systemer

Oppdragsgiver: PPM Robotics AS
Kontaktperson: Trygve Thomessen

Sammendrag (norsk og engelsk):

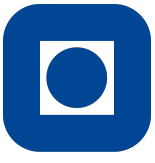
Det er laget et robotisert system for tilberedning av kaffe og kake. Det fungerer som en demonstrasjon på hvordan tilberedning av kaffe og kake kan automatiseres på sykehjem for å frigjøre verdifull tid for pleierne. En industrirobot er montert på en kjøkkenbenk. Roboten fyller kopper med kaffe og plasserer fylte kaffekopper og kake på et serveringsbrett. Serveringsbrettet er montert på en servicerobot som kan frakte varene. For smartere magasinering er det utviklet et sensorsystem integrert i kopp- og kakeholderne. Gjennom et brukergrensesnitt i FlexGUI er alt knyttet sammen til et komplett system.

A robotic system has been created for the preparation of coffee and cake. It functions as a demonstration of how the preparation of coffee and cake can be automated in nursing homes to free up valuable time for caregivers. An industrial robot is mounted on a kitchen counter. The robot fills cups with coffee and places filled coffee cups and cake on a serving tray. The serving tray is mounted on a service robot capable of transporting the items. For smarter storage, a sensor system has been developed and integrated into the cup and cake holders. Through a user interface in FlexGUI, everything is connected to form a complete system.

Stikkord norsk: - Fremtidens Sykehjem - **Nachi MZ04** - **Kompaï K3** - **Servering av kaffe og muffins** - **FlexGUI** - **PCB** - **Bluetooth** - **Raspbery Pi** - **NHL (Nurcing home lab)** - **PPM robotics**	**Stikkord engelsk: - The futures Nursing home** **- Nachi Mz04** **- Kompai K3** **- Serving of coffee and muffins** **- FlexGUI** **- PCB** **- Bluetooth** **-Raspberry Pi** **-NHL (Nurcing home lab)** **- PPM robotics**

Robotisert tilberedning av kaffe og kaker
Prosjektrapport Bacheloroppgave 2023

Karoline Margrethe Mørkeng
Renate Heierstad Klemetsdal
Astrid Meen Wold



NTNU

Abstract

A robotic system has been created for the preparation of coffee and cake. It functions as a demonstration of how the preparation of coffee and cake can be automated in nursing homes to free up valuable time for caregivers. An industrial robot is mounted on a kitchen counter. The robot fills cups with coffee and places filled coffee cups and cake on a serving tray. The serving tray is mounted on a service robot capable of transporting the items. For smarter storage, a sensor system has been developed and integrated into the cup and cake holders. Through a user interface in FlexGUI, everything is connected to form a complete system.

Sammendrag

Det er laget et robotisert system for tilberedning av kaffe og kake. Det fungerer som en demonstrasjon på hvordan tilberedning av kaffe og kake kan automatiseres på sykehjem for å frigjøre verdifull tid for pleierne. En industrirobot er montert på en kjøkkenbenk. Roboten fyller kopper med kaffe og plasserer fylte kaffekopper og kake på et serveringbrett. Serveringsbrettet er montert på en servicerobot som kan frakte varene. For smartere magasinering er det utviklet et sensorsystem integrert i kopp- og kakeholderne. Gjennom et brukergrensesnitt i FlexGUI er alt knyttet sammen til et komplett system.

Innhold

Figurer

Tabeller

Kodelister

Akronymer

CFD Compact Fujikoshi Daihen. 42

NHL Nursing Home Lab. 3

RMU Robot Monitoring Unit. 42

ROS Robot Operating System. 75

TP Teach Pendant. 42

Kapittel 1

Introduksjon

Mangelen på pleiere i eldreomsorgen er et stadig økende problem i Norge. I tillegg går landet en eldrebølge i møte. En måte å møte utfordringen på er gjennom effektivisering. En rekke oppgaver som må gjøres på sykehjem trenger ikke nødvendigvis utføres av en pleier, og ved å automatisere disse oppgavene vil deler av pleiernes verdifulle tid frigjøres. PPM Robotics, oppdragsgiveren for denne bacheloroppgaven, arbeider med automatisering av utvalgte arbeidsoppgaver på sykehjem ved hjelp av robotisering. I den sammenheng består gruppens bacheloroppgave av å robotisere tilberedning og servering av kaffe og kake i en sykehjemskontekst.

1.1 Problemstilling

Oppgavetittelen er "Fremtidens sykehjem med service-roboter og informasjonsteknologi – Oppgave 2 Robotisert tilberedning av kaffe og kaker", og oppgaven består av å videreutvikle PPMs Nursing Home Lab (NHL) med et automatisk robotbasert system for servering av kaffe og kake. Systemet skal settes opp i NHL og fungere som en demonstrator for hvordan fremtidens kjøkken kan effektiviseres for å gjøre arbeidet på sykehjem mer effektivt.

1.2 Forutsetninger og avgrensinger

Det robotiserte systemet skal bestå av én industrirobot og én servicerobot, som skal programmeres til å utføre henholdsvis tilberedning og servering. Tilberedning vil da omtale alt som angår tilberedning av kaffe og kake, fra magasinering og plukking, til plassering på serveringsbrett. Dette vil utføres av industriroboten NACHI MZ04. Servering er alt som går fra å ta imot det NACHI tilbereder, til å levere de tilberedte produktene til mottaker. Dette gjøres av serviceroboten Kompaï K3. Magasinering av kopper og kaker skal utføres med kopp- og kakeholdere. Disse skal være utformet med et sensorsystem som kan informere om magasinenes status, slik at NACHI kan

operere etter oppdatert data. Det skal også utvikles et serveringsbrett med tilsvarende sensorsystem, som skal kommunisere med Kompaï. Både kopp- og kakemagasinene og serveringsbrettet, må utvikles.

1.3 Rapportens oppbygning

- **Kapittel 1: Introduksjon** Introduksjon til problemstillingen og dens forutsetninger og begrensninger.
- **Kapittel 2: Systembeskrivelse** Overordnet beskrivelse av systemets helhet.
- **Kapittel 3: Komponentbeskrivelse** Detaljerte beskrivelser av systemets ulike deler og komponenter.
- **Kapittel 4: Integrasjon** Hvordan systemets ulike deler og komponenter er satt sammen med hverandre for å danne et helhetlig system.
- **Kapittel 5: Resultat** Hva systemet oppnår.
- **Kapittel 6: Videre arbeid** Hva som kan gjøres videre med systemet ved eventuell videreutvikling.
- **Kapittel 7: Konklusjon**

Kapittel 2

Systembeskrivelse

2.1 Oppsett av lablokalet

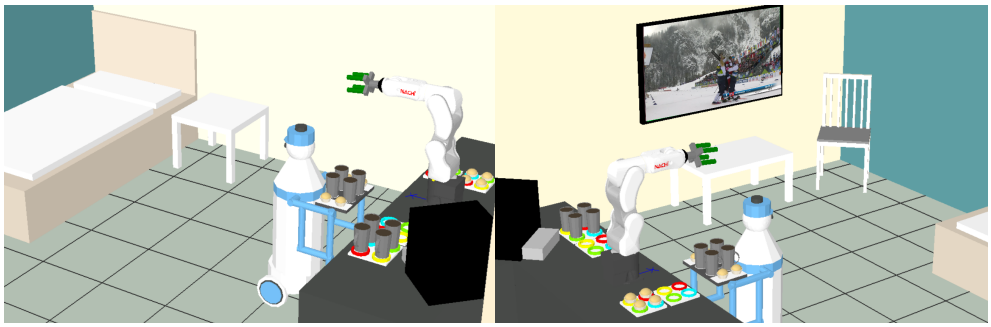
2.1.1 Lokaler

Systemet er satt opp hos PPM Robotics' lokaler, og benytter seg av én industrirobot (NACHI MZ04) og én servicerobot (Kompaï K3). Løsningen fungerer som en demonstrasjon av hvordan fremtidens kjøkken kan effektiviseres for å avlaste pleiere på sykehjem. NACHI tilbereder kaffe/kake og plassere dette på et serveringsbrett, og Kompaï serverer de tilberedte varene til beboerne. Systemets aktiviteter vil foregå i PPMs lokaler, spesifikt i deres Nursing Home Lab (NHL). PPMs Nursing Home Lab er satt opp som et rom egnet for en eldre beboer med omsorg- og assistansebehov. I dette rommet foretar PPM utprøving av nye teknologiske løsninger for eldreomsorg. Rommet har en åpen løsning med eget kjøkken. I figur 2.1a er kjøkkenet fotografert slik det var før montering av komponenter. Resten av rommet, bestående av blant annet en seng, hylle, bord og stoler, er fotografert i figur 2.1b (dette bildet ble tatt med ryggen mot kjøkkenbenken).

(a) Kjøkkenet **(b)** Resten av rommet

Figur 2.1: Fotografier av NHL

2.1.2 Kjøkkenbenken



(a) Kjøkkenet **(b)** Resten av rommet

Figur 2.2: Skjermdump av simulert system i FDonDesk.

Systemets oppsett er tilsvarende figur 2.2. I figuren er systemet avbildet slik det ser ut når NACHI er ferdig med å fylle serveringsbrettet etter å ha startet med fulle magasiner. Her er NACHI utstyrt med en fingergriper. Robotens base er montert på kjøkkenbenken med nødvendige komponenter plassert innenfor robotens arbeidsområde. Kompaï er posisjonert foran benken, med serveringsbrettet tilgjengelig for NACHI. Det er kopp- og kakemagasiner med plass til 8 kopper og 8 kaker. Serveringsbrettet kan holde 4 kopper og 4 kaker. Det er altså lagt opp til at én oppfylling av magasinene er nok til to serveringsrunder.

2.2 Funksjonalitet

NACHI skal tilberede et serveringsbrett med kaffe og kake. Kopper plasseres i kaffemaskinen og kaffemaskinen startes. Kaker plasseres på serveringsbrettet. Innen kaken er på serveringsbrettet er kaffen ferdig disponert av kaffemaskinen. Den fylte kaffekoppen flyttes deretter til serveringsbrettet. Denne prosessen gjentas fire ganger, slik at serveringsbrettet er fullt. Serveringsbrettet er montert på Kompaï, som da er klar til å utføre en serveringsrunde.

Magasineringen av kopper og kaker foregår med et sensorsystem integrert i magasinene. Disse er i stand til å informere om magasinenes status, slik at påfylling kan skje fortløpende. Det legger til rette for at magasinene kan fylles når noen er innom, og ikke bare må skje ved planlagte tidspunkt.

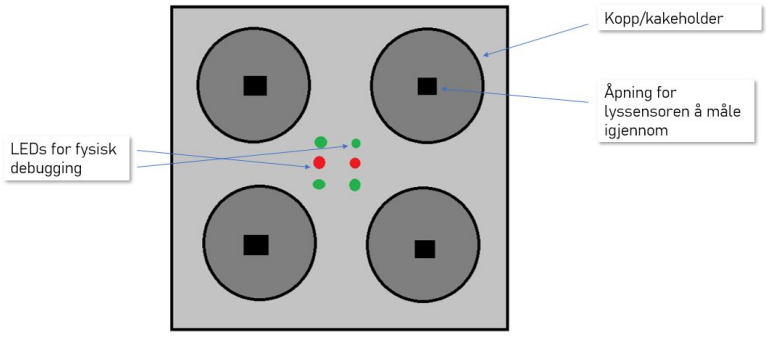
Kapittel 3

Komponentbeskrivelse

3.1 Sensorsystem: 3D-modeller

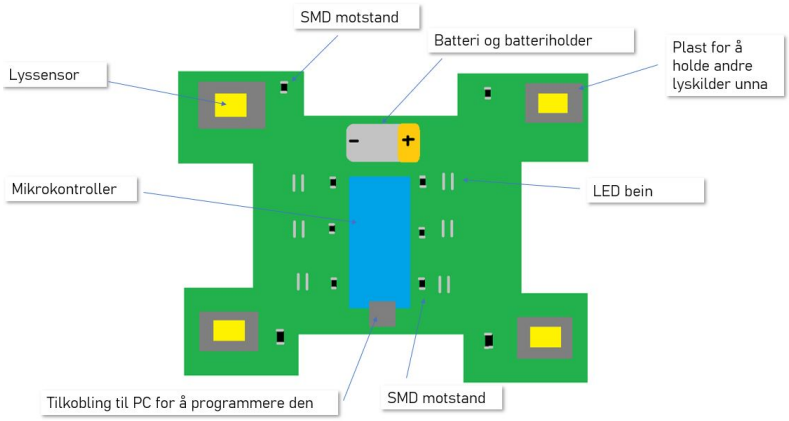
3.1.1 Konseptskisse

Konseptskissen laget i løpet av forprosjektperioden er fungerer som en god intro til sensorsystemets oppbygning og var en planen man fulgte hele veien med få endringer. De viser hvordan 3D-modeller, PCB-en og programvaren skal fungere sammen



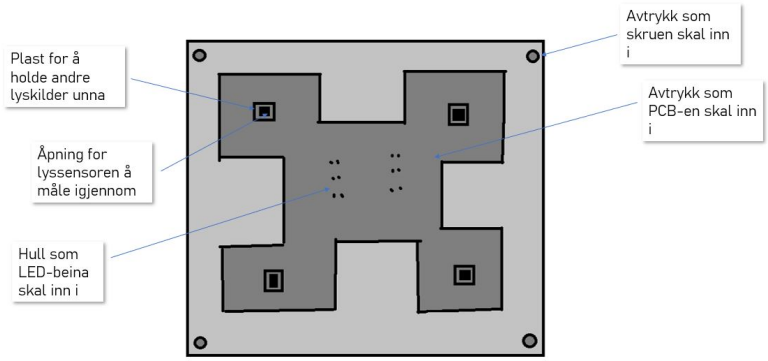
Figur 3.1: Øverste lag av forsiden

Figur 3.1 viser hvordan kopp/kakeholderen med plass til fire objekter med en åpning i midten der en lyssensor skal være for å detektere om det er et objekt i posisjonen eller ikke. I tillegg er det LEDs i midten av holderne for å fysisk kunne se hvilke signaler som sendes over. Dette er for at det skal være lett for en eventuelt bruker, pleier, å se om signalene tolkes rett samt for å forenkle debuggingen under testing.



Figur 3.2: Midterste lag av forsiden

Det midteste laget av forsiden (figur 3.2) tilsvarer PCB-en. I midten er mikrokontrollerne som styrer det hele. Den er koblet til LEDs og LDR-er via motstander. LED-ene lyser hvis det er et objekt i en av holderne og er slukket ellers. Det gjør de basert på lyssensorenes målinger. Dette er programmert i koden. I tillegg er det et batteri med holder på PCB-en



Figur 3.3: Øverste lag av baksiden

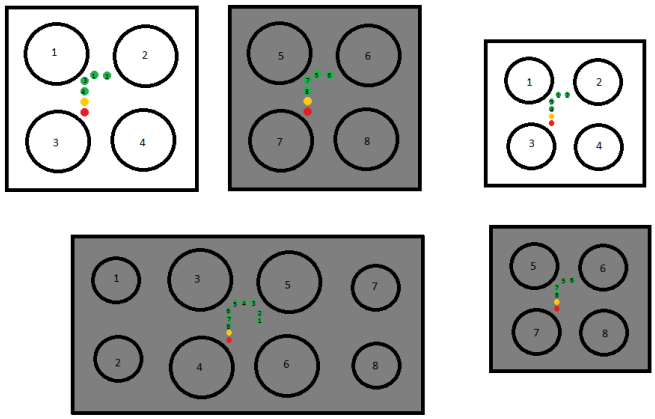
Figur 3.3 viser at 3D-modellen skal ha et avtrykk på baksiden som PCB-en kan legges nedi. I tillegg skal det være hull til LED-ene og LDR-ene slik at de får tilgang til oversiden. For at PCB-en skal sitte ordentlig lages det også en flat 3D-modellbakside med skruer.

Serveringsbrettet skulle benytte seg av samme prinsipp som kopp- og kakeholderne, bare med plass til åtte objekter, fire kaker og fire kopper.

Posisjoner i sensorsystemseksjonene

Videre i sensorsystemet vil ulike posisjoner bli omtalt, disse vises i figur 3.4 og gjelder for sensorsystem-seksjonene. Disse følger i hovedsak nummereringen av komponenter i PCB-en. LEDs og LDR-er med samme tall hører sammen,

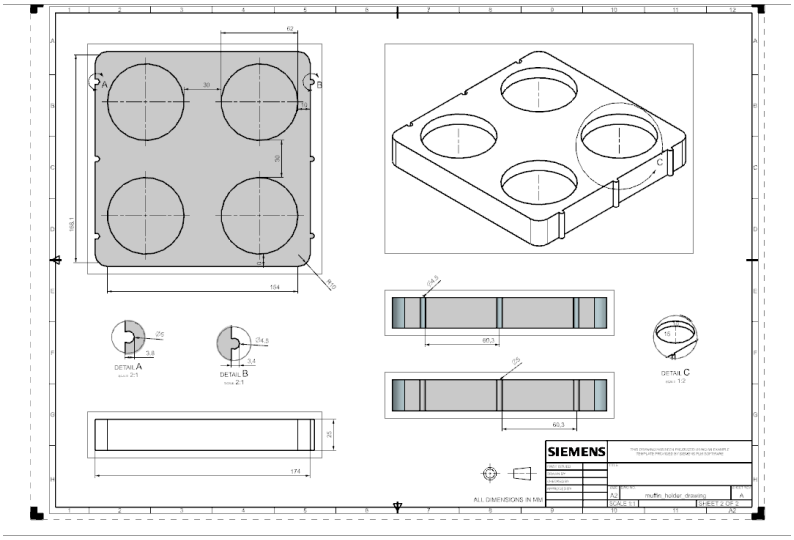
altså dersom det er LED 3 lyser er det et objekt over LDR 3. NACHI følger en annen sekvens av posisjoner



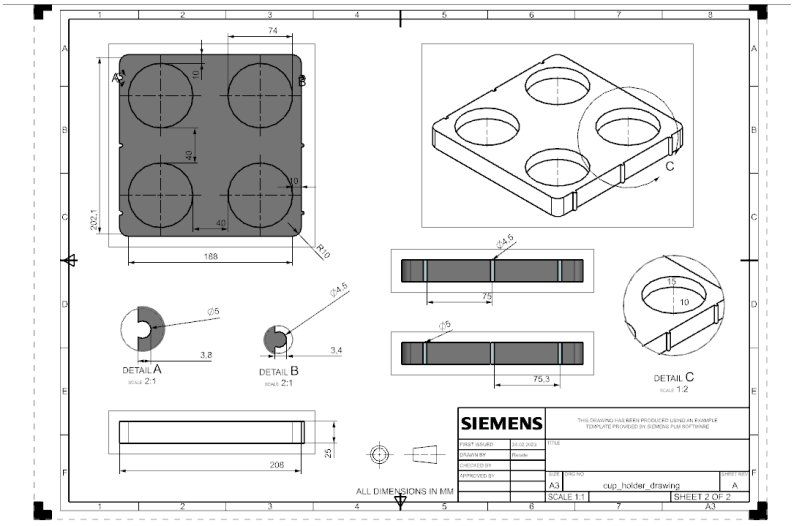
Figur 3.4: Posisjoner for sensorsystemet

3.1.2 Dimensjoner for 3D-modeller

3D-modellene designes med CAD-verktøyet Siemens NX og har ulike dimensjoner. Dette er fordi kopper har større diameter enn kaker. Kakestørrrelsen er standardisert i muffinsformer.

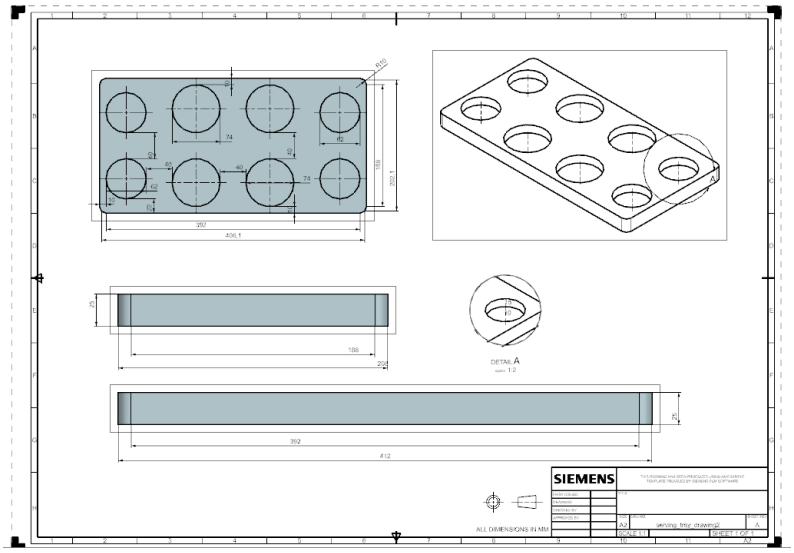


Figur 3.5: Dimensjoner for kakeholder



Figur 3.6: Dimensjoner for koppholder

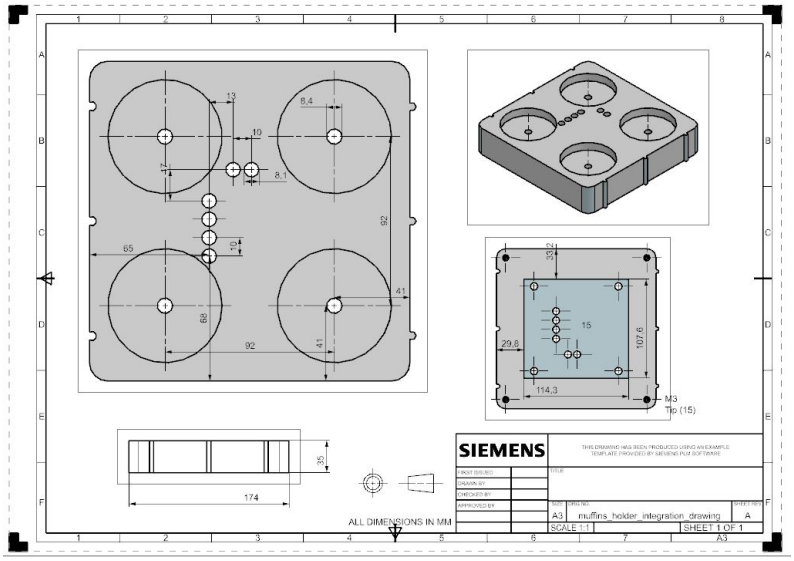
Kopp- og kakeholderne skal stå på kjøkkenbenken og sørge for at NACHI har bestemte posisjoner å jobbe utifra.



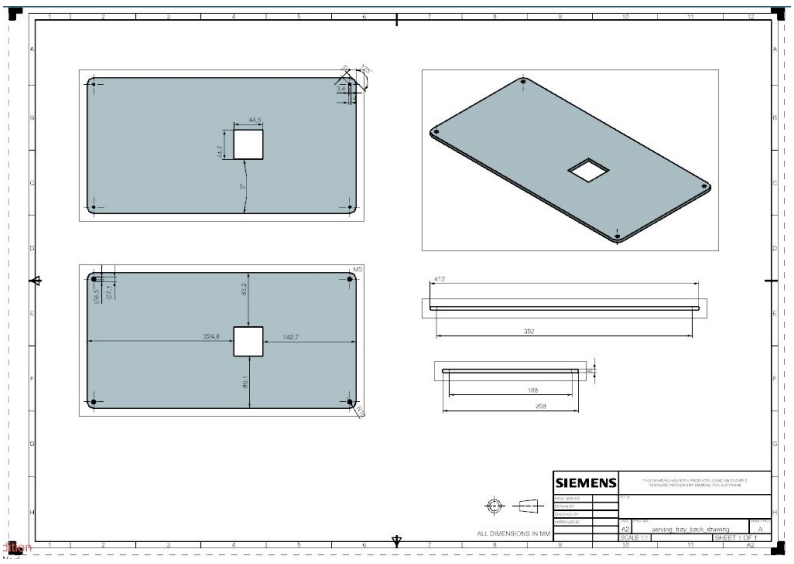
Figur 3.7: Dimensjoner for serveringsbrett

Serveringsbrettet er et kombobrett med plass til både kaffe, som står innerst, og kake som står på begge sider av kaffekoppene. Se figur **??**

3.1.3 Integrasjon mellom 3D-modeller og PCB-er



Figur 3.8: Forsiden av kakeholderen



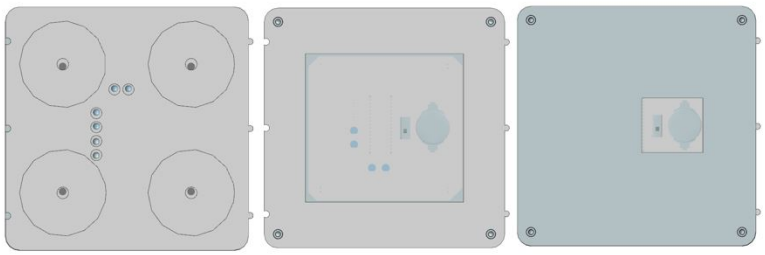
Figur 3.9: Baksiden av serveringsbrettet

Forsidene og baksidene for de ulike modellene ble laget tilsvarende figurene 3.8 og 3.9. Resten av tegningene er i vedlegget

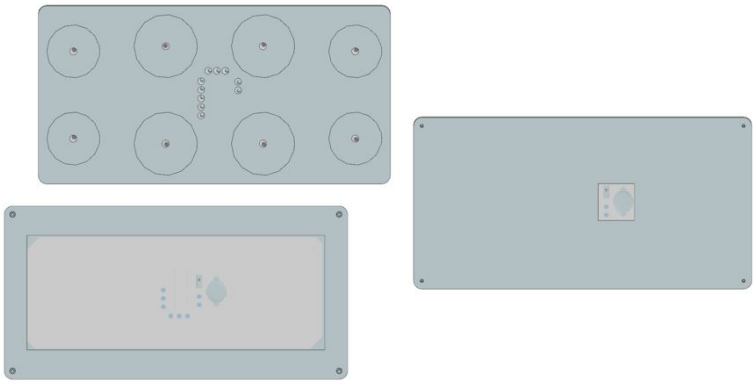
For å integrere PCB-ene inn i 3D-modellene benyttes koordinater og målinger i gjort i Altium, forholdsregning, nyttige operasjoner og virkemåte i NX, og arbeidstegninger for de ulike komponentene.

Det ble sjekket at PCB-ene passet inn i 3D-modellene, både forsiden og baksiden, ved å assemble de i NX. For å gjøre dette så nøyaktig som mulig ble 3D-modellen av PCB i Altium benyttet (se figur 3.10) og omformet til STL-format. Det ble gjort som følgende:

- Eksporterer PCB-ene til VRML-filer i Altium
- Konverterer fra VRML til STL på nettside, siden VRML er uparametriserte objekter i NX
- Importerer STL-filer i NX slik for å assemble alle dele (sjekke at de passer sammen før man printer)



Figur 3.10: PCB-en passer i 3D-modellen for kakeholderen



Figur 3.11: PCB-en passer i 3D-modellen for serveringsbrettet

3.1.4 3D-printing

Det ble brukt en Ultimaker 3 som ifølge spesifikasjonene har et byggevolum på 21.5 cm x 21.5 cm x 20.0[1]. I realiteten er byggevolumet derimot en del mindre bla. grunnet at printerhodet ikke kommer til og for å lage støtte. 3D-modellene måtte deles i flere biter for å få plass i 3D-printeren. Filamentet som ble brukt var av typen PC (polykarbonat). PC er et slitestrekt og fleksibelt materiale som tåler både varme og kjemikaler[2]. Altså egner det seg godt til mindre, robuste fiksturer innen industrien. For å printe med PC-filament kreves det at printeren settes på opp mot 300°og en printer som klarer å jobbe med materialet. Det finnes i fargene svart, hvit og gjennomsiktig og oppbevares tørt. Grunnen til at denne filamenttypen ble valgt, var fordi det var den filamenttypen bedriften hadde mesterfaring med og ønsket å bruke.

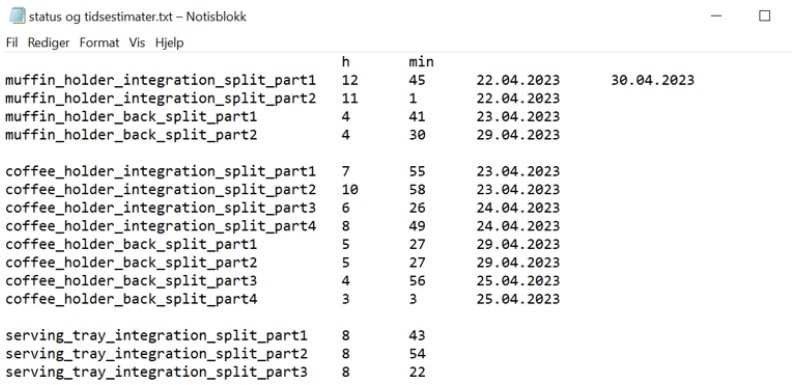
Innstillinger i Ultimaker Cura:

- **Quality**
  - Layer height: 0.2 mm
- **Shell**
  - Wall thicknesss: 0.9 mm
  - Wall line count: 2
  - Top/bottom thickness: 0.9 mm
  - Top layers: 2
  - Bottom thickness: 0.9 mm
  - Bottom layers: 2
  - Horizontal expansion: 0 mm
- **Infill**
  - Infill density: 10%
  - Infill pattern: Triangles
- **Material**

- Printing temperature: 260°C
- Build plate temperature: 110°C

- **Speed**
  - Print speed: 50 mm/s
- **Travel**
  - Enable retraction: Yes
  - Z hop when retracted: No
- **Cooling**
  - Enable print cooling: Yes
  - Fan speed: 0%
- **Support**
  - Generate support: Yes
  - Support placement: Touching buildplate
  - Support overhang angle: 30°
- **Build Plate Adhesion**
  - Enable prime blob: Yes
  - Build plate adhesion type: Brim

Alle 3D-printene ble estimert til å ta 264 timer totalt. Printeren ble tatt med hjem og man måtte planlegge når og hvem som skulle sette på og ta ut 3D-delene. Målet var minst mulig nedetid og feilprints. Det ble gikk noen timer til feilsøking og vedlikehold av printeren blant annet fordi mekaniske deler som hadde flyttet litt på seg


status og tidsestimater.txt – Notisblokk

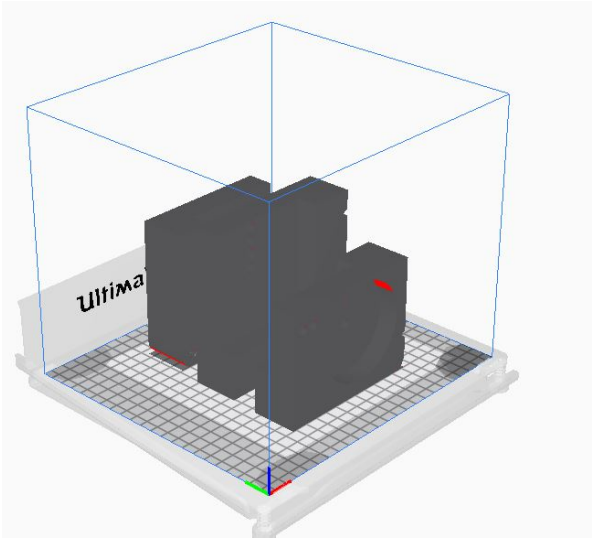
Fil Rediger Format Vis Hjelp

	h	min		
muffin_holder_integration_split_part1	12	45	22.04.2023	30.04.2023
muffin_holder_integration_split_part2	11	1	22.04.2023	
muffin_holder_back_split_part1	4	41	23.04.2023	
muffin_holder_back_split_part2	4	30	29.04.2023	
coffee_holder_integration_split_part1	7	55	23.04.2023	
coffee_holder_integration_split_part2	10	58	23.04.2023	
coffee_holder_integration_split_part3	6	26	24.04.2023	
coffee_holder_integration_split_part4	8	49	24.04.2023	
coffee_holder_back_split_part1	5	27	29.04.2023	
coffee_holder_back_split_part2	5	27	29.04.2023	
coffee_holder_back_split_part3	4	56	25.04.2023	
coffee_holder_back_split_part4	3	3	25.04.2023	
serving_tray_integration_split_part1	8	43		
serving_tray_integration_split_part2	8	54		
serving_tray_integration_split_part3	8	22		

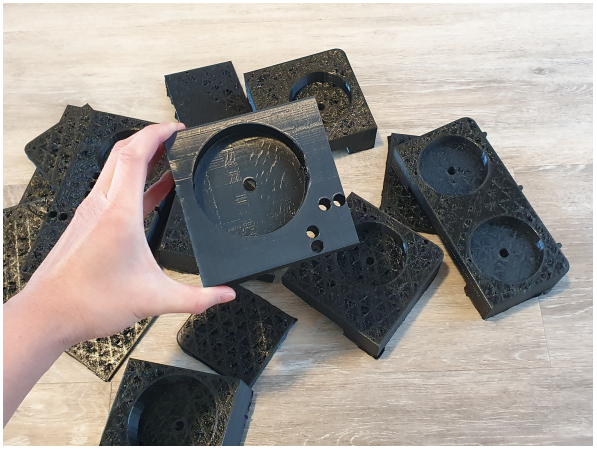
Figur 3.12: Loggføring av 3D-printinger

Det ble etter hvert funnet ut at kvaliteten på delene økte betrakelig dersom man printer dem på siden. I tillegg minimerte det etterbehandlingen delene krevde. Nå trengte man kun å file litt samt å lime dem sammen, istedet for å måtte bruke mye unødvendig tid på å file bort støttmaterialer, fikse hull som

oppsto når fjerningen av støttematerialet ved uhell skapte hull i 3D-printe. Samt at det ble vesentlig enklere å løsne printene fra byggeplatene. Ikke minst kunne man printe flere deler samtidig. Dette sparte man flere timer på å gjøre.



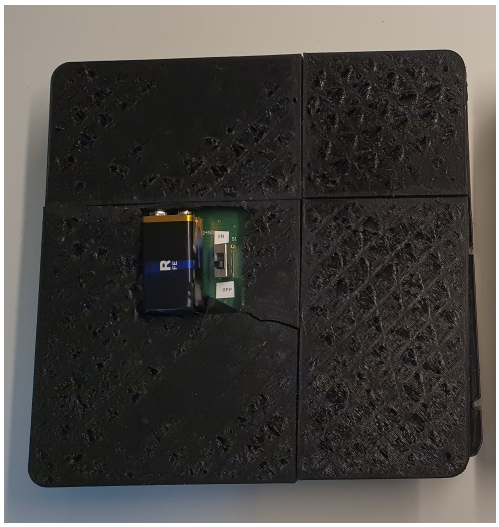
Figur 3.13: Delene til 3D-modellene i Ultimaker Cura-programmet



Figur 3.14: Justeringer i 3D-printfilen ga mye bedre 3D-modelldeler

Lengden og bredden på både forsidene og baksidene passet PCB-ene

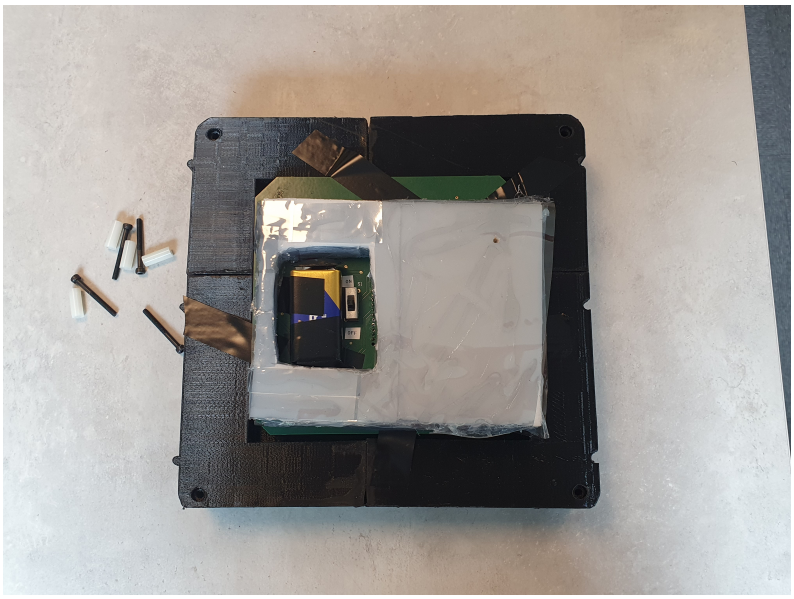
Figur 3.15: Forsiden av 3D-modellene passet PCB-ene



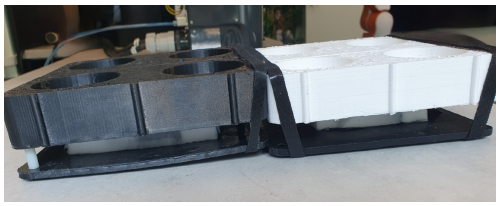
Figur 3.16: Baksiden av 3D-modellene passet PCB-ene

Det ble derimot bommet på høyden med ca. 13 mm. Dette skjedde i hovedsak på grunn av endringer i PCB-designet undervies. Headerne ble ikke tatt med i beregningen samt at batteriet og mikrokontrolleren ble litt høyere enn sine oppgitte mål. Grunnen til dette er at det ikke er mulig å teipe batteriet rett på PCB-en og at målene var litt unøyaktige og at mikrokontrolleren kan settes rett i headeren ved bøying av to av beina. Dette gjorde at 3D-modellene sto skjevt og at PCB-ene ikke lenger kunne legges i avtrykket sitt. Det var begrenset hva man kunne gjøre med dette underveis, ettersom 3D-printingen

og PCB-designendringene skjedde i parallell



Figur 3.17: 3D-modellenes høyder ble fikset med antistatiske svamper



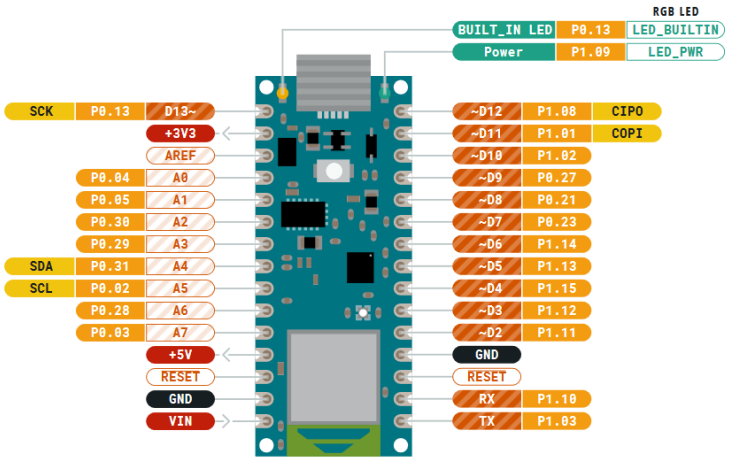
Figur 3.18: Borrelås og skruer med nylonspacere holder modellene sammen

PCB-ene ble først satt inn i 3D-modellforsidene og teipet på plass. Deretter fikk svampene halvert tykkelsen sin og det ble skjært hull til batteriet og knappen. Dette ble gjort for at 3D-modellene skal støtte seg på den samme rette flaten, altså balanse og dersom teipen skulle svikte. Svampene ble dekket med antistatisk plast for å beskytte PCB-en. Man satt på nylonspacere på skruene. Både nylonspacerne og svampene hadde høyden 1.5 mm og var laget av ulike plastmaterialer. Så skrudde man sammen 3D-modellenes forsider og baksider. For 3D-modellene som ikke sto på siden under printingen, kunne det mangle skruehull. Det gikk ikke an å drille egne fordi det var så mye tomrom inni dem. For å løse dette ble det benyttet borrelås. Dette holdt de på plass (se seksjon **??** Resultater)

3.2 Sensorsystem: PCB-er

3.2.1 Komponentvalg

Valg av mikrokontroller

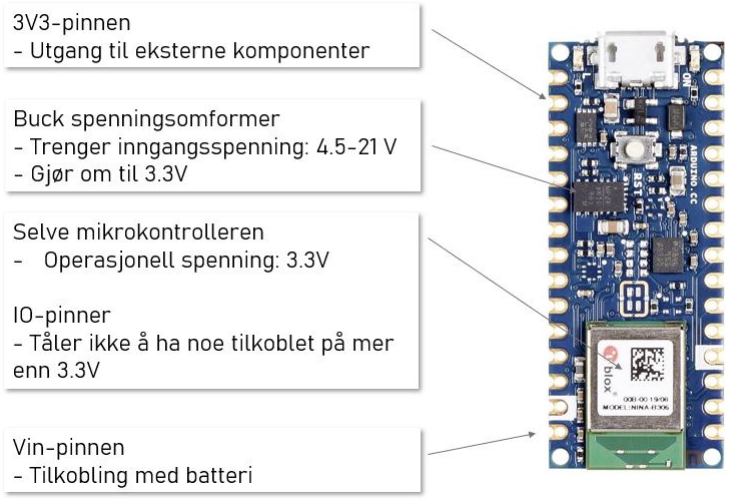


Figur 3.19: Arduino Nano BLE pinout[3]

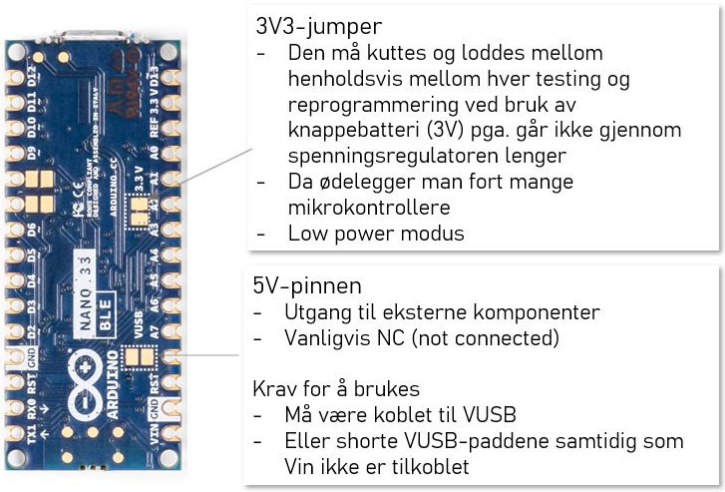
Grunner til å velge Arduino Nano BLE

- 2,4 GHz antenne: Slipper å skaffe antenne som også må loddes på PCB-en. Flat antenne som også passer bedre når mikrokontrolleren skal være inni en boks
- A0-A7[3]: Lyssensor-kretser, analoge innganger
- D2-D11: LED-kretser, utganger som kun trenger verdiene 0 og 1
- Arduino IDE: Brukervennlig og tidligere kjennskap til
- Nordic chip- nRF52840: Kan brukes med ulike trådløse teknologier som Bluetooth, Zigbee, Thread, Matter, NFC..[4] Har allerede en nRF52840 DK liggende som jeg kan teste på
- Muligheten til å bruke VSCode med nRF Connect[5] for mer debugging, kontroll på lavere nivåer eller andre trådløse kommunikasjonsprotkoller
- Vin: Mulighet for eksternt batteri
- Kun 3,3 V
- Liten størrelse: 45 x 18 mm[6]

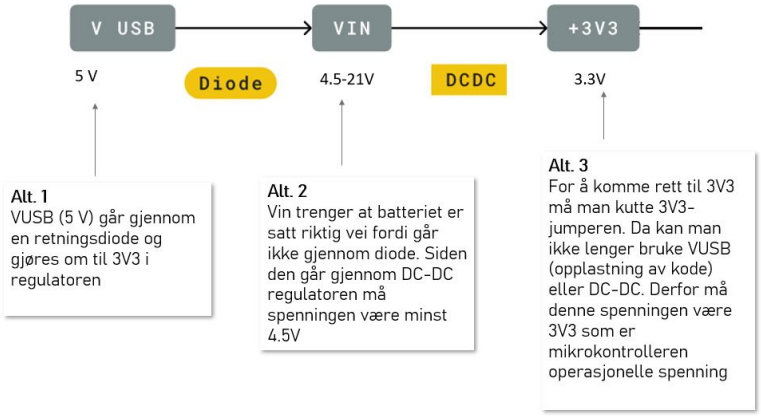
Spenningsgrenser på mikrokontroller



Figur 3.20: [7][6]



Figur 3.21: Sammenhengen mellom loddepadder på baksiden av mikrokontrolleren og spenningsgrenser[8][9]



Figur 3.22: Spenning inn på mikrokontorlleren[7][3][6]

Reset på mikrokontrolleren

Det er både en reset-knapp og to reset-pinner på mikrokontrolleren. Å trykke på reset er som å ta strømmen ut og inn av mikrokontrolleren, det opplastede programmet starter på nytt. Dersom man trykker på reset to ganger, går mikrokontrolleren i bootloader modus[10], altså at mikrokontrolleren er klar for å laste opp kode. Da pulserer den oransje LED-en. Ved jording av reset, slettes mikrokontrollerens bootloader. Mikrokontrolleren får fortsatt strøm, men man kan ikke laste opp noe program lenger. Dette kan fikses, men man trenger noe bestemt hardware og må lodde rett på mikrokontrolleren for å gjøre det

Andre mikrokontrollere som ble vurdert

Valg av sensorer og komponenter

Det ble valgt å bruke LDR-er, ettersom det virker mest hensiktsmessig og praktiske da man bare trenger å ha de under der objektene står istedet for å beregne vekt eller sette ut noen avstandssensorer her og der. Oppgaven deres er å deteketere antallet og hvor kopper og kaker står i de ulike enhetene. Mørkt vil si at det er et objekt i holderposisjonen, mens lyst vil si at det ikker er det. I tillegg er det billig og lett å få tak i. Andre sensorer som ble vurdert var kraft/vektsensor og ToF avstandssensor. Grunnen til at sistnevnte ikke ble valgt var fordi det virker litt omvei å se etter hvor objektene står dersom de har faste plasser. For kraft/vektsensor ville det blitt veldig vanskelig designmessig med tanke på at man trenger en flate som objektene skal veies på. I tillegg kan ulike kaker og kopper veie forskjellig. I tillegg ble kamera vurdert, men ikke valgt grunnet begrenset tid. Ved bruk av kamera måtte man ha utviklet en algoritme for å se etter objektene. Denne algoritmen kan ta unødvendig lang tid både å utvikle og for prosessoren koblet til robotarmen å gjennomføre dersom den skal tilpasses mange ulike kopper og kaketyper.

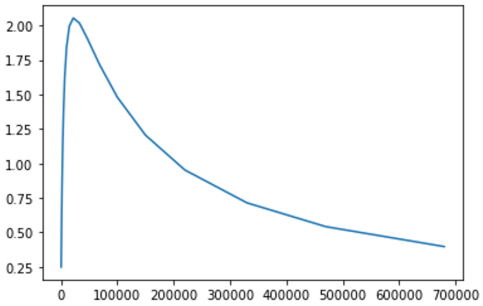
Mikrokontroller	**Vurdering**	**Bruke denn**
Arduino Mega	- Altfor stor - Ujevn flate og kan ikke loddes på PCB	Nei
Arduino Uno med WiFi	- For stor - Ujevn flate og kan ikke loddes på PCB	Nei
Arduino Nano BLE Sense	- Har flere sensorer, men skal være inni en boks	Nei
Arduino Nano IoT[11]	- Mye spennende funksjonalitet - Ikke kjennskap til SAMD21 Cortex-M0+ 32 bit low power ARM MCU	Kanskje
STM32duino	- Virker ustandardisert	Nei
Arduino Nano Every	- Uten trådløs kommunikasjon (antenne)	Nei
ESP32	- Kjent mikrokontroller som har all ønsket funksjonalitet	Backup pla
ESP8266	- Mangler Bluetooth	Nei
Raspberry Pi 3	- For stor - Unødvendig med eget operativsystem	Nei
Raspberry Pi Zero	- Unødvendig med eget operativsystem	Nei
Raspberry Pi Pico W[12]	- Bluetooth i beta	Nei
WizFi360-EVB-Pico	(den nye versjonen dens er Raspberry Pi Pico)	Nei

Tabell 3.1

- Low power LEDs[13][14][15]
  - Strømtrekk: 2 mA (vanlig LED: 10-30 mA)
  - Spenning: 1.7-1.9 V (vanlig LED: 1.8-3.6 V)
- 750 ohm fra E24-serien. Regnet ut med og plottet Python
- LDR
  - Med beskyttelse og skjerming fra sidene[16]
  - Lettere å lodde og lage 3D-design til
  - Lite utvalg, prøvde å finne en med litt kvalitet
- SMD: Dekker 62,3% av spenningene mellom 0-3.3 V med en 22kOhm motstand. Brukte formlene 3.2.1 og 3.2.1
- Skyveknapp
  - Av typen SPDT (single pole double throw)[17], den enkleste knappen for formålet jeg kunne få tak i.
  - Tåler 300 mA -> krever at man beregner strømtrekket i hele kretsen
- CR2450
  - Grei å få tak i
  - Doblet levetid i forhold til CR2032: 560 mAh[18]
- Batteriholder[19]
  - Passer CR2450 som er litt større

$$R_s = \frac{V_s - V_{LED}}{I_{LED}} \tag{3.1}$$

[20]



Figur 3.23: LDR-resistorverdier med tilhørende rekkevidde

$$U(R) = U_0 \cdot \frac{R}{R^* + R} \tag{3.2}$$

[21]

$$R = \sqrt{R^*_{min} \cdot R^*_{max}} \tag{3.3}$$

[21]

Estimering av strømtrekk

Mikrokontrolleren CPU: 6.3 mA[4]
Sending av BLE-signaler: 16.4 mA[4]
Program: 25 mA

I/O LED: 2 mA * 10 = 20 mA[7]
LDR: 330 µA * 8 = 3 mA[16], LTSpice simulering
3V3-pin: 50 mA

For enkelhetens skyld, antas det at strømmen er konstant og at alt på mikrokontrolleren og innganger/utganger kjører uten stans

(6.3 + 16.4 + 25 + 20 + 3 + 50) mA = 120,7 mA < 300 mA Skyveknappen ser ut til å ha god margin

Beregnet batterilevetid: 560 : 120.7 = 4.64 h Strømsparingstiltak i programvare vil bli implementert

NB: det er en stor usikkerhet i disse estimatene, ettersom jeg ikke har noen av komponentene og dermed ikke får mulighet til å sjekke med multimeter. Antar at det er bedre å beregne litt mer enn mindre.

Batteritype	**Vurdering**	**Bruke denne?**
Lipo	- Stor overflate, men tynn - Bevegelig - Kommer nok ikke til å bruke oppladningsfunksjonen - Brannfarlig	Nei
9V batteri	- Stor og klumpete - Kommer ikke til å trenge så høye spenninger	Nei
Knappecelle	- Liten størrelse - Ubevegelig	Ja
AA/AAA	- Stor og bevegelig	Nei

Tabell 3.2

Valg av batteri

Ved batterivalg ble det først og fremst sett på hva slags batteri som ville være lett å designe 3D-modeller etter, var lett å få tak i slik med tanke på pleiere som skal kjøpe det samtidig som å gi nok spenning til mikrokontrolleren og kretsen

3.2.2 Innledende tester før PCB

Formålet med testene var å sjekke om antakelsene rundt det praktiske var rett, altså at man sjekker at man ikke har glemt noe essensielt og oppdager det uforutsette før det er for sent å gjøre noe særlig med det. Noen av dem var ganske grunnleggende

Testene som ble utført var: Pinnetest [Er det nok pinner på mikrokontrolleren og oppfører de seg som forventet?] Testing med eksempelkode laget for Nano BLE [Få erfaring med mikrokontrolleren, nyttig å ta med seg videre] Bruk av 3,3 V-pinnen [Er det nok spenning til alle?] BLE vs Zigbee-test i NHL [Hvilken av de trådløse teknologiene får raskest (best) signal i NHL, brukervennlighet] Måling av strømtrekk [Viktig for valg av komponenter, PCB-design, batterilevetid og om man burde implementere strømsparingstiltak] Egenlagd kode tilpasset problemstillingen med BLE og pinner [Tenke litt i retning den ferdige løsningen, koden vil bli nyttig for senere testing]

For BLE vs Zigbee-testen ble det brukt nRF52840 DK og nRF52840 Dongle, ettersom testen kun var for å måle de trådløse teknologiene opp mot hverandre og det allerede fantes kode for det.

Både BLE og Zigbee virket bra i NHL, men BLE er mye mer tilgjengelig mtp. at de fleste elektroniske enheter har støtte for Bluetooth. Mange programmer er også laget for å vise data sendt og stryke på signalet for den trådløse teknologien. Ikke minst er det mye mer dokumentasjon og informasjon å finne om det på nettet

Dette ser ut til å kunne brukes Pinneantall og –typer ✓

3.3 V pinne ✓
Zigbee ✓
BLE ✓
Strømtrekk ✓

Diverse erfaringer Arduino Nano BLE har to COM-porter den bytter mellom Eksempelkodene var kun laget for en enhet å koble seg på, altså må man finne en måte å sørge for at det er kun RPi-en som får tilgang og eventuelt øke antallet Signalene som ble mottatt byttet rekkefølge og kanskje oversatt (ASCII) avhengig av datatype BLE-signalet i NHL ble oppfattet som godt. Man kunne stå på andre siden av rommet og styre mikrokontrolleren mens den var halvveis dekket av plast Bluetooth Mesh var tregt BLENotify, BLERead og BLEWrite som man skriver i starten av koden handler om hva sentralen (mobilappen) skal ha tilgang til Enkel testing med BLE-mobilapper

Testene gikk uten problemer

Basert på testene, ble endelige valget for mikrokontroller og trådløs kommunikasjonsmetode tatt. Det ble henholdsvis Arduino Nano BLE og Bluetooth LE.

Det var begrenset hva man fikk målt for strømtrekk, ettersom man ikke har alle komponentene eller skriptet som skal brukes ennå

Resultatet fra strømmålingene var: Mikrokontroller: 20 mA (aktive strømsparingstiltak: avskrudd power LED, I2C pull up-resistorer og innebygde sensorer) - LED: 0.2 mA * 8 = 1.6 mA LDR: 5 mA * 10 = 50 mA

Derfor ble det gjort enda et anslag ved bruk av simuleringer, lesing av datablad og å se rundt på nettet

3.2.3 PCB-design

Plassering av komponenter

- Mikrokontroller
  - Plassering: ble ganske fort midtpunktet når alt kobles derfra
  - Plassering: på mikrokontrolleren
  - Støy: tror en ren antennemodul ville skapt litt støy. Her går antennestrømmen (16 mA) på PCB-en på mikrokontrolleren, altså er det ferdig fikset
- Batteri
  - Plassering, støy: I nærheten av mikrokontroller og antenne som trenger mest strøm, men adskilt av knapp (blir den største enkeltstrømmen i kretsen). Plasseres på baksiden for lett tilgang for bruker. Det er nesten kun dens egne ledninger rett over seg på framsiden av PCB-en
  - Tykkere ledninger: viktig at kretsen får strøm

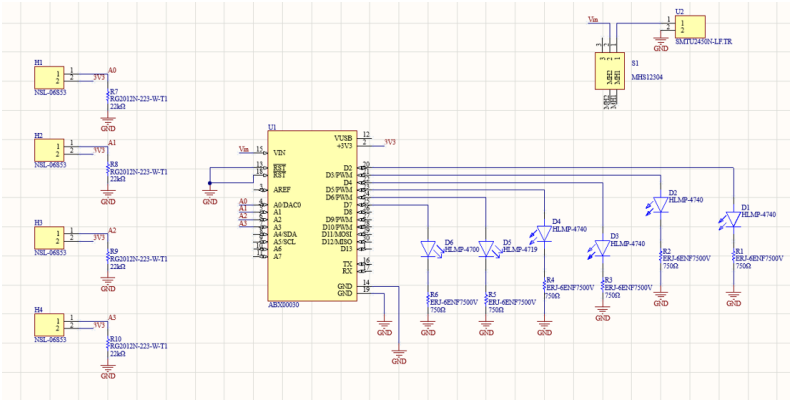
  - Direkte kobling mellom Vin og AGND

- LDR-er
  - Signal: 100-330 µA analoge innganger
  - Plassering: langt unna, midt i hver av kaffe/kakeholderne
  - Tykkere ledninger: for må nå langt med små strømmer
  - Direkte kobling mellom 5V og AGND, tror ikke det er så viktig her, er mest for å få litt kortere ledningsbaner

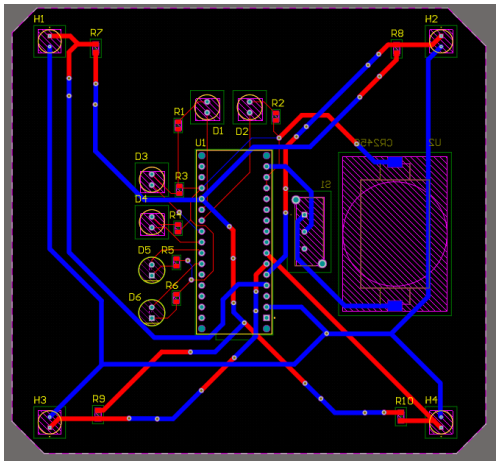
- LEDs
  - Signal: 2mA digitale utganger
  - Plassering: Sentralt nær mikrokontroller grunnet applikasjonens utforming og da kan jeg ha tynne korte ledninger

Jording og støy

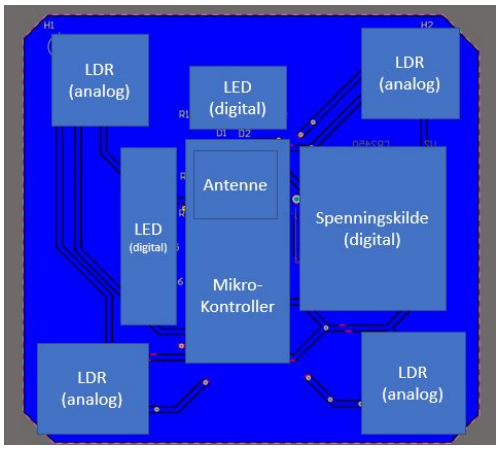
Flesteparten av de analoge signalene er rutet på bunnlaget, mens digitale signaler er først og fremst på topplaget Det er et DGND-jordplan, mens AGND er koblet i stjernejord LDR-ene har AGND mens batteriet har fått DGND sammen med LED-ene. Dette er fordi spenningskilder og digitale signaler er sterkere signaler som tåler mer støy før det gir utslag. Det er forsøkt minst mulig krysninger og avstand mellom de ulike typene signaler De har hver sine GND-pinner på mikrokontrolleren. På grunn av applikasjonens utforming er AGND/DGND blitt litt Oppstykket til tross for at de fortsatt har ulike soner. Designet er symmetrisk.



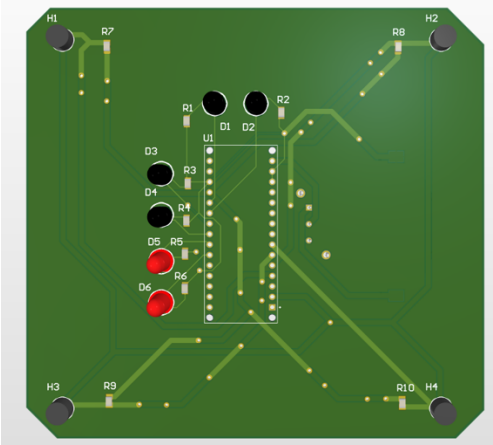
Figur 3.24: Skjemategning



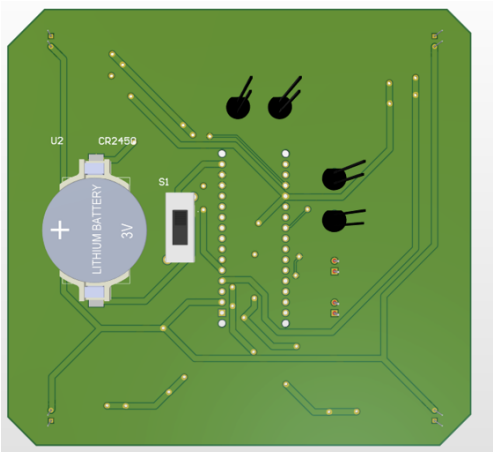
Figur 3.25: Ruting av kobberbaner på PCB



Figur 3.26: Oppdeling av seksjoner på PCB



Figur 3.27: Forsiden av PCB-en



Figur 3.28: Baksiden av PCB-en

3.2.4 Testing av PCB

Det ble ikke lagt til noen ekstra hull, testpunkter eller spenningskilde tiltenkt testing. Dette er fordi det allerede er mange hull og komponentbein man kan koble seg på kretsen eksternt. Utstyret som ble brukt var spenningskilde, ledninger, multimeter, PCB-filene i Altium, Arduino Nano BLE-pinouten og opploddede PCB-er. Mikrokontrollen hadde også plass til USB-kabel i designet sitt fra før av

Det ble testet i to omganger grunnet forsinkelser med levering. Først ble det gjort elektriske tester og sjekking av loddingen

- Sende strøm gjennom hver LED-krets og få de til å lyse

- Måle spenningen over LDR-ene med og uten tildekning. Sjekke at spenningen er >2,5 V uten tildekning og < 0.7 med tildekning (avhengig av lysforhold man måler i)
- Måle spenning rett over batteriet og sjekke at det er ca. 3.3/9V uavhengig av knappeposisjon. Sjekke at det kun er en knappeposisjon som gir spenning på Vin og GND - Grunnet bytte av batteri fra 3.3V til 9V ble denne testen utført to ganger

De siste testene handler om at man laster opp kode på mikrokontrolleren og er støytester, ettersom det vil sendes signaler til og fra alle pinnene samtidig. (Man kunne også sjekket disse analogt ved å måle spenningen over LDR-pinnene på headeren)

- Man laster opp koden på mikrokontrolleren og sjekker at alle LED-ene blinker samt at de verdiene man får inn fra LDR-ene på de analoge pinnene ser greie ut, altså med og uten tildekning. Verdien som kommer inn på mikrokontrolleren er endeverdien
- Ha test/applikasjonskode med BLE på mikrokontrolleren mens bruker batteriet inkl. LED- og LDR-er. Dette tester helheten

Alle de 5 PCB-ene besto testene etter noen justeringer

Endringer

1. Sette mikrokontroller på header i stedet for å lodde direkte på PCB-en For å ikke direkte shorte eller skade mikrokontrolleren Den er svært vanskelig å avlodde Mer praktisk under testing og programmering
2. Bytte fra 3.3 V knappebatteri til 9 V batteri Dette var den nest mest praktiske størrelsen Gir minst mulig endring av PCB/krets og 3D-modellenes baksider Forhindre å ved uhell gi mikrokontrolleren for mye spenning eller ved gjentatte shortinger/loddinger
3. Bøye reset-pinnene på selve mikrokontrolleren slik at de ikke kommer inn i headeren Denne feilen forårsaket at mikrokontrollerne koblet til PCB-en fikk bootloaderen sin slettet Endringene medfører at DGND-ene på reset i kretsen kun blir DGND-viaer som sitter på headeren. Reset får en floating verdi Det vurderes å i stedet gjøre pinnene til interne pullup-er i koden

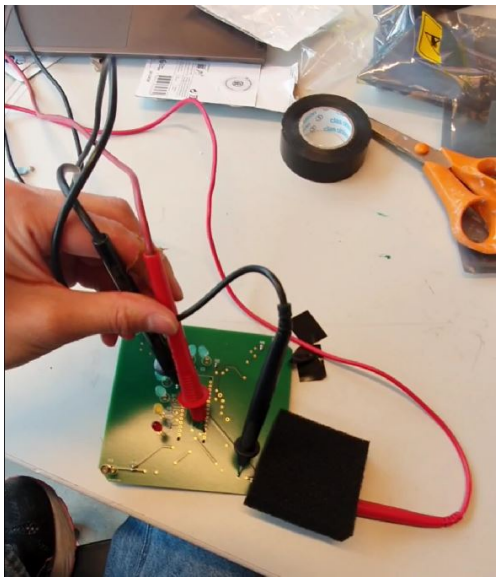
Andre feil som ble oppdaget under testing var short under en motstand, LDR-er som satt løst og en defekt motstand. Disse ble oppdaget og rettet opp

Merking, teiping og isolering

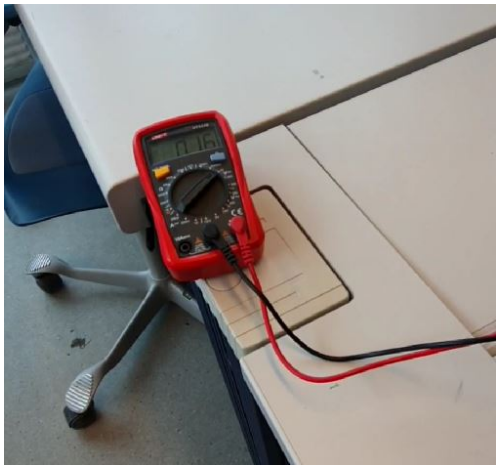
Ettersom mikrokontrolleren var svært sensitiv overfor høye spenninger, ble det iverksatt tiltak for å øke holdbarheten dens

Tiltak	**Uønsket hendelse**
Kutte komponentbein	Shorts, PCB-en får ikke plass i 3D-modellen
Isolere ledninger på batteri	Frying av mikrokontroller
Merke ON og OFF på skyveknapp	Frying av mikrokontroller
Merke hvor USB skal være	Frying av mikrokontroller
Bøye inn og teipe reset pinner på mikrokontroller	Sletting av bootloaderen påmikrokontrolleren
Teipe fast batteri	Batteriet får ikke plass i 3D-modellen

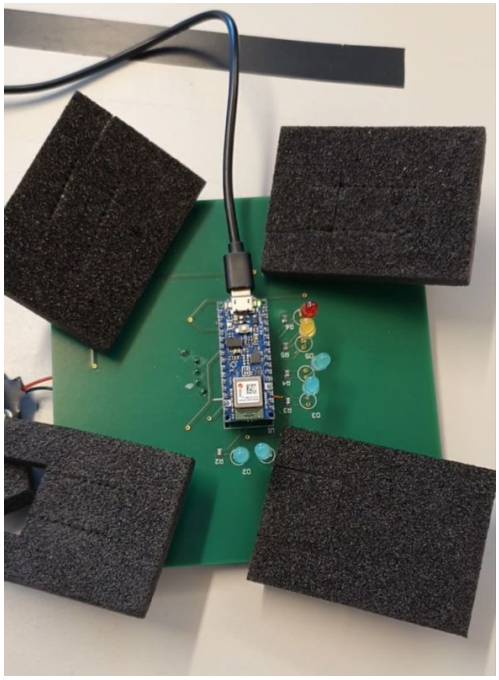
Tabell 3.3: Merking, teiping og isolering av PCB



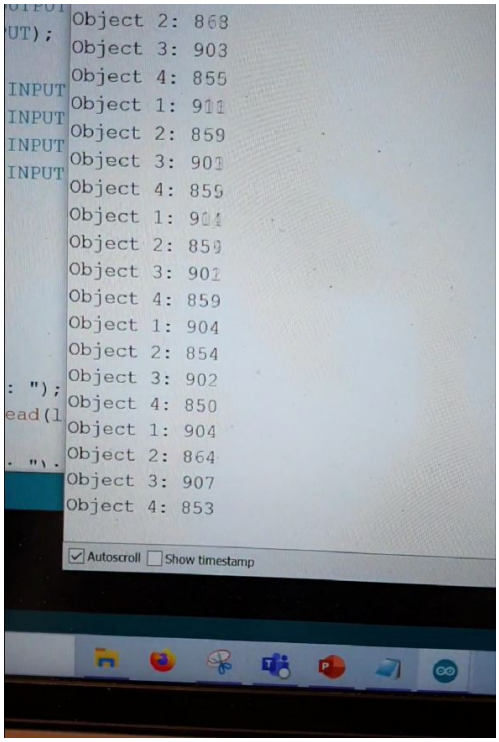
Figur 3.29: Måling av spenning over LDR



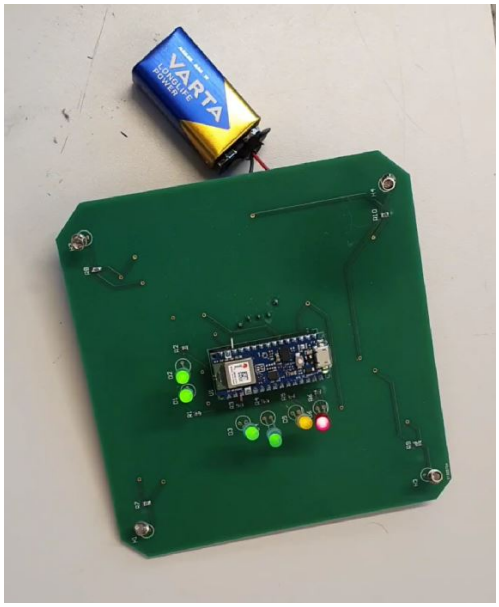
Figur 3.30: Lesing av LDR-spenning med mulitmeter



Figur 3.31: Tildekking av LDR-er



Figur 3.32: Printing av LDR-verdier over UART



Figur 3.33: Testing av at batteri og LEDs

Bluetooth Classic	- Bluetooth versjon 1-3 - Kontinuerlig sending av større mengder data (lyd, video) - Laget for å erstatte kabel - Krever mer strøm
Bluetooth Low Energy (BLE)	- Bluetooth versjon 4 og oppover - Passer for IoT og enheter som går på batteri
Bluetooth Mesh	- Bygd oppå BLE - Bluetooth versjon 4 og oppover - Mer sikkerhet og topologmuligheter

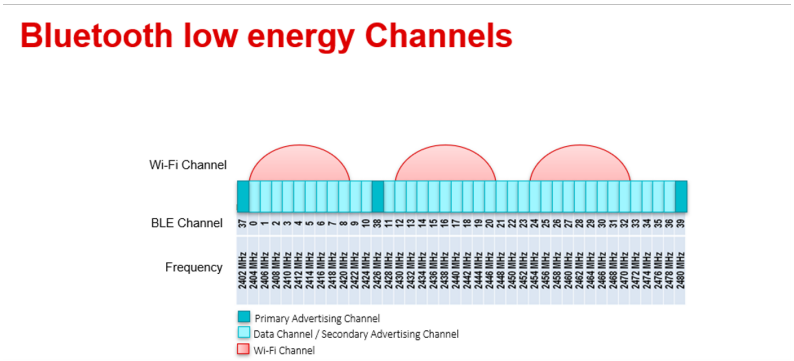
Tabell 3.4: Bluetooth versjoner[22][23]

3.3 Sensorsystem: Programmering

3.3.1 Valg av trådløs teknologi

Ved valg av trådløs kommunikasjon har følgende blitt vektlagt: pålitelighet, strømsparing samt mulighet for skalerbarhet og sikkerhet med tanke på senere utvidelser. Mens hastigheter nær sanntid, sending av store signaler og med stor signalstyrke (effekt) og blitt prioritert i mindre grad. Dette er fordi det skal sendes små signaler over korte avstander og roboten er ikke særlig rask. Passende topologier for formålet er stjerne eller mesh.

Bluetooth



Figur 3.34: BLE-kanaler og interferens med WiFi[**ble_channels**]

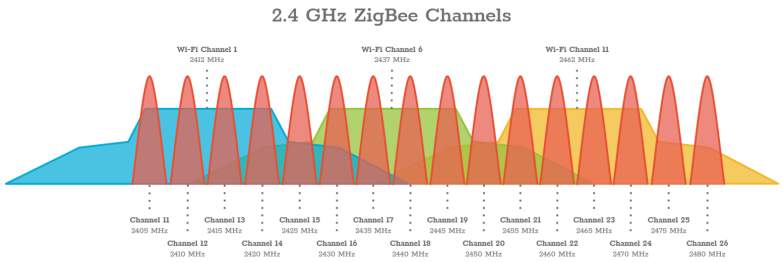
Bluetooth (både Classic og LE) og WiFi er de mest brukte trådløse protokollene innen hjemmelektronikk. For å tilpasse seg WiFi og andre trådløse teknologier på 2,4 GHz-båndet, benytter BLE seg av «adaptive frequency hopping» (AFH) som betyr at BLE-signalet hopper mellom frekvenskanaler for alltid å prøve å være der det er ledig rom når noe skal sendes[24]. BLE er frekvensmodulert

	BLE	**Zigbee**
Høyest datarate/hastighet	✓	
Lengst avstander		✓
Flest topologimuligheter	✓(mesh)	✓
Strømsparing	?	?
Høyest signalstyrke (effekt)	✓	
Størst pakkestørrelse		✓
Nettverkstype		✓
Lavest pris	✓	
Støtte med andre OS	✓	
Sikkerhet	✓(mesh)	✓

Tabell 3.5: Sammenlikning av Bluetooh Low Energy og Zigbee[22][27][28][25][29]

og bruker «cyclic redundancy check» (CRC) for feildeteksjon[25]

Zigbee



Figur 3.35: Zigbee-kanaler og interferens med WiFi[26]

Zigbee benytter seg av fasemodulasjon, lytter etter ledig rom for transmisjon (CSMA/CA) på kanalen og sender signalene som ikke kommer fram på nytt[27]. På den andre siden, når det oppstår interferens mellom Zigbee og WiFi er det oftest slik at førstnevnte forsvinner i WiFi-signalet[26]. Dette er fordi Zigbee er veldig low power i motsetning til WiFi som har middels signalstyrke.

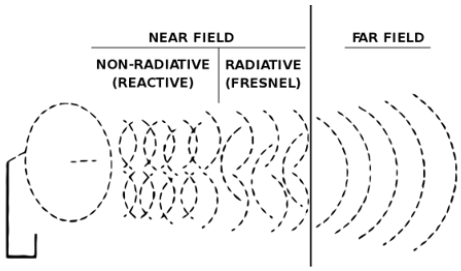
BLE vs Zigbee

Det er ganske jevnt mellom de to ulike protokollene. I tillegg kan det avhenge av versjon, hva slags antenne som er brukt og testforhold. Vurderingene

overfor er basert på inntrykket man har fått fra flere kilder der både Zigbee og BLE benytter seg av 2,4 GHz båndet og er innendørs

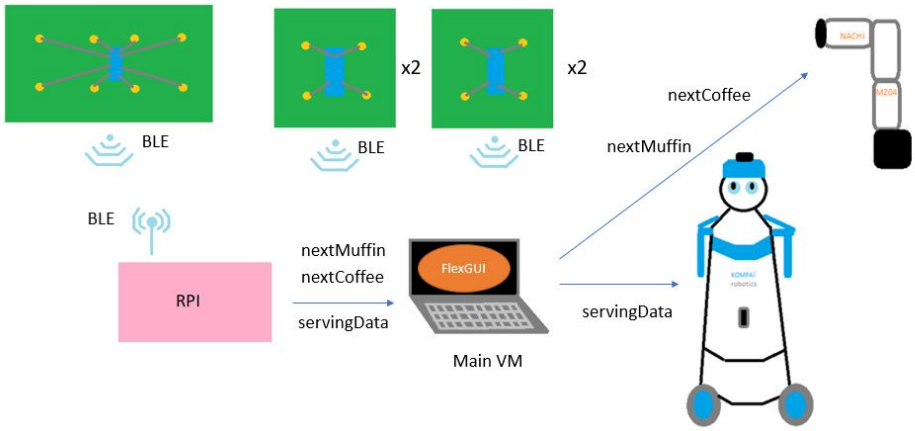
868 MHz-båndet

Lite interferens[30] Det er lovlig å bruke uten radiolisens i Europa[31] Beregnet for lengre avstander grunnet propagasjonen i nærfeltet[32] Vanskeligere å få tak i antenne, og da spesielt flate antenner Generelt mer utilgjengelig



Figur 3.36: Nærfeltet til signaler på 868 MHz-båndet[33]

3.3.2 Overordnet struktur



Figur 3.37: Kommunikasjonssystemets struktur

Mikrokontrollerene på hver av koppholdere, kakeholder og serveringsbrett er periferienheter som alle er koblet til sentralen RPi. Disse kommuniserer ved bruk av Bluetooth Low Energy. Verdiene som sendes fra kopp- og kakeholderne er hvilken posisjon NACHI skal plukke opp et objekt fra. Mens serveringsbrettet forteller om hvilke av holderne det står objekter i. Verdiene blir hentet og kort bearbeidet i RPi før de sendes videre til FlexGUI, som er en GUI oppå ROS. RPi-en er en node med tre publishers som sender til tre ulike

Trådløs kommunikasjonsteknologi	**Vurdering**	**E**
WiFi (2.4 GHz, 5 GHz)	- Robotene og mange andre enheter er kolet til WiFi-nettet på PPM og generelt i bygningen. Internett-tilgangen er allerede treg - Krever middels power for kommunikasjon - Dårlige erfaringer fra tidligere prosjekter	N
Bluetooth Low Energy (2.4 GHz)	- Skal ha både pålitelighet, strømsparing, skalerbarhet og sikkerhet (se sammenlikning mellom BLE og Zigbee i tabell X)	J
Zigbee (2.4 GHz, 868 MHz)	- Skal ha både strømsparing, pålitelighet, skalerbarhet og sikkerhet (se sammenlikning mellom BLE og i tabell X)	K
LoraWAN (868 MHz, 2.4 GHz)	- Virket litt low cost og upålitelig	N
Sigfox (868 MHz)	- Lite utvalg av moduler og de man fant har utstikkende antenner - Står lite om teknologien på nettet	N
Z-wave (868 MHz)	- Brukes i liten grad	N
ESPNOW (2.4 GHz)	- For ESP32 og ESP8266 - Større fordeler å velge en mye brukt standard mtp. dokumentasjon, tilgjengelighet, flere funksjoenr og integrasjon i eksisterende produkter	N
nRF24L01 (2.4 GHz)	- Den mest brukte trådløse modulen som kan fås som hardware innen hobbyelektronikk - Vil heller ha antenne på mikrokontrolleren	N

Tabell 3.6: Vurdering av trådløse teknologier

	MAC-adresse	Service	Characteristic
muffinHolderWhite	A0:9A:B7:3A:DE:F3	muffinManager (CA0E)	nextMuffin (CA1E)
muffinHolderBlack	E4:FB:DF:C7:30:A4	muffinManager (CA0E)	nextMuffin (CA2E)
coffeeHolderWhite	FB:DB:2B:41:E2:E3	coffeeManager (C0FE)	nextCoffee (C0F1)
coffeeHolderBlack	31:6D:13:CA:BA:1B	coffeeManager (C0FE)	nextCoffee (C0F2)
servingTray	A0:61:91:54:42:97	servingManager (0B0C)	servingData (0B1C)

Tabell 3.7

topics: nextMuffin, nextCoffee og servingData. Robotene NACHI og Kompaï henter igjen disse fra FlexGUI og tar beslutninger basert på dette.

3.3.3 Mikrokontrollerkode

Mikrokontrollerkoden er skrevet i C/C++ og benytter seg av ArduinoBLE-biblioteket[34]. Alle kodene starter med at BLE settes opp ved å definere servicen muffinManager og characteristicen nextMuffin. Hver av disse får sin egen UUID. UUID-ene som defineres er custom og står på formen 0000XXXX-0000-1000-8000-00805F9B34FB. Characteristicen defineres som en streng med lengde 5, mens BLERead og BLENotify er tillatelsene som sentralen får når den kobler seg til kakeholderen. Bluetooth LE starter. Det lokale navnet er det som kommer opp når man skanner og kobler seg på enheten. I tillegg advertiser mikrokontrolleren (periferienheten) seg selv, og servicen muffinManager, for at andre skal klare å discover den. Den advertiser på bestemte intervaller. nextMuffin legges til som en characteristic under servicen muffinManager. BLEDevice-objektet settes til å være sentralen som kobler seg på. I denne koden går det kun an for en annen enhet å koble seg på. Ved tilkobling skrives MAC-adressen på sentralen som koblet seg på. Mens den er koblet på kjører koden. I dette tilfellet, leser mikrokontrolleren fra LDR-ene og sender signaler over Bluetooth Low Energy. I tillegg indikerer LED-ene hvilke verdier som tolkes og sendes over. Kake- og koppholderne har lik kode, mens serveringsbrettet sin er litt annerledes. Førstnevnte sender et signal som tilsvarer posisjonen NACHI skal plukke opp et objekt fra, mens for serveringsbrettet er hver av posisjonene en bit i et streng array. Ellers, når det ikke er noen sentral tilkoblet, printes det bare at sentralen er frakoblet

Kodeblokk 3.1: BLE-struktur på mikrokontrollerne

```
1 #include <ArduinoBLE.h>
2
3 BLEService muffinManager("CA0E");
4 BLEStringCharacteristic nextMuffin("CA2E", BLERead |
    BLENotify, 5);
5
6 //defining pin numbers
7
```

```
void setup() {
 Serial.begin(9600)
 //setting LEDs as outputs and LDRs as inputs

 if (!BLE.begin()) {
    Serial.println("Starting Bluetooth Low Energy failed");
    digitalWrite(errorPin, HIGH);
    while (1);
  }

  BLE.setLocalName("Black muffin holder");
  BLE.setAdvertisedService(muffinManager);
  muffinManager.addCharacteristic(nextMuffin);
  BLE.addService(muffinManager);
  BLE.advertise();
  Serial.println("Muffin holder peripheral");


void loop() {
  BLEDevice central = BLE.central();

  if (central) {
    Serial.print("Connected to central: ");
    Serial.println(central.address());
    digitalWrite(wirelessPin, HIGH);

    while (central.connected()) {
    //turn on and off LEDs and send BLE-signals based on LDR-
     values

  }
  }
  digitalWrite(wirelessPin, LOW);
  Serial.print("Disconnected from central");
  Serial.println(central.address());
}
```

BLE-signaler

De svarte og hvite holderne har ulike signaler for å minske antallet signaler som blir sendt til NACHI via FlexGUI. Det sendes tre signaler videre istedet for de fem som RPi-en får inn. Altså er det mikrokontrollerene som finner ut hvilken holder NACHI skal plukke opp et objekt fra (se avsnitt **??**)

I tabell 3.9 betyr X = 1 at det er et objekt i posisjon, mens ved X = 0 er posisjonen tom. Signalene havner til slutt hos Kompaï

	nextMuffin (white)	**nextCoffee (white)**	**nextMuffin (black)**	**nextCoffee (black)**
Plukk opp objekt i pos. 1	0000	0000	0100	0100
Plukk opp objekt i pos. 2	0001	0001	0101	0101
Plukk opp objekt i pos. 3	0010	0010	0110	0110
Plukk opp objekt i pos. 4	0011	0011	0111	0111
Ingen BLE-signaler mottatt	1110	1110	1110	1110
Ingen objekter på holderen	1111	1111	1111	1111

Tabell 3.8: Kopp- og kakeholder signaler

X	X	X	X	X	X	X	X
Pos. 8 (kake)	Pos. 7 (kake)	Pos. 6 (kopp)	Pos. 5 (kopp)	Pos. 4 (kopp)	Pos. 3 (kopp)	Pos. 2 (kake)	Pos. 1 (kake)

Tabell 3.9: Oppbygning av serveringsbrettsignalet

Strømsparingstiltak

Tiltak som har blitt gjort for å spare strøm er å skru av power LED, sensorer og I2C-en på selve mikrokontrolleren siden at dette ikke brukes. Mikrokontrolleren har kun LED-ene på når den er tilkoblet RPi-en og det sendes ikke verdier på alle mikrokontrollerne samtidig. I tillegg kan man skru av UART-kommunikasjoen og sette RESET til digitale pullups slik at den ikke flyter mellom høy og lav.

3.3.4 Raspberry Pi - leddet mellom BLE og ROS

En Raspberry Pi 3B+ brukes som sentral enhet og tilsvarer leddet mellom BLE og ROS. Grunnen til at RPi-en brukes som sentral enhet er i hovedsak at den allerede har Linux som operativsystem og dermed er ROS-kompatibel (både ROS1 og ROS2). Kun et begrenset antall mikrokontrollere er kompatible med ROS1 fra før av[35], altså ville det å velge en ROS1-kompatibel mikrokontroller ført til mindre utvalg i funksjonalitet. Det er enkelte nyere mikrokontrollere å velge mellom dersom man bruker ROS2[36], men da vil man på et senere tidspunkt måtte bridge mellom de to ROS-typene[37]. Det ville også vært en risiko mtp. tidsbruk og å få et ustabilt system dersom man skrev egne drivere for å få mikrokontrollerne til å virke med ROS1. Ikke minst er det unødvendig at alle mikrokontrollerene skal være sine egne noder. I tillegg har den en Cypress CYW43455-chip med Bluetooth versjon 4.2[38]

Konfigurasjon av RPi

For å laste ned Ubuntu 20.04 Server (64-bit) på RPi ble Raspberry Pi Imager benyttet i henhold til denne guiden[39]. Mens for nedlastning av ROS1 Noetic og oppsett av catkin ble denne fremgangsmåten brukt[40].

Kommandoer for å få bluetooth til virke[41]:

- Installere nødvendig programvare: `sudo apt install bluez bluez-tools pi-bluetooth` [42]
- Sjekke at bluetooth-servicen er aktiv: `sudo systemctl enable bluetooth.service` , `sudo systemctl start bluetooth.service` , `systemctl status bluetooth.service`
- Gi brukeren tilltalse til å bruke bluetooth: `sudo adduser USER bluetooth` [43]
- Unblokke bluetooth: `rfkill` , `rfkill unblock bluetooth`
- Sjekke manuelt at bluetooth fungerer ved å koble seg til en annen enhet: `bluetoothctl` `power on` `agent on` `scan on` `pair XX:XX:XX:XX:XX:XX` `connect XX:XX:XX:XX:XX:XX` [44]

3.3.5 Kommunikasjonssystemet

Bleak: For å kommunisere med BLE mellom RPi-en og mikrokontrollerne ble Python-biblioteket Bleak benyttet. Python-biblioteket er konsentrert rundt GATT-funksjonalitet og baserer seg på asyncio-biblioteket, som er ikke-blokkerende, asynkront og kan kjøre flere kodeblokker samtidig[45]. Bleak kan brukes på flere operativsystem, som Windows, Linux og macOS[46] Dette ble installert med `pip install bleak`

RPi-en er koblet på en og en enhet av gangen, fordi den ikke klarer å holde og lese fem tilkoblinger samtidig. Da faller tilkoblingene ut. Dette fører til litt forsinkelser i forhold til at alle er tilkoblet og sender verdier samtidig. På grunn av dette, gikk man litt vekk fra strømsparingsprinsipper og heller prøvde å få verdier så ofte som mulig fra mikrokontrollerne for å oppdatere robotene kjapt nok. Opprinnelig stoppet koden å kjøre hver gang en tilkoblingsfeil oppsto, noe som var uheldig mtp. at det var vanskelig å time slik at enhetene kobler seg på rett etter hverandre. Da måtte man printe melding før en og en enhet skulle slås på, telle til 2 og deretter håpe det beste. Dette var lite robust. Man trenger ofte flere en et forsøk for å få til å koble på en enhet. Dette ble løst ved å skrive en kode for feilhåndtering ved tilkoblingsproblemer ved bruk av *try* og *expect*. Koden under *try* er koden som forsøkes å kjøres, dersom dette ikke går, kommer man inn i *except*-blokka. De aller fleste feilmeldingene som har oppstått under utviklingen av koden, som dreier seg om BLE-kommunikasjon er tatt med, slik at koden skal kjøre til tross for tilkoblingsproblemer som måtte oppstå. Det forsøkes kun å koble på enheten to ganger før man går til neste,

ettersom det er bedre enn at man prøver igjen og igjen å koble på en enhet som av en eller annen grunn ikke er tilgjengelig.

```
try:
            print("Initiating connection with white muffin holder")
            await white_muffin_holder.connect()
            next_muffin_white = await white_muffin_holder.
    read_gatt_char(muffin_holder_white_charac)
            print("FETCHED next_muffin_white: ", next_muffin_white)
            await asyncio.sleep(1)
        except BleakDBusError:
            print("Retrying to connect")
            try:
                await asyncio.sleep(2.5)
                await white_muffin_holder.connect()
                next_muffin_white = await white_muffin_holder.
    read_gatt_char(muffin_holder_white_charac)
                print("FETCHED next_muffin_white: ",
    next_muffin_white)
                await asyncio.sleep(1)
            except Exception:
                print("Reconnecting white muffin holder failed, we'
    ll try again later")
        except (BleakDeviceNotFoundError, BleakError):
            print("Device not found")
            try:
                await white_muffin_holder.disconnect()
                await asyncio.sleep(2.5)
                await white_muffin_holder.connect()
                next_muffin_white = await white_muffin_holder.
    read_gatt_char(muffin_holder_white_charac)
                print("FETCHED next_muffin_white: ",
    next_muffin_white)
                await asyncio.sleep(1)
            except Exception:
                print("Reconnecting white muffin holder failed, we'
    ll try again later")


        await white_muffin_holder.disconnect()
        await asyncio.sleep(2.5)
```

Kodeblokk 3.2: Feilhåndtering ved tilkoblingsproblemer

I kode 3.2 blir først den hvite kakeholderen forsøkt tilkoblet og lest verdien fra. Hvis dette ikke funker grunnet en BleakDBusError, som oftest betyr at RPi-en ikke klarte å koble seg på mikrokontrolleren, ventes det 2.5 ms før man prøver det samme på nytt. Dersom dette ikke funker sendes det en feilmelding. Om grunnen til at det ikke gikk første gangen, resulterer i en BleakDeviceNotFound eller BleakError, som henholdsvis tilsvarer at enheten ikke er synlig for RPi-en fordi den ikke ble frakoblet på rett måte sist gang eller at man ikke finner UUID-en på characteristicen dens, forsøkes det først

å koble fra BLE. Deretter prøves det å lese verdien på nytt. Hvis dette ikke går, sendes bare en feilmelding om at man prøver å koble seg på igjen neste runde. Derimot leses vanligvis verdien i løpet av to tilkoblingsforsøk

```
1
2         next_muffin_white = str(next_muffin_white)
3         if ("bytearray" in next_muffin_white):
4             next_muffin_white = next_muffin_white[12:16]
5
6         next_muffin_black = str(next_muffin_black)
7         if ("bytearray" in next_muffin_black):
8             next_muffin_black = next_muffin_black[12:16]
9
10
11        if (next_muffin_white != "1111"):
12            next_muffin = next_muffin_white
13
14        elif (next_muffin_black != "1111") and (next_muffin_white ==
      "1111"):
15            next_muffin = next_muffin_black
16
17        else: #next_muffin_black and next_muffin_white = 1111
18            next_muffin = "1111"
19
20        muffin_manager.publish(next_muffin)
21        await asyncio.sleep(2.5)
```

Kodeblokk 3.3: Valg og publisering av kakeholderverdi

I kodeblokk 3.3 blir først verdien fra hvit kakeholder muffin_holder_white og fra svart kakeholder muffin_holder_black gjort om til strenger. Siden den opprinnelige verdien som blir printet står på formen "bytearray(b'XXXX'" blir dette fjernet ved indeksering. Siden NACHI skal starte fra hvit kakeholder, sendes verdien der så lenge den ikke er tomt for objekter i holderen (1111). Mens dersom det er tomt for kake i hvit kakeholder, men ikke i svart kakeholder, er det denne som blir valgt. Dersom begge kakeholderne er tome, sendes bare 1111. Enda et tilfelle vil være dersom kakeholderen ikke har fått oppgitt verdi samtidig som den har blitt valgt. Da vil feilkoden 1110 sendes videre til ROS. Signalet som publiseres er next_muffin og går under topicen nextMuffin

RPi-en tilsvarer noden *raspi_gateway* i ROS. Den består av tre publishers som sender verdier til tre ulike topics. De blir publisert etter hvert som de hentes. Disse topicene kalles nextMuffin, nextCoffee og servingData. Videre hentes de av på VM-en i FlexGUI og robotene får informasjon de kan ta beslutninger på som følge av dette.

Tilkobling med ROS

For å få koden til å virke med ROS, måtte man lage en package. Pakken ble kalt raspi_gateway, mens skriptet heter ble_pub_node.pyDette ble gjort på

følgende måte:[47]

1. Gå til kildekode-mappen i catkin workspacet: `cd catkin_ws/src`
2. Lage pakken med dependencies: `catkin_create_pkg raspi_gateway std_msgs rospy`
3. Gå til dit skriptene skal ligge: `cd raspi_gateway/src`
4. Laste ned koden fra nettet: `wget https://pastebin.com/raw/XXXXXXXX`
5. Bytte fra "usynlige" Windows til Linux-endelser: `dos2unix ble_pub_node.py`
6. Gjøre skriptet executable: `chmod +x ble_pub_node.py`
7. Gå tilbake til catkin_ws: `cd /catkin_ws`
8. Bygge filen: `catkin_make`
9. Source workspacet i bashrc: `/catkin_ws_devel/setup.bash`
10. Starte ROS-master: `roscore`
11. Kjøre skriptet: `rosurn raspi_gateway ble_pub_node.py`
12. Sjekke aktive noder: `rosnode list`
13. Sjekke aktive topics: `rostopic list`
14. Følge med på verdiene som sendes på ulike topics:
    `rostopic echo nextCoffee` `rostopic echo nextMuffin` `rostopic echo servingData`

CMakeLists.txt ble generert av catkin_make og krevde ingen endringer

package.xml: De eksterne Python-bibliotekene som ikke er ROS-packages måtte legges til package.xml-filen i tillegg sammen med de andre dependency taggene der bla. rospy og std_msgs allerede står [48][49]
<build_depend>python-asyncio</build_depend>
<exec_depend>python-asyncio</exec_depend>
<build_depend>python-bleak-pip</build_depend>
<exec_depend>python-bleak-pip</exec_depend>

Oppbygning av RPi-koden

1. *Shebang*-linje med path til python-tolkeren på RPi-en
2. Importering av nødvendige biblioteker
3. Starting av event-loop og kalling av main-funksjonen. Main-funksjonen består av punktene 4-19. Event-loopen skal kjøre til den har gått gjennom koden, noe som aldri skjer fordi den inneholder en while True (punktene: 9-19)
4. Definering av MAC-adresser og characteristic UUID-ene til kaffe-, kakeholderne og serveringsbrettet
5. Definering av bleakClient-objektene
6. Sette initialverdier for signalene på enhetene

7. Initiere *raspi_gateway*-noden og publisherne
8. Avslutte alle tidligere BLE-tilkoblinger mellom RPi-en og mikrokontrollerene
9. Koble til, lese av verdi og koble fra fra hvit koppholder med feilhåndtering ved tilkoblingsproblemer
10. Koble til lese av verdi og koble fra svart koppholder med feilhåndtering ved tilkoblingsproblemer
11. Koble til, lese av verdi og med feilhåndtering for tilkoblingsproblemer
12. Omgjøring og indeksering av verdier til strenger og bestemmelse av om man sender verdien fra hvit eller svart koppholder
13. Publisering av den valgte verdien til ROS topicen nextCoffee
14. Koble til, lese av verdi og koble fra hvit kakeholder med feilhåndtering ved tilkoblingsproblemer
15. Koble til, lese av verdi og koble fra svart kakeholder med feilhåndtering ved tilkoblingsproblemer
16. Omgjøring og indeksering av verdier til strenger og bestemmelse av om man sender verdien fra hvit eller svart kakeholder.
17. Publisering av den valgte verdien til ROS topicen nextMuffin
18. Koble til, lese av verdi og koble fra serveringsbrettet med feilhåndtering ved tilkoblingsproblemer
19. Omgjøring og indeksering til streng, samt publisering av servingData til ROS
20. Programmet avsluttes ved KeyboardInterrupt av brukeren

3.4 NACHI MZ04: Oppsett

3.4.1 NACHI MZ04 spesifikasjoner

NACHI MZ04 er en industrirobot produsert av NACHI-FUJIKOSHI CORP. i Japan. Roboten er en artikulert manipulator med seks frihetsgrader. Relevante spesifikasjoner for NACHI MZ04 er lagt fram i tabell 3.10.

NACHI MZ04 spesifikasjoner

Konstruksjon		Artikulert
Antall akser		6
Drive system		AC Servo Motor
Maksimalt bevegelsesområde	Akse 1	±170°
	Akse 2	-145 ~90°
	Akse 3	-125 ~280°
	Akse 4	±190°
	Akse 5	±120°
	Akse 6	±360°
Maksimal nyttelast		4 kg
Posisjonsrepeterbarhet		±0,02 mm
Robotmasse		26 kg

Tabell 3.10: Tabell med spesifikasjoner for NACHI MZ04 [50]

3.4.2 Maskin- og programvare

NACHI MZ04 styres av en Compact Fujikoshi Daihen (CFD)-kontroller. Brukergrensesnittet til kontrollenheten er tilgjengelig på NACHIs Teach Pendant (TP). Dette er en håndkontroller som benyttes for konfigurasjon og programmering av roboten. Den opereres gjennom knapper og en touch-skjerm. For kommunikasjon med eksterne enheter er CFD-en utstyrt med en Ethernet-port og et "Mini I/O board"av typen CFD-OP150-C. Gjennom Ethernet kan kontrollenheten kobles opp til et nettverk for å motta/sende data. Mini I/O brettet benyttes for fysiske koblinger til eksterne enheter. Den består av åtte innganger og åtte utganger, av typen NPN.

Av sikkerhetsmessige hensyn er systemet utstyrt med en Robot Monitoring Unit (RMU), som til enhver tid overvåker og dobbeltsjekker robotens leddposisjoner. Det er et redundant system som garanterer sikker styring og operasjon av roboten. RMU-en klassifiseres som PL e kategori 4 i henhold til standarden ISO 13849-1. Videre møter roboten “IEC 61508 SIL3”.

3.4.3 Utgangspunkt

NACHI MZ04 var originalt fastmontert til en fikstur, avbildet i figur 3.38. Selve robotarmen var installert på topplaten av fiksturen. CFD-en og RMU-en sto på undersiden av fiksturen. Roboten var satt opp og konfigurert for et tidligere prosjekt. Dette prosjektet var en demonstrasjon hvor visittkort ble flyttet fra en kortholder til en annen, på topplaten av fiksturen. NACHI var utstyrt med en sugekopp og digitalt styrt ejektor for styring av lufttrykk til sugekoppen. Videre var det satt opp en PLS-styrt laserskanner.

Figur 3.38: Originalt oppsett for NACHI MZ04

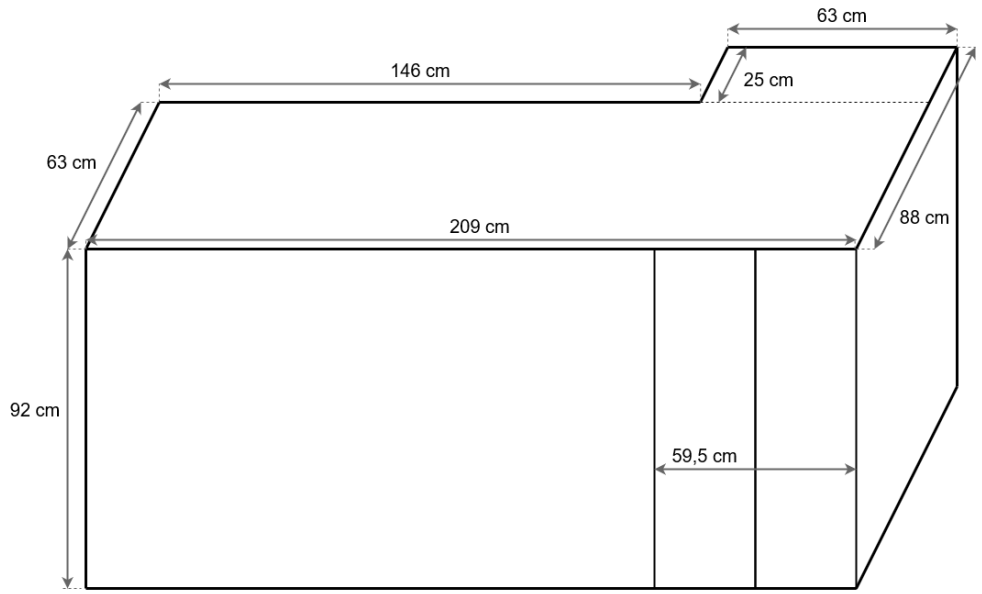
3.4.4 Planlegging og simulering

For å sikre at NACHI ble montert i en god posisjon på benken, hvor alt ville være innen rekkevidde, var det nødvendig med god planlegging. Planleggingen gikk ut på å ta mål av kjøkkenbenken og de nødvendige komponentene

(kjøkkenbenkens mål er tegnet opp i figur 3.39). Målene ble brukt til å lage en 3D-modell av NHL, og gjennomføre en simulering av bevegelsene til robotene. Det er flere enheter som skulle få plass på kjøkkenbenken, og det måtte tas hensyn til begrenset med plass. Videre har NACHI MZ04 et begrenset arbeidsområde. Følgende enheter må ha plass på kjøkkenbenken, innenfor NACHIs rekkevidde:

- NACHI MZ04 med griper
- To koppholdere, hver av dem med dimensjoner på 20,8x20,8 cm
- To kakeholdere, hver av dem med dimensjoner på 17,4x17,4 cm
- Kaffemaskin

Videre må det være plass til et kontrollpanel bestående av fire brytere, men denne skal ikke opereres av NACHI, og behøver derfor ikke være innenfor robotens rekkevidde.



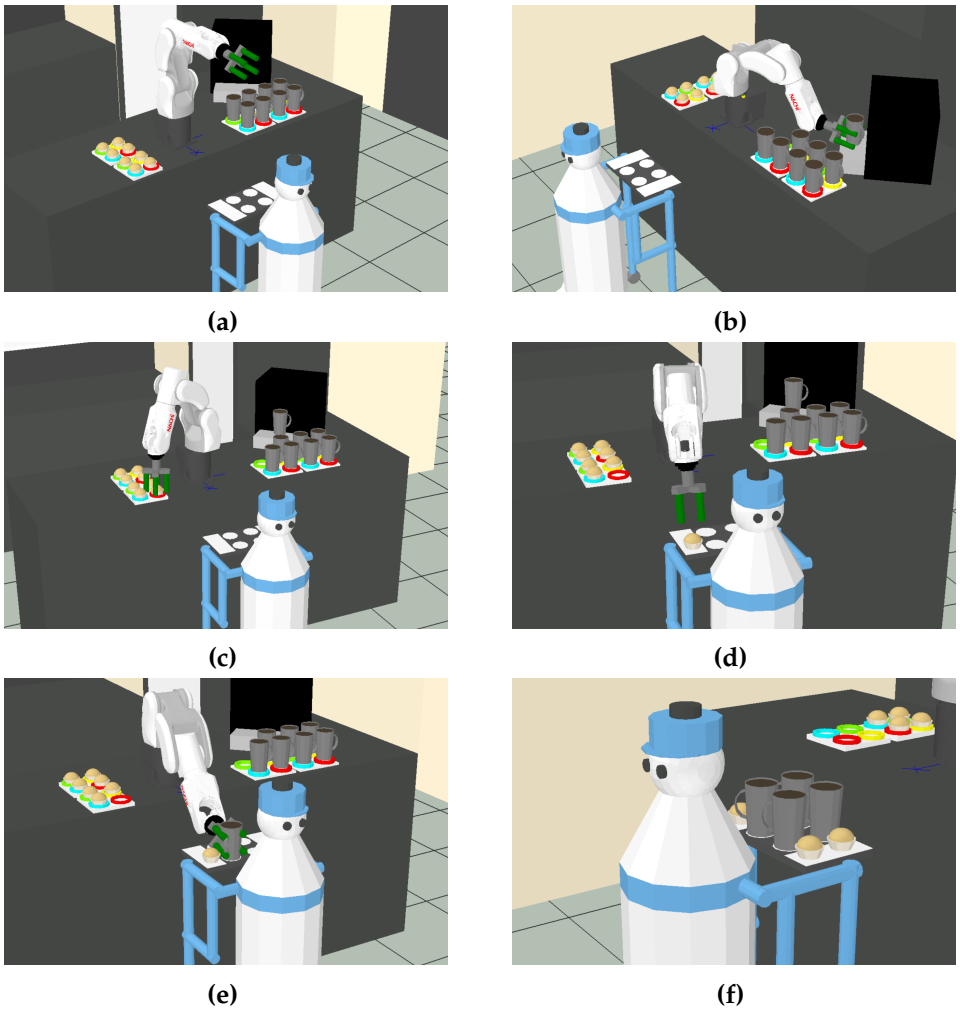
Figur 3.39: Målene til kjøkkenbenken på NHL.

FDonDesk

For å planlegge posisjonen til NACHI og de andre enhetene var det nødvendig å planlegge en sekvens og teste posisjonene i en simulering. For å gjøre dette ble NACHIs egen programvare for PC benyttet, FDonDesk. Med FDonDesk kan CFD-kontrollerens programvare styres fra PC. Gjennom FDonDesk har man tilgang til alt som er tilgjengelig når man styrer roboten direkte med CFD-kontrolleren. Dersom roboten kobles opp til PC, kan dette benyttes til fjernstyring. Det var imidlertid ikke behov for å fjernstyre roboten i dette prosjektet, og programvaren ble kun brukt til simulering. I et eget vindu var det mulig å lage 3D-objekter og danne modeller. I et annet vindu har man

alle knappene som finnes på TP-håndkontrolleren, samt skjermen. Det siste vinduet har knapper for styring og lesing av I/O signaler.

Til å begynne med ble det tatt mål av kjøkkenbenken og alle de relevante komponentene. NHLs dimensjoner ble også målt. Deretter ble alt modellert i FDonDesk. Det var allerede planer om å bruke en fingergriper med fire fingre, og det ble tatt utgangspunkt i vanlige mål for denne typen griper. Griperen ble modellert og montert på NACHI-modellen så det var mulig å ta utgangspunkt i rekkevidden til roboten når griperen er montert. Det ble prøvd ut ulike plasseringer av NACHI på kjøkkenbenken, og sjekket at alle relevante enheter var innenfor NACHIs rekkevidde. Det ble også testet hvor Kompaï med serveringsbrettet må stå for å være innenfor rekkevidde når NACHI skal plassere tilberedte varer. For å sikre at posisjoneringen ville fungere som planlagt, ble programmeringen til NACHI først planlagt i FDonDesk, og simulert. Planen er detaljert i seksjon 3.4.4.



Figur 3.40: Skjermdump fra simulering i FDonDesk.

Sekvens

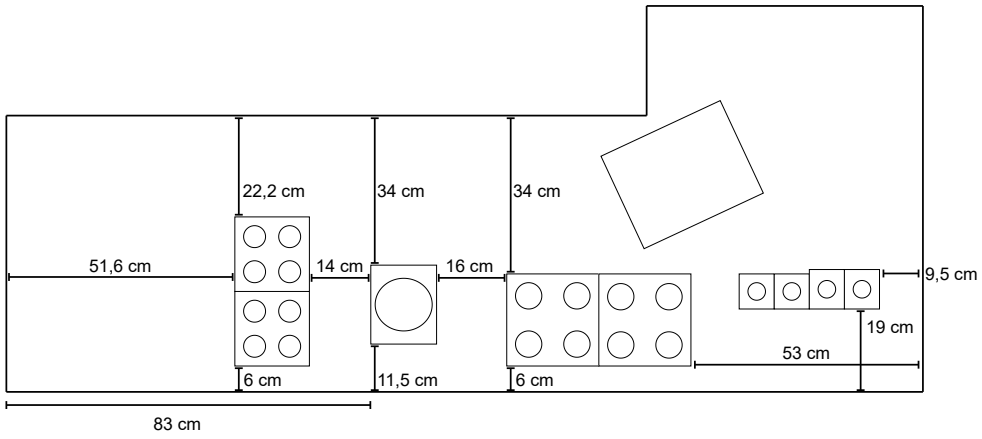
Under simuleringen ble alle posisjoner bestemt, og det ble bestemt hvordan NACHI skulle utføre sine oppgaver. Det er fire kopper med kaffe og fire kaker som serveres på én runde. Hver runde består altså av fire sekvenser for å produsere en kopp med kaffe og en kake på serveringsbrettet. Én slik sekvens er illustrert i figur 3.40, fra figur 3.40a til figur 3.40e. Figur 3.40a viser oppsettet før programmet er begynt, med fulle magasiner. I figur 3.40b har NACHI grepet den første kaffekoppen, og er i ferd med å plassere den under kaffemaskinen. Når koppen er plassert under kaffemaskinen skal NACHI trykke på knappen for å starte kaffemaskinen. Deretter løfter NACHI en kake og plasserer den på serveringsbrettet, vist i henholdsvis figur 3.40c og 3.40d. I figur 3.40e plasserer NACHI kaffekoppen på serveringsbrettet. Da er én sekvens gjennomført. Figur 3.40f viser et fullt serveringsbrett, slik det skal være etter at NACHI har fullført fire sekvenser. Følgende punkter er verdt å legge merke til:

- Kopper blir grepet fra siden, mens kaker blir grepet ovenfra. Dette er fordi koppene skal plasseres inn i kaffemaskinen, og griperen og robotarmen vil være for store for å få plass dersom de kommer ovenfra. Kakene er lave og gripes ovenfra. Hvis de gripes fra siden, vil det kun være de to nederste griperfingrene som holder kaken. Ovenfra vil alle de fire fingrene være med på å gripe kaken, slik at det er bedre grep.

- Kjøkkenbenken er delt inn i to soner, til venstre og til høyre for NACHI (fra NACHIs perspektiv, når roboten er vendt ut mot rommet og Kompaï). Til venstre står koppmagasinene og kaffemaskinen, og til høyre er kakemagasinene. Med dette oppsettet reduseres strekningen mellom kopp og kaffemaskin mest mulig. Da minimerer man risikoen for at uforutsette hendelser finner sted på veien. Videre er det en tom del av benken foran kaffemaskinen, slik at NACHI har bevegelsesrom til å interagere med kaffemaskinen. Kakemagasinene har god plass på høyre side av NACHI.

- Sekvensen begynner med at NACHI plasserer koppen i kaffemaskinen, og trykker på knappen for at maskinen skal fylle koppen med kaffe. Fra knappen på kaffemaskinen trykkes, til kaffemaskinen er ferdig med å fylle koppen med kaffe, tar det 45 sekunder. Det vil si at det må gå minst 45 sekunder fra knappen trykkes, til NACHI griper koppen for å sette den på serveringsbrettet. På bakgrunn av dette begynner sekvensen med å plassere en kopp i kaffemaskinen og starte maskinen.

3.4.5 Plassering og montering



Figur 3.41: Mål av plasseringer av komponenter på kjøkkenbenken.

NACHIs base ble montert til kjøkkenbenken med skruer og muttere, samt en egnet treplate for å styrke benkplaten i monteringsarealet. Deretter ble planlagte posisjoner for de ulike komponentene markert opp. Figur 3.41 viser komponentenes posisjoner med mål på avstander. Fra venstre står først de to kakemagasinene. Deretter står NACHI midt på benken, etterfulgt av koppmagasinene og kaffemaskinen. Helt til høyre står det fire knapper, som fungerer som systemets kontrollpanel for start, stopp, nullstilling og nødstopp av roboten.

3.5 NACHI MZ04: Utstyr

3.5.1 Valg av griper

For å gjøre NACHI i stand til å utføre de planlagte oppgavene var det nødvendig å anskaffe en griper som var i stand til å utføre dem. Det var tre ulike alternativer som virket aktuelle:

Vakuumgriper
Det var fra før godt grunnlag for å benytte vakuum, på bakgrunn av systemets originale utforming. En enkel sugekopp ville imidlertid ikke hatt nok kraft til å løfte en full kopp, så det måtte isåfall blitt et oppskalert vakuumsystem. Det ble besluttet at vakuum ville vært problematisk i å løfte kaker med løse partikler, og at det ville vært vanskelig å løfte objekter med sylindrisk form (kopper).

Servoelektrisk griper
Det var fra før en servoelektrisk griper liggende på lageret, av typen Schunk 50-110. Den har innebygd kraftkontroll og ethernet-tilkobling. Umiddelbare ulemper er blant annet størrelsen, da den har en total lengde og bredde på 16x14,5 cm. Det er begrenset plass på benken, og roboten har et relativt lite arbeidsområde. Med såpass små marginer vil dimensjonene til griperen by på utfordringer. I tillegg har griperen en egenvekt på 1,2 kg, noe som er i meste laget. Denne griperen er best utformet for industrielle applikasjoner hvor det skal gripes mer regulære objekter.

Figur 3.42: Fotografi av Schunk 50-110 servoelektrisk griper

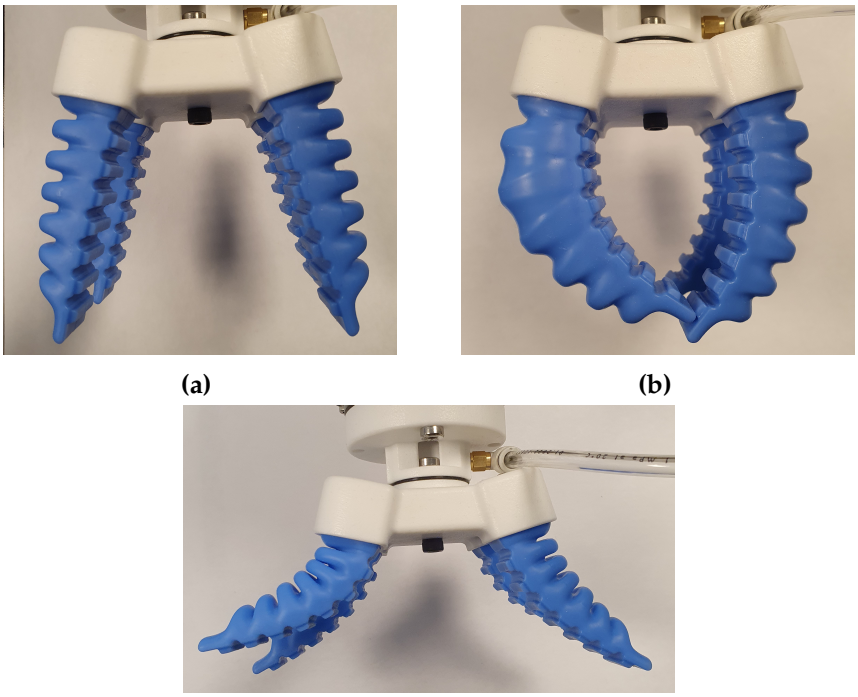
Pneumatisk griper
Det siste alternativet, og det som til slutt ble valgt, er en pneumatisk griper. Det har fordeler i at bevegelsen kun styres av lufttrykk. Den skiller seg fra vakuumgriperen i det at den bruker både vakuum og trykkluft. Det gir mer fleksibilitet når det skal gripes irregulære objekter.

3.5.2 4 Finger Parallell SoftGripper

Den pneumatiske griperen som er benyttet i dette prosjektet er en SoftGripper. Denne er levert av den tyske leverandøren SoftGripping, som fokuserer

på myke gripere styrt av lufttrykk. SoftGripperens tilstand styres eksklusivt av lufttrykket som tilføres griperen. Den blir indirekte styrt gjennom de elektriske signalene som går til lufttrykkskomponentene. SoftGripperen tåler et maksimalt lufttrykk på 1 bar, og et minimalt lufttrykk på -0,3 bar. Griperen er utformet på en måte som ligner en menneskehånd, og den egner seg derfor godt til oppgaver som vanligvis blir utført av menneskehender. Utformingen til griperen gjør at den er i stand til å gripe objekter av ulike størrelser, former og materialer. Den er altså i stand til å gripe både kopper og kaker, og eliminerer behovet for verktøybytte. Griperen har følgende egenskaper:

- Den har tre posisjoner: Grip (trykkluft, figur 3.43b), Relax (0 bar, figur 3.43a) og Release (vakuum, figur 3.43c).
- Den er lufttrykkoperert, og dermed i stand til å gripe både myke og harde objekter uten å skade dem.
- Silikonmaterialet til fingrene er FDA-godkjent for håndtering av mat.
- Utformingen til fingrene, i kombinasjon med materialet, bidrar til god friksjon i kontakt med de fleste materialer. Dette styrker gripeevnen.



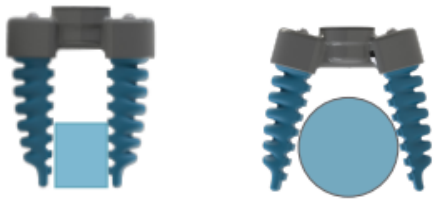
Figur 3.43: Griperen i posisjon a) Relax, b) Grip, c) Release

SoftGripperen leveres i mange ulike konfigurasjoner. De ulike valgmulighetene må velges slik at griperen blir best mulig tilpasset bruksområdet. Relevante beslutninger gikk på antall fingre, vinkel, orientering, adapter og materiale. Resten av denne seksjonen legger fram den endelige konfigurasjonen, og

redegjørelse for beslutningene som er tatt.

Antall fingre: 4
Griperen kan leveres med 2 til 8 fingre. Med to fingre ville griping av kopper og kake blitt vanskelig, da objektene kan ha lett for å rotere når det blir grepet på kun to punkter. Fordi det skal løftes fylte kaffekopper er det spesielt viktig at griperen har godt grep, også med tanke på vekten av en fylt kaffekopp. Seks fingre eller mer ville vært overflødig, da det ikke skal gripes objekter med en lengde som ville benyttet alle fingrene. Fire fingre er optimalt.

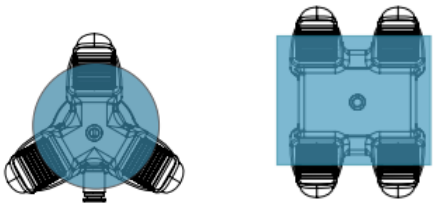
Vinkel: 15°
Griperen kan leveres med fingervinkel på 0°(figur 3.44a) eller 15°(figur 3.44b). Uansett vinkel vil fingrene være i stand til å møte hverandre når de griper. På bakgrunn av koppenes størrelse, vil 0°medføre at det blir knapp plass mellom fingrene. Det er derfor valgt 15°vinkel.



(a) **(b)**

Figur 3.44: Vinkelkonfigurasjon, 0°eller 15°(fra SoftGripper katalogen [51]).

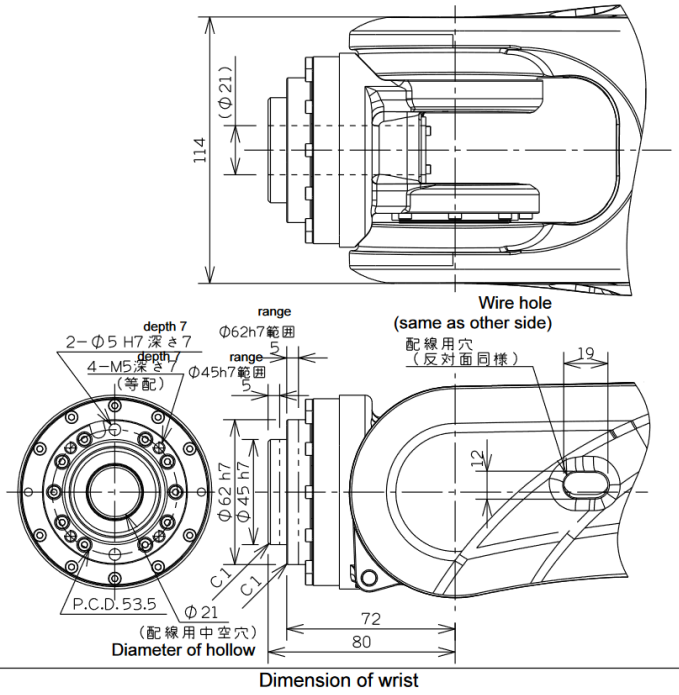
Orientering: Parallell
Det måtte velges mellom sentrisk og parallell griper. Ved sentrisk konfigurasjon vil alle fingrene bevege seg langsmed linjer som krysser hverandre i griperens midtpunkt (figur 3.45a). Ved parallell konfigurasjon vil fingrene som står ovenfor hverandre bevege seg i linje med hverandre, slik at hvert par med fingre har sitt eget midtpunkt (figur 3.45b). For å gripe runde kaker vil en sentrisk griper være best egnet, men for kopper med tilnærmet sylindrisk form som gripes fra siden vil ikke en sentrisk griper være egnet. En parallell griper vil være optimal for koppene, og de vil fungere med kakene. Koppene har også høyest prioritet med tanke på vekt, og at dårlig grep på koppene vil ha størst konsekvenser.



(a) **(b)**

Figur 3.45: Orientering, sentrisk eller parallell (fra SoftGripper katalogen [51]).

Adapter: Tilpasset
Griperen må monteres til NACHI. Standard valgmuligheter er ISO 9409-A50-R, ISO 9409-A31,5-R og Delta Adapter 31,5 - G3/8. Adapteren er 3D-printet av produsenten og det var derfor ikke noe problem å få en spesialtilpasset adapter. Målene som ble sendt til produsenten er i figur 3.46.



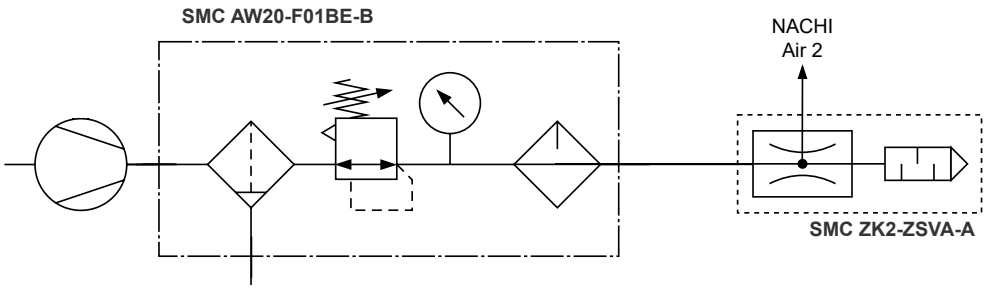
Figur 3.46: Mål på installasjonsflaten (fra NACHI manipulator manual [50]).

Materiale: Hygienisk (FDA-godkjent)
Fingrene kan leveres i to ulike silikonmaterialer: normalt materiale eller FDA-godkjent materiale for håndtering av mat. Fordi griperen skal benyttes til å håndtere kaker direkte, er det valg FDA-godkjent materiale.

3.5.3 Pneumatisk system

Originalt system

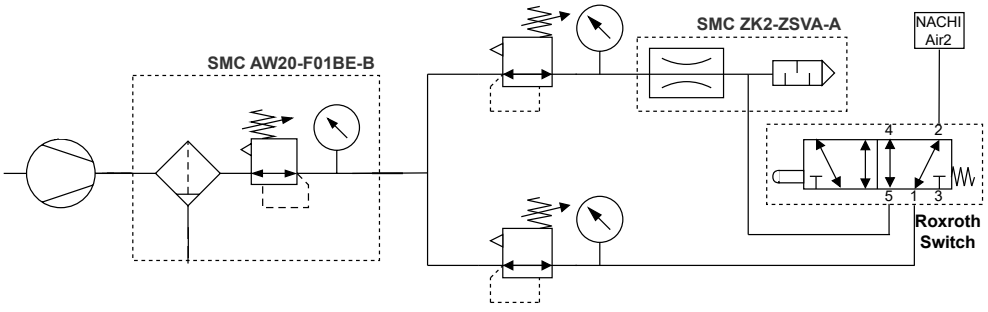
Som lagt fram i seksjon 3.4.3, var NACHI MZ04 allerede tilkoblet et system for lufttrykk. Dette systemet besto av en kompressor, en reguleringsventil med filter og en ejektor. Pneumatisk diagram for det gamle systemet er avbildet i figur 3.47. Griperen var en enkel sugekopp, som benyttet seg av vakuumet generert av det pneumatiske systemet. Vakuumet var styrt digitalt av I/O-portene til NACHI. Valget av en pneumatisk griper gjorde at det var mulig å benytte seg av de eksisterende lufttrykkomponentene og følge samme oppsett for digitale signaler.



Figur 3.47: Det originale oppsettet for pneumatikk

Videreutvikling av originalt system

Det måtte utvikles et pneumatisk system som er i stand til å utnytte griperens funksjonalitet. Mange pneumatiske komponenter var tilgjengelige på lageret. Det ble utviklet et manuelt pneumatisk system til å begynne med, på bakgrunn av at griperens funksjon er kritisk i programmering og testing av resten av systemet. Det var regnet med at det ville ta tid å få det komplette pneumatiske systemet på plass. Det manuelle systemet la grunnlaget for hvordan det digitalt styrte systemet skulle se ut.



Figur 3.48: Manuelt oppsett for pneumatisk system for griper

Det pneumatiske systemet må møte følgende krav:

- Den må fungere med kompressoren. Siden kompressoren ikke kan gå kontinuerlig over lengre tid, er det kritisk at luften blir stoppet når griperen ikke trenger trykkluft eller vakuum.
- Det må være mulig å sende vakuum og trykkluft separat fra hverandre.
- Systemet skal kunne styres av de 24 V signalene på I/O portene til NACHI.
- Det må operere mellom -0,3 og 1,0 bar på utgangen som går til griperen.

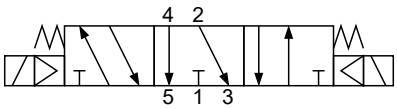
Komponenter

Det nye pneumatiske systemet oppfyller alle kravene for å fungere med griperen og NACHI. Den består av en kompressor, tre regulatorventiler (hvorav en av dem har filter), 3 magnetventiler (en 5-ports og to 3-ports), en vakuumejektor og en trykkluftkontroller.

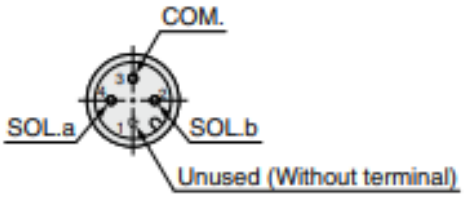
Kompressor: Faller AS18
Som kilde til trykkluft benyttes det en ensylindret stempelkompressor, med automatisk start og stopp på henholdsvis 3 og 4 bar. Fordi griperen opererer på lavt trykk, er dette mer enn nok. Den er portabel og veier 3,6 kg, slik at den vil passe inn selv med begrenset plass.

Regulatorventil: SMC AW20-F01BE-B
En regulatorventil med innebygd filter og trykkmåler. Den er første leddet luften fra kompressoren går gjennom. Det sørger for at luften fra kompressoren filtreres før den går inn i resten av systemet, slik at de andre komponentene er beskyttet fra partikler. Regulatoren er stilt inn på 2 bar.

5-ports magnetventil: SMC SY5400-5UI (Hovedventil)
En magnetventil med 5 porter, av typen normalt lukket (figur 3.49a). Inngangen mottar den filtrerte luften fra regulatoren. Den styres av digitale signaler fra NACHIs I/O. Dersom det ikke er påtrykket et signal, vil ventilen være lukket. Den vil da sørge for å opprettholdet trykket på utgangen til kompressoren, slik at den ikke går på tomgang. Ventilen vil sende luft til utgang 4 når SOL.a påtrykkes signal, og utgang 2 når SOL.b påtrykkes signal. Den er koblet med M12-kobling illustrert i figur 3.49b.

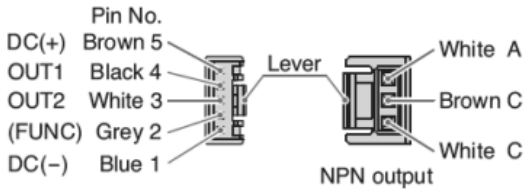


(a) Diagram for 5-ports magnetventil



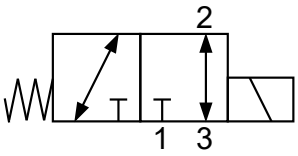
(b) Pinout for 5-ports magnetventil. Fra manualen [52]

Vakuumejektor: SMC ZK2-ZSVA-A (ZK2)
Vakuumejektoren mottar luft fra en regulatorventil koblet til utgang 2 på 5-ports magnetventilen. Den er koblet opp i henhold til figur 3.50, hvor FUNC er den inngangen som styrer vakuumet av og på.

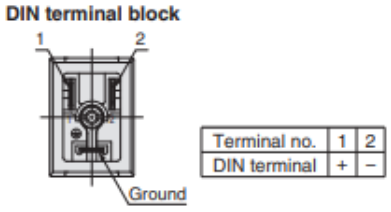


Figur 3.50: Pinout for vakuumejektor. Hentet fra manualen [53]

3-ports magnetventil SMC VT307(V)-5DZ1-01F-Q (Trykkventil og Vakuum-ventil)
Det er satt opp to 3-ports magnetventiler, hvor den ene er egnet for trykkluft og den andre for vakuum. De har som hensikt å sikre at det kun sendes luft til NACHI fra én utgang om gangen, og forhindrer tilbakestrømning av luft.



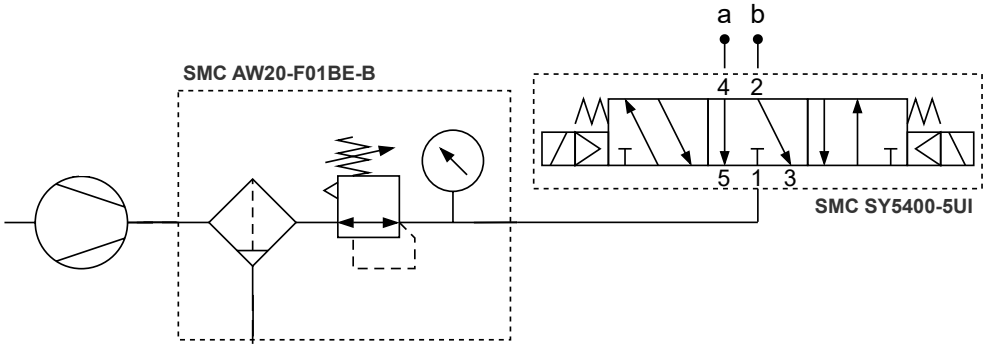
(a) Diagram for 3-ports magnetventil



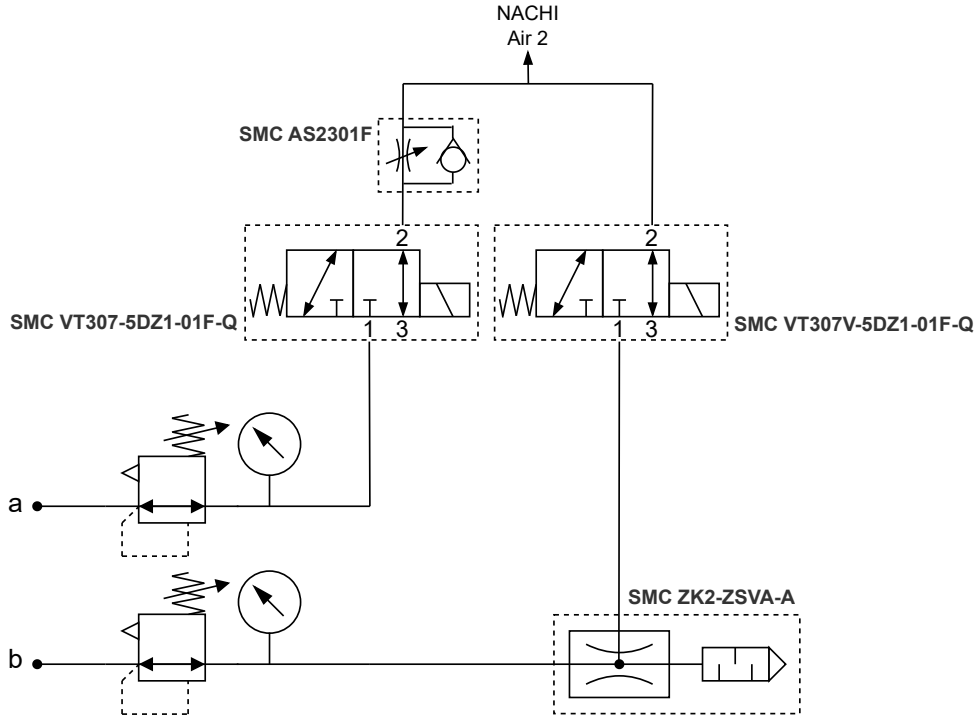
(b) Pinout for 3-ports magnetventil. Hentet fra manualen [54]

Trykkluftkontroller: SMC AS2301F
På utgangen til trykkluftventilen (3-ports) er det festet en trykkluftkontroller som luften på utgangen går gjennom. Griperen opererer på lavt trykk, og overtrykk kan skade den. Trykkluftkontrolleren sørger for å slippe ut litt luft dersom trykket stiger til for høye verdier, altså reduserer den det dynamiske avviket. Dette skjer når ventilen åpner seg etter å ha vært stengt.

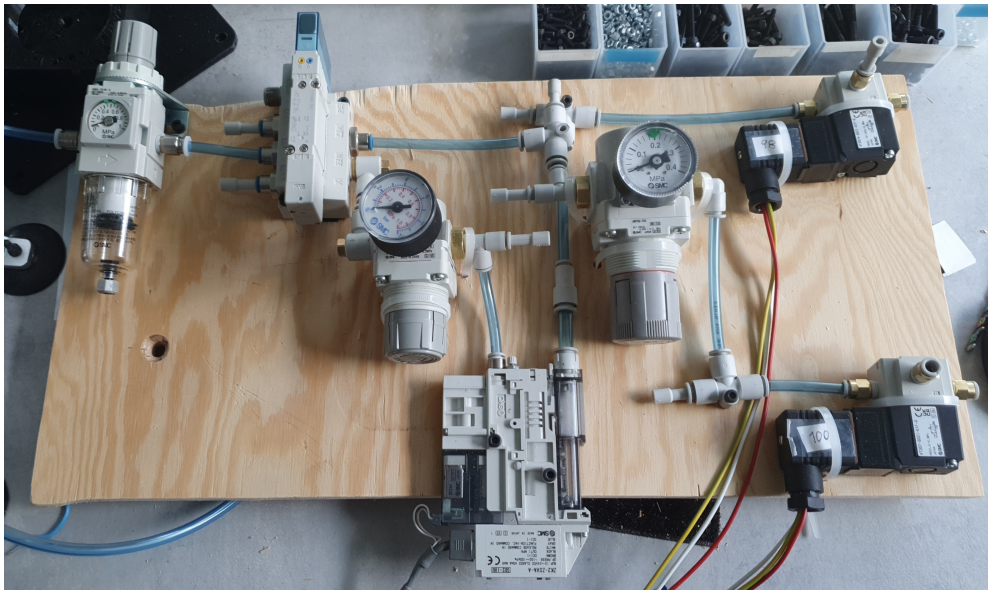
Oppkobling



Figur 3.52: Pneumatisk diagram fra kompressoren til magnetventil med 5 porter



Figur 3.53: Pneumatisk diagram etter magnetventil med 5 porter



Figur 3.54: Fotografi av det ferdige pneumatiske systemet

Komponenter som håndterer lufttrykk skal helst fastmonteres for å sørge for at de ikke flytter på seg når de påtrykkes lufttrykk. Når alle komponentene var klare for å monteres til et endelig system, ble de posisjonert på en treplate og fastmontert. Platen ble plassert i en kjøkkenskuff rett over CFDen, slik at den kan skjules, men likevel være lett tilgjengelig. I samme høyde er det en hylle til venstre hvor kompressoren står. Plasseringen gir kort vei for ledninger/rør å kobles til I/O/kompressor.

3.5.4 Variabler, innganger og utganger

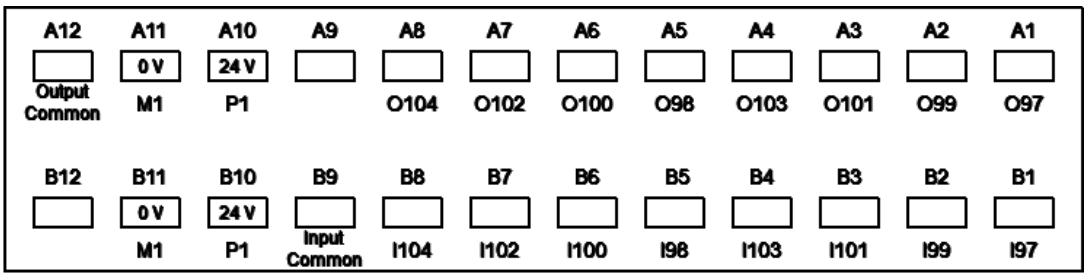
Det er benyttet variabler av forskjellige typer for kommunikasjon og styring av eksterne enheter. For dette prosjektet er det benyttet de typene som er lagt fram i tabell 3.11

Variabeltype	Beskrivelse
Heltallsvariabler	Det benyttes globale heltallsvariabler til egendefinerte signaler som kan leses og skrives i FlexGUI. For eksempel signalet for neste kopp-posisjon.
Inngangsvariabler	Inngangsvariabler styres av fysiske signaler på inngangsporter og den interne PLS-en. For eksempel knappetrykk.
Utgangsvariabler	Utgangsvariabler styrer fysiske signaler på utgangsporter og valgfrie utgangssignaler om robotens status. For eksempel påtrykking av 24 V signal til ekstern enhet.
Faste inngangsvariabler	Faste inngangsvariabler kan kun leses, og er knyttet opp til faste inngangsporter, ofte system-/sikkerhetsrelatert
Faste utgangsvariabler	Faste utgangsvariabler kan kun leses, og er knyttet opp til faste utgangsporter, ofte system-/sikkerhetsrelatert

Tabell 3.11: Oversikt over variabeltyper benyttet av NACHI.

3.5.5 Mini I/O

NACHI er utstyrt med et Mini I/O-brett med til sammen 24 porter. Diagram for disse er tegnet opp i figur 3.55.



Figur 3.55: NACHI CFD Mini I/O Board, innganger og utganger [55]

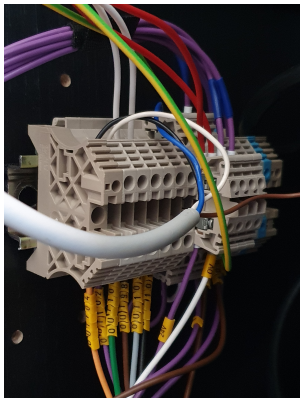
Port	Navn	Ekstern tilkobling
A1	O97	ZK2: Function vac. cmd in
A2	O99	ZK2: Release cmd in
A3	O101	IO Terminal: O101
A4	O103	IO Terminal: O103
A5	O98	IO Terminal: O98
A6	O100	IO Terminal: O100
A7	O102	IO Terminal: O102
A8	O104	IO Terminal: O104
A10	P1 (24 V)	ZK2: DC+
A9		IO Terminal: 0 V
A11	M1 (0 V)	
A12	Output Common	
B1	I97	ZK2: Out1
B5	I98	IO Terminal: I98
B6	I100	IO Terminal: I100
B7	I102	IO Terminal: I102
B8	I104	IO Terminal: I104
B11	M1 (0 V)	ZK2: DC-
B9	Input Common	IO Terminal: 24 V
B10	P1 (24 V)	
B12		

Tabell 3.12: Tabell over koblinger av innganger og utganger på Mini I/O-brettet.

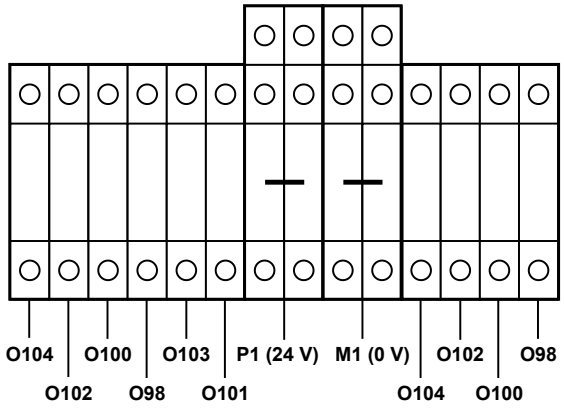
I tabell 3.12 er det laget en oversikt over alle portene på Mini I/O-brettet. Her refererer ZK2 til vakuumejektoren som var koblet opp fra før. Andre koblinger som eksisterte fra før var koblingen mellom A9, A11 og A12, samt koblingen mellom B9, B10 og B12. Merk at rekkefølgen på portene er byttet i henhold til disse sammenkoblingene. Inngangsport B2 til B4 er utelatt, da de er koblet til komponenter som forblir ubrukt i denne sammenhengen. Output Common er koblet til 0 V, og Input Common er koblet til 24 V. Det vil si at utgangssignalene er 0 V og inngangssignalene er 24 V.

Portene er koblet til I/O-brettet med en kontakt av typen FCN-361J024-AU, fra produsenten Fujitsu. Den er laget for at portene skal loddes. For lettere tilgang til de ledige I/O portene, 24 V og 0 V, ble det loddet ledninger til de relevante portene, og satt opp en terminal hvor ledningene kunne festes i koblingsklemmer. Dette er IO terminalen. Det er denne som refereres til i tabell 3.12. Terminalen er forklart i neste seksjon, 3.5.5.

IO Terminal



(a) Fotografi av IO terminalen



(b) Oversikt over signaler inn på IO terminalen.

IO terminalen består av koblingsklemmer på en DIN-rail. Hver av koblingsklemmene er koblet til en port på Mini I/O-brettet. Fra IO terminalen er komponentene koblet slik som i tabell 3.13. Utgangssignalene til IO terminalen styrer det pneumatiske systemet, detaljert i seksjon 3.5.5. Inngangsportene på IO terminalen mottar signaler fra kontrollpanelet, detaljert i seksjon 3.5.5.

Kobling av komponenter

Komponent	Komponentside	Terminalside	
Hovedventil	SOL.a	O104	Pressure Port
	SOL.b	O102	Vacuum Port
	COM	24 V	
Trykkventil	1	24 V	
	2	O100	Pressure Valve
	Ground	0 V	
Vakuumventil	1	24 V	
	2	O98	Vacuum Valve
	Ground	0 V	
Startknapp (S2)	1	I98	Start Button
	2	0 V	
Stoppknapp (S1)	1	I100	Stop Button
	2	0 V	
Resetknapp (S3)	1	I102	Reset Button
	2	0 V	

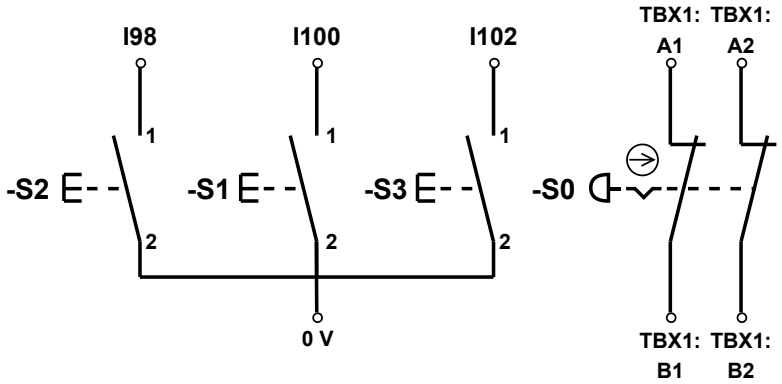
Tabell 3.13: Tabell over komponentenes koblinger med IO terminalen.

Styring av pneumatisk system

Det pneumatiske systemet styres av utgangsvariabler. Disse kan settes til høy eller lav i programkoden. For at griperen skal få trykkluft, må signalene for Pressure Port og Pressure Valve være på. For at den skal få vakuum må Vacuum Port, Vacuum ON (ZK2) og Vacuum Valve være på. Signalene for trykkluft og vakuum skal ikke være på samtidig. For å binde trykkluft og vakuumsignalene sammen, ble det opprettet et kombinert utgangssignal for hver av dem.

Styring av kontrollpanel

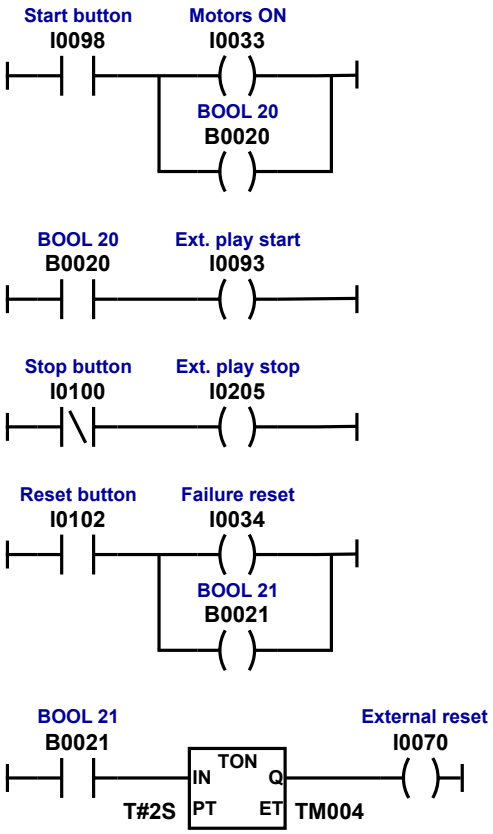
Kontrollpanelet består av fire brytere. Bryterne for start, stopp og reset, er brytere betjent ved trykk med automatisk retur og sluttekontakt. Den siste bryteren er nødstoppbryteren. Denne har to brytekontakter og manuell tilbakestilling. Den er ikke koblet opp til de vanlige IO-portene, men til koblingsklemmene på NACHIs RMU sikkerhetssystemer (TBX1). Der er det to dedikerte innganger og utganger for ekstern nødstoppbryter [56]. Så lenge nødstoppbryteren er koblet opp til disse portene, er det ikke nødvendig å gjøre noe annet for at bryteren skal fungere. Koblingsskjema for kontrollpanelet er i figur 3.57.



Figur 3.57: Koblingsskjema for kontrollpanel.

For logikken til kontrollpanelet er det satt opp et program i NACHIs interne PLS, illustrert i figur 3.58. Startknappen aktiverer først inngangssignalet for å skru på motorene. Ved andre trykk blir Ext. play start satt til høy. Denne funksjonen kjører programmet dersom roboten er i auto-modus. Den kjører det programmet som er bestemt av Program selection bits. Program selection bits er koblet opp til 8 ulike inngangssignaler som blir satt til høy eller lav i den interne PLS-en, og til sammen lager en binær verdi. Den binære verdien bestemt av inngangssignalene tilsvarer programnummeret til programmet som vil kjøres. Videre er stoppknappen koblet til Ext. play stop, som stopper

programmet, men ikke motoren. Roboten gjenopptar programmet der det ble stoppet dersom Ext. play start blir påtrykket etter stopp. Resetknappen er knyttet til Failure reset, som gjør at feilmeldinger kan nullstilles uten å bruke håndkontrolleren. Den er også knyttet til en bool-verdi, som vil aktivere External reset dersom den holdes høy i 2 sekunder. External reset vil nullstille programmet slik at det kan begynnes fra begynnelsen av når start trykkes, istedenfor å gjenoppta der den slapp.



Figur 3.58: Diagram av PLS-programmet for kontrollpanelet.

3.6 NACHI MZ04: Programmering

3.6.1 Variabler

For å få til styring av NACHI-programmet fra FlexGUI er det benyttet heltallsvariabler som kan skrives og leses av både NACHI og FlexGUI. Disse er redegjort i tabell 3.14.

NACHI integer variabler

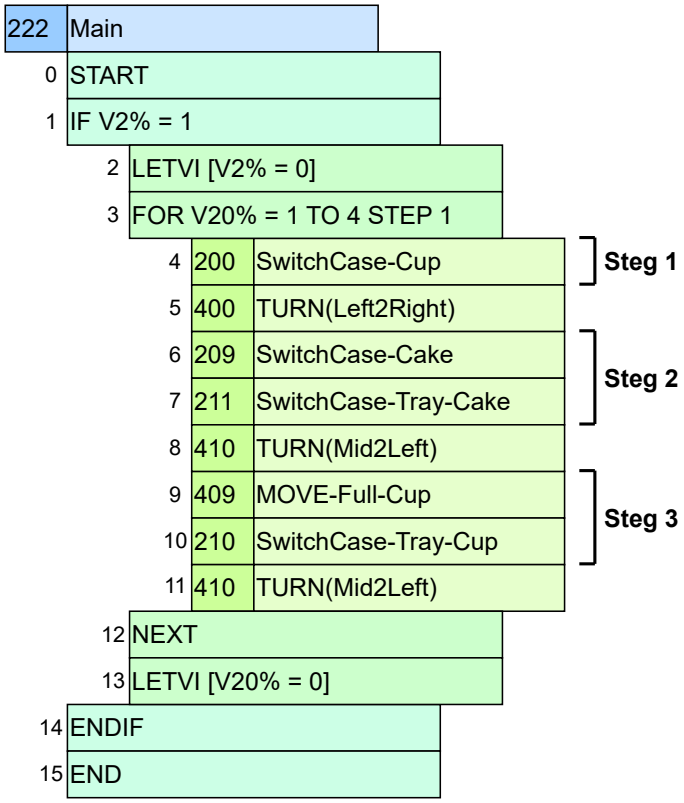
Navn	Internt	Verdier	Beskrivelse
EnterLoop	V2	[0, 1]	Må være 1 for at programmet skal kjøre. Er 1 når StartRequest og KompaiHello er 1, og TrayFinished er 0.
StartRequest	V3	[0,1]	Startsignal fra brukeren, blir 1 når startknappen trykkes i brukergrensesnittet.
TestMode	V4	[0,1]	Brukes kun til testing når deler av programmet skal hoppes over.
NextCupPos	V10	[0,8]	Hvilken posisjon neste kopp står i.
NextCakePos	V11	[0,8]	Hvilken posisjon neste kake står i.
Iteration	V20	[0,4]	Hvilken iterasjon programmet er på. Settes til 0 når alle iterasjoner er kjørt. Det er 4 iterasjoner i én runde.
WaitingForKompai	V22	[0,1]	Settes til 1 når StartRequest er 1, men KompaiHello er 0.
KompaiHello	V23	[0,1]	Er 1 når Kompai står klar med serveringsbrettet i riktig posisjon.
TrayFinished	V24	[0,1]	Settes til 1 når serveringsbrettet er klart til å serveres. Stilles tilbake til 0 når KompaiHello går til 0.

Tabell 3.14: Tabell med heltallsvariablene som benyttes i FlexGUI og NACHIs programvare.

3.6.2 Hovedprogram

Hovedprogrammet består av en if setning som kjører hovedloopen dersom klarsignalet EnterLoop mottas fra FlexGUI. Så lenge dette signalet ikke er mottatt, vil programmet loope uten å gjøre annet enn å oppdatere statusvariabler. Ved startforespørsel vil programmet vente på signal om at Kompaï er klar med serveringsbrettet i riktig posisjon. Når FlexGUI registrerer at Kompaï er klar vil det klarsignalet sendes til NACHI som da kjører hovedloopen. Hovedprogrammet er illustrert i figur 3.59. Noen steg som kun innebærer

endring av statusvariabler brukt av FlexGUI er utelatt av figuren (for detaljer se tabell **??**).



Figur 3.59: SLIM-sekvens for hovedprogrammet.

Det er markert tre steg i figuren:

Steg 1 Plukker kopp fra koppmagasinet, plasserer koppen i kaffemaskinen og trykker på knappen for å starte kaffemaskinen.

Steg 2 Plukker kake fra kakemagasinet, og plasserer kaken på serveringsbrettet.

Steg 3 Plukker den fulle kaffekoppen fra kaffemaskinen og plasserer den på serveringsbrettet.

Disse stegene befinner seg inne i en for-loop som vil gjennomføre fire iterasjoner. Switchcase-programmene brukes for å bestemme hvilket program som skal kjøres ut ifra variablene. SwitchCase-Cup bestemmes av verdien til NextCup og SwitchCase-Cake av verdien til NextCake. SwitchCase-Tray-Cake og SwitchCase-Tray-Cup bestemmes av verdien til iterasjonsvariabelen, altså vil kopper og kaker alltid plasseres på serveringsbrettet i den samme rekkefølgen.

3.6.3 Plukking og plassering

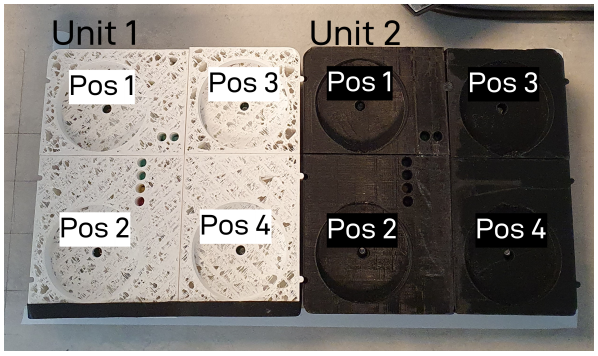
Plukking av kopper og kaker fra magasinene følger en gitt prioriteringsrekkefølge. Posisjonene er navngitt etter enhet (magasin 1 eller 2) og posisjon på enheten. Altså er første plass på første enhet Unit 1 Pos 1, tredje plass på andre enhet Unit 2 Pos 3, etc.. Prioriteringen går som i tabell 3.15. Rekkefølgen for prioriteringen er samme for både kopp- og kakemagasin.

Prioritering	Posisjon
1	Unit 1 Pos 1
2	Unit 1 Pos 2
3	Unit 1 Pos 3
4	Unit 1 Pos 4
5	Unit 2 Pos 1
6	Unit 2 Pos 2
7	Unit 2 Pos 3
8	Unit 2 Pos 4

Tabell 3.15: Oversikt over posisjonsnavn og prioritering.

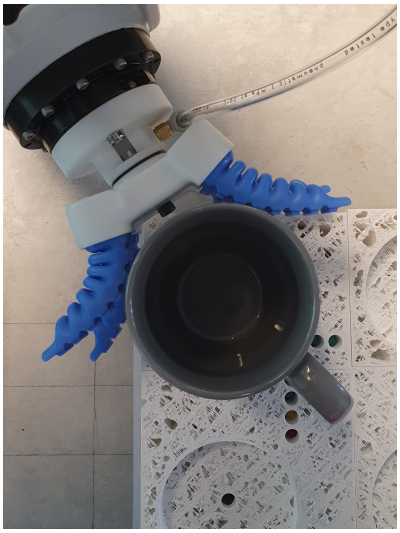
Plukking av kopper

Plukking av kopper fra koppmagasinet følger prioriteringsrekkefølgen ovenfor. Posisjonene er illustrert i figur 3.60. For koppmagasinet er det kritisk at prioriteringen følges for å unngå kollisjoner, da koppene gripes fra siden. For å ha god nok rekkevidde til å nå koppene lengst unna, må magasinet stå forholdsvis nært NACHI. Dette medfører at de første to koppene må gripes fra en vinkel. Griping fra front er vist i figur 3.61a, og fra siden i figur 3.61b.



Figur 3.60: Oversikt over posisjoner for kopper i magasin.

(a) Griping av kopp fra front.



(b) Griping av kopp fra siden.

Navn	Nr	Beskrivelse
Cup-Unit1-Pos1	300	Plukker kopp fra Unit 1 Posisjon 1
Cup-Unit1-Pos2	301	Plukker kopp fra Unit 1 Posisjon 2
Cup-Unit1-Pos3	302	Plukker kopp fra Unit 1 Posisjon 3
Cup-Unit1-Pos4	303	Plukker kopp fra Unit 1 Posisjon 4
Cup-Unit2-Pos1	304	Plukker kopp fra Unit 2 Posisjon 1
Cup-Unit2-Pos2	305	Plukker kopp fra Unit 2 Posisjon 2
Cup-Unit2-Pos3	306	Plukker kopp fra Unit 2 Posisjon 3
Cup-Unit2-Pos4	307	Plukker kopp fra Unit 2 Posisjon 4

Tabell 3.16: Oversikt over programmer for plukking av kopper

NACHI har ett program for hver av kopposisjonene, notert i tabell 3.16. Hvilken kopp som gripes blir bestemt av NextCup-variabelen, styrt fra Flex-GUI. Alle programmene for å gripe kopper følger samme struktur:

- Gjør klar til å gripe kopp ved å posisjonere i henhold til om koppen skal gripes fra side eller front.
- Griper Release.
- Flytt griperen til koppen som skal plukkes. Den siste bevegelsen må følge endeffektors Z-akse for å komme rett på koppen.
- Griper Grip.
- Løft koppen ved å bevege griperen i positiv retning langsmed robotens Z-akse.
- For kopper plassert forbi kaffemaskinen, trekk armen tilbake så koppen ikke kolliderer med kaffemaskinen på vei dit.

Programmet for å gripe koppene er plassert i starten av et program som

sørger for at koppen blir plassert riktig i kaffemaskinen, og skrur på kaffemaskinen når den er plassert der. Fordi de to første koppene gripes fra en annen vinkel enn de andre, må de plasseres litt ulikt i kaffemaskinen slik at de ender opp med å bli plassert likt.

A	PREP-Cup-**V10%**	
0	START	
1	REM ["PREP-Cup-**V10%**"]	
2	407	START-Pos-Left
3	**B**	Cup-**V10%**
4	**C**	PLACE-Cup-Machine-**Y**
5	405	PUSH-Button
6	407	START-Pos-Left
7	END	

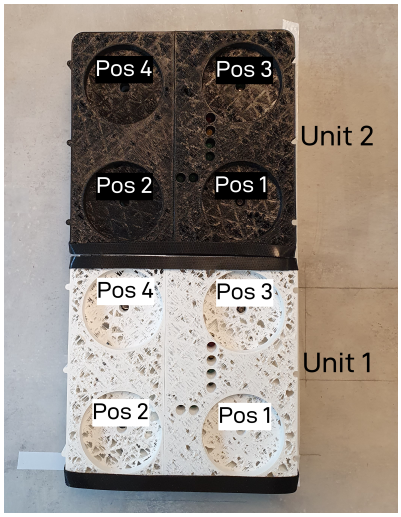
Figur 3.62: Oppsett av SLIM-sekvens for programnr. 201 til 208.

Parametre				
V10%	**A**	**B**	**C**	**Y**
1	201	300	406	SIDE
2	202	301	406	SIDE
3	203	302	404	FRONT
4	204	303	404	FRONT
5	205	304	404	FRONT
6	206	305	404	FRONT
7	207	306	404	FRONT
8	208	307	404	FRONT

Tabell 3.17: Tabell med parametre for figur 3.62

Plukking av kaker

Plukking av kaker følger samme regler som plukking av kopper. De skiller seg imidlertid fra hverandre i det at kakene gripes ovenfra, og kollisjonsrisikoen er derfor langt mindre. Likevel vil plukking av kaker følge samme prioriteringssystem. Rekkefølgen er også lik, med posisjonsnavnene lagt fram i figur 3.63. Ved å følge denne rekkefølgen vil roboten alltid gripe den kaken som står nærmest serveringsbrettet, slik at den alltid vil ha kortest mulig rute til serveringsbrettet. Dermed optimaliseres syklustiden og risikoen for å glippe vil være mindre.



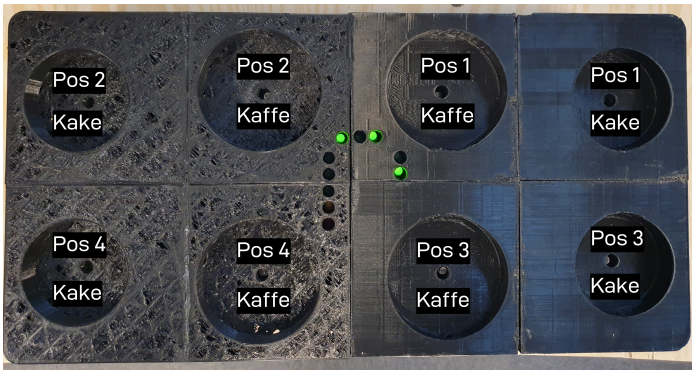
Figur 3.63: Oversikt over posisjoner for kaker i magasin.

Navn	Nr	Beskrivelse
Cup-Unit1-Pos1	300	Plukker kake fra Unit 1 Posisjon 1
Cup-Unit1-Pos2	301	Plukker kake fra Unit 1 Posisjon 2
Cup-Unit1-Pos3	302	Plukker kake fra Unit 1 Posisjon 3
Cup-Unit1-Pos4	303	Plukker kake fra Unit 1 Posisjon 4
Cup-Unit2-Pos1	304	Plukker kake fra Unit 2 Posisjon 1
Cup-Unit2-Pos2	305	Plukker kake fra Unit 2 Posisjon 2
Cup-Unit2-Pos3	306	Plukker kake fra Unit 2 Posisjon 3
Cup-Unit2-Pos4	307	Plukker kake fra Unit 2 Posisjon 4

Tabell 3.18: Oversikt over programmer for plukking av kaker

Plassering på serveringsbrett

På serveringsbrettet er det lagt opp til at en kaffekopp og en kake som tilberedes under samme syklus, plasseres ved siden av hverandre. Ellers er rekkefølgen bestemt med hensyn til at kaffekoppene skal ha minst mulig kollisjonsrisiko. De bakerste plasseres først, så de ikke er blokkerende for de ledige plassene. Videre plasseres de til høyre først, altså lengst unna kaffemaskinen. På denne måten vil roboten begynne ytterst slik at ingenting kolliderer. Rekkefølgen til posisjoner på serveringsbrettet er illustrert i figur 3.64.



Figur 3.64: Oversikt over posisjoner for serveringsbrettet.

Navn	Nr	Beskrivelse
Cup-UnitS-Pos1	310	Plasserer kopp på serveringsbrett i kopposisjon 1
Cup-UnitS-Pos2	311	Plasserer kopp på serveringsbrett i kopposisjon 2
Cup-UnitS-Pos3	312	Plasserer kopp på serveringsbrett i kopposisjon 3
Cup-UnitS-Pos4	313	Plasserer kopp på serveringsbrett i kopposisjon 4
Cake-UnitS-Pos1	330	Plasserer kake på serveringsbrett i kakeposisjon 1
Cake-UnitS-Pos2	331	Plasserer kake på serveringsbrett i kakeposisjon 2
Cake-UnitS-Pos3	332	Plasserer kake på serveringsbrett i kakeposisjon 3
Cake-UnitS-Pos4	333	Plasserer kake på serveringsbrett i kakeposisjon 4

Tabell 3.19: Oversikt over programmer for plassering av kopper og kaker på serveringsbrett

3.6.4 Andre programmer

Det er mange programmer som blir benyttet i hovedprogrammet, utover de som direkte plukker og plasserer kopper og kaker. Disse er lagt fram i tabell 3.20.

Programmer

Navn	Nr	Beskrivelse
TURN(Left2Right)	400	Roboten snut fra kaffeside til kakeside
TURN(Right2Left)	401	Roboten snur fra kakeside til kaffeside
PLACE-Cup-Machine-FRONT	404	Plasserer kopp grepet fra front, i kaffemaskinen
PUSH-Button	405	Trykker knappen på kaffemaskinen
PLACE-Cup-Machine-SIDE	406	Plasserer kopp grepet fra siden, i kaffemaskinen
START-Pos-Left	407	Startposisjon kaffeside
START-Pos-Right	408	Startposisjon kakeside
MOVE-Full-Cup	409	Griper full kopp fra kaffemaskinen og flytter den over serveringsbrettet, klar til å plasseres
TURN(Mid2Left)	410	Snur fra serveringsbrettet til kaffeside
READY-Grab-SIDE	411	Klargjør posisjon for griping av kopp fra siden
READY-Grab-FRONT	412	Klargjør posisjon for griping av kopp fra front
READY-Place-Machine-SIDE	413	Klargjør posisjon for å plassere kopp i kaffemaskin (for kopp grepet fra siden)
READY-Place-Machine-FRONT	414	Klargjør posisjon for å plassere kopp i kaffemaskin (for kopp grepet fra front)
Fix-Kompai	415	Justerer posisjonen til brettet til Kompai.

Tabell 3.20: Oversikt over øvrige SLIM-programmer.

3.6.5 Bevegelsesparametre

Bevegelsesparametre

Bevegelsesparameter	Verdier	Beskrivelse
Speed	[1 mm/s, 5000 mm/s]	Farten til endeffektor
Accuracy	[1, 10]	Hvor mye snarvei som kan tas mellom posisjoner
Pause	[0, 1]	Om det skal gjennomføres stopp i posisjonen
Tool nr.	[1, 32]	Verktøynummer, konfigurert til griper
Acceleration	[0, 3]	Akselerasjon, ved 0 vil roboten akselerere ved maks kapasitet.
Smoothness	[0, 3]	Justerer mykheten til bevegelsene ved å endre hastigheten til aksenes akselerasjon

Tabell 3.21: Forklaring av tilgjengelige bevegelsesparametre

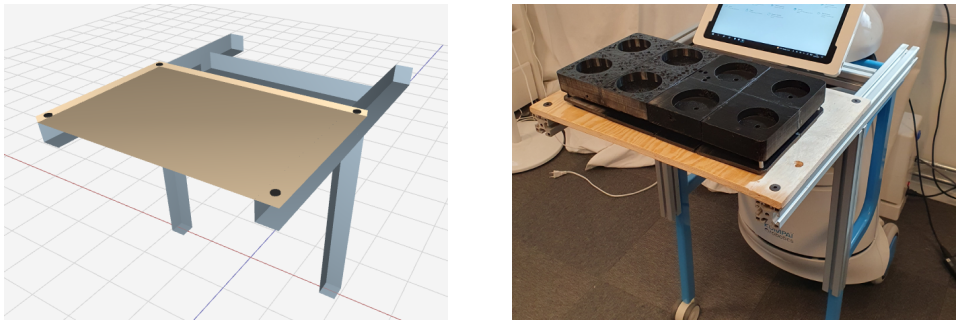
De ulike bevegelsesparametrene til NACHI er kort forklart i tabell **??**. Disse blir brukt til å tilpasse robotens bevegelser til oppgaven som blir utført. **Speed** opererer på to ulike verdier, 100 mm/s eller 250 mm/s. Maksfarten er begrenset til 250 mm/s for å innfri kravene for å kjøre roboten uten bur [57]. For best mulig syklustid er dette systemets standardfart. Det er imidlertid noen bevegelser som bør foregå varsomt. Farten vil senkes til 100 mm/s for bevegelser som interagerer med kopper, eller andre bevegelser med risiko (for eksempel ved trykking av knappen på kaffemaskinen, eller justering av Kompaï sitt brett). **Accuracy** er for det meste satt til A5. Ved for store snarveier vil det være risiko for uforutsigbarhet eller kollisjon, men for lite rom for snarveien gjør bevegelsene bråere. Istedenfor å endre accuracy kan man benytte pauser ved de stegene hvor roboten må treffe punktet for å gå videre. **Pause** blir brukt for eksempel når kaffekoppen går inn i kaffemaskinen. Da blir den satt i riktig posisjon i XY planet litt over den horisontale flaten som holder koppen. I dette punktet pauser den før den beveger seg nedover i Z-retning. Denne bevegelsen er så liten at det krever at det første punktet blir truffet slik at den vil ha en rett bevegelse i Z-retning. **Tool number** er nr. 23 for alle bevegelser. Tool 23 er konfigurert til fingergriperen. **Acceleration** benyttes ved veldig sårbare prosesser, som ved håndtering av full kaffekopp. **Smoothness** brukes på alle bevegelser som interagerer med noe annet, for å gjøre bevegelsene mykere og minske vibrasjoner.

3.7 Kompaï K3

Kompaï K3 er serviceroboten som skal servere kaffe og kake til beboerne. Serveringsbrettet er fastmontert på serviceroboten. Når kaffe og kake skal tilberedes, skal roboten stille seg på riktig plass slik at serveringsbrettet står tilgjengelig i riktig posisjon for at NACHI skal plassere kaffekopper og kake på brettet. Serviceroboten skal stå der helt til alt er ferdig tilberedt. Når tilberedningen er gjennomført skal den gå en runde for å servere kaffekoppene og kakene. Kompaï var i utgangspunktet ikke utformet med tilbehør/tilrettelegging for frakt av objekter. Det har derfor blitt utformet komponenter som innfrir dette behovet.

3.7.1 Oppsett av Kompaï

For at Kompaï skal være i stand til å frakte serveringsbrettet, er det utformet en fikstur som er festet til støtten på baksiden av roboten, modellert og fotografert i figur 3.65. Det er benyttet aluminiumsprofiler til å lage en struktur som stemmer overens med støttens mål. Den er designet for å sitte tett til oversiden av støtten, og er festet med borrelås for enkel montering. Profilenes materiale og oppbygging gir god strukturell integritet. På oversiden av strukturen er det festet en kryssfinerplate som kan holde serveringsbrettet. På denne måten vil ikke serveringsbrettet være i kontakt med metall. Det forhindrer blokkering av trådløse signaler og varmedannelse i batteriet. Det er benyttet borrelås for å sørge for enkel fastmontering av serveringsbrettet til platetoppen.

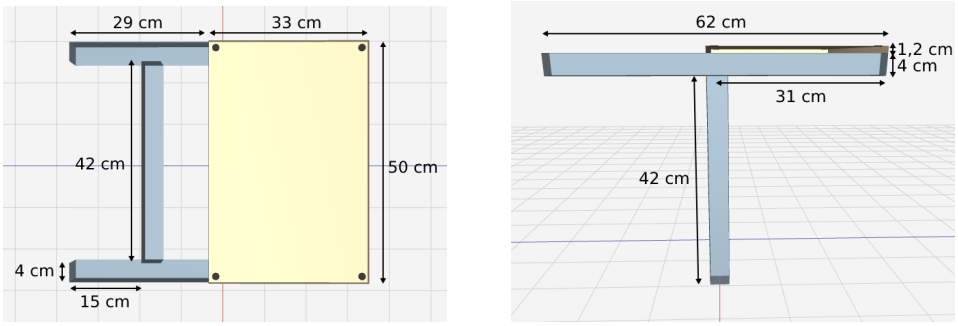


(a) **(b)**

Figur 3.65: 3D-modell og fotografi av fiksturen.

Det er benyttet T-spor aluminiumsprofiler med en høyde og bredde på 4x4 cm. Profilene er festet til hverandre med skruer, T-spor muttere og standard koblinger. Målene til fiksturen er illustrert i figur 3.66. Den består av to ben på 42 cm, med en horisontal profil festet symmetrisk på toppen av hver slik at de danner en T-form. De to T-formene er festet til hverandre med en profil som går på tvers mellom dem som sørger for at avstanden og vinklene blir rette. Den er festet på motsatt side av topplaten for å bidra til jevnere vektfordeling.

Både profilen som går på tvers og topplaten er festet med god avstand til skjermens bevegelsesområde.



Figur 3.66: 3D-modell av fiksturen med mål.

3.7.2 Kompaï K3

Kompaï K3 er en CE sertifisert servicerobot, produsert av Kompaï robotics i Frankrike. Serviceroboten er designete for å hjelpe folk i helsevesenet, for eksempel på sykehjem. Kompaï er designet med to støteben som er laget for at pasienter skal kunne gå med Komapï som med en vanalig rulator. Kompai har integrerte lidar-sensorer for hjelpe med navigering, og avstandssensorer som er i stand til å oppfatte hindringer opptil 10m unna Kompaï. Kompaï kjører av sikkerhetsmessige grunner med en maksfart på 1m/s og senker farten om en hindring er nærmere enn 1,25m og stoppe om den er nærmere en 0.5m. Kompaï har to blindesoner foran og to bak, som gjør at smale hindringer kan gå udetektert. Kompai tilfredstiller de uropeiske satnderne for 2006/95 /EC på elektronisk utstyr som kan brukes med lav spenning og 2006/42/EC på maskineri. [58]

3.7.3 Styring av Kompaï

For å styre Kompaï ble det først forsøkt å koble en rapberrypi opp til Kompaï for å så kommunisere med Komapï via raspberrypien, da Kompai er en egen rosmaster og så man kan ikke kommunisere fra en rosmaster på PCen og direkte med Kompaï. Dette bød på flere utfordringer med å få kontakt med Kompaï sine egne rosnoder samt å overføre VScode filer. Derfor valgte man istedenfor å programmere Kompaï ved at en virtuell maskin (VMbox) ble koblet direkte opp til Kompaï slik at man fikk tilgang til alle rosnoden og slapp å kommunisere igjennom en tredjepartner.

Kompaï blir styrt av en node i ROS i Ubuntu 20.04, der det er laget en egen rosservice for hver posisjon som Kompaï skal gå til. Servicene blir kalt via flexgui, som gir beskjed om når Kompaï skal gå til neste posisjon. Alle posisjonene er lagret med Kompaï sin posisjons-lagringsfunksjon, ved å flytte

Kompaï til den ønskede posisjonen og så trykke «Tool box, +, skrive inn POI navn, save» på skjermen til Kompaï. Kompaï sine egne sensorer sørger for at Kompaï ikke kjører inni noe, selv om rommet skulle blitt ommøblert. For at serveringsbrettet på Kompaï ikke skal kollidere med ting har lengden Kompaï er programmert til å vite at den har, blitt endret fra 0.75 meter til 0.95 meter.

Kompaï kode

For å kunne styre Kompaï ble det benyttet en pc koblet direkte opp til Kompaï, med full tilgang til alle Kompaï sine noder i ROS. Etter å ha fått tilgang til Kompaï sine noder ble det laget et catkin workspace i ROS, med en package i Python, der man lagde en node i Visual studio code (VScode) for å kunne styre Kompaï. Node som styrer Kompaï er kalt movement_node.py og ligger i packagen kompai_controller. Kompaï blir styrt ved at servicen blir kalt fra flexgui, for at dette skal være mulig å kalle servicen så må movvement_node.py bli startet (rosrun) først. Koden i movement_node.py funger som følgende:
Importer alle moduler som trengs for å styre Kompaï, inkludert tre unike moduler for Kompaï, Navigate og NavigateRequest for å kunne navigerer Kompaï og Docking for å kunne sjekke om Kompai er i laderen eller ikke.
For hver posisjon som Kompaï skal gå til er det laget en funksjon som blir kalt som er rosservice. Det er i tillegg laget en funksjon som sjekker om Kompaï står i laderen eller ikke 3.68, siden Kompaï må gå ut av laderen før den kan få beskjed om å gå til en posisjon. Denne funksjonen blir bare brukt i funksjonen on_move_request 3.67 som får Kompaï til å stille seg opp foran Nachi, da Kompaï for alle andre posisjoner må være ute av laderen sin allerede.

For koden til hele node se veddlegg A.1

```
22  #If Kompai is docking it will disconnect from the charger
23  def on_move_request(obj):
24      global dock_status
25      if dock_status == 3:
26          disconnect = rospy.ServiceProxy('/dock/disconnect', Empty)
27          disconnect.call()
28
29      #wait 5 seconds
30      time.sleep(5)
31
32
33      # using the builtin service in Kompai /naviagat/goto to make Kompai go to the wanted possition
34      navigate = rospy.ServiceProxy("/navigation/goto", Navigate)
35      response = navigate.call(NavigateRequest (name = 'nachi2'))
36
37      print(response)
38      return EmptyResponse()
39
```

Figur 3.67: Funksjonene on_move_request

```
11
12  #when dock_statuse = 3 then Kompai is in the charger
13  dock_status = 3
14
15
16  #this definition i checinkg Komapis dock statuse
17  def on_dock_statuc_changed(obj):
18      global dock_status
19      dock_status = obj.status
```

Figur 3.68: Funksjonen dock_statuse

Kapittel 4

Integrasjon

For å integrere komponentene til et helhetlig system er det benyttet Robot Operating System (ROS) og FlexGUI. PPM Robotics har utviklet FlexGUI, som baserer seg på ROS for å gjøre robotintergrasjon mer sømløst. Det kan benyttes for å lage brukergrensesnitt, og styre ROS. Bálint Tahi hos PPM har konfigurert en virtuell maskin med Ubuntu med ROS Noetic og ferdig integrasjon med FlexGUI. Denne virtuelle maskinen er grunnlaget for all ROS-kommunikasjonen i systemet. ROS-kommunikasjonen baserer seg på noder med topics og services, hvor topics

4.1 Oppsett av virtuell maskin og ROS

Det er fire maskiner som må kommunisere med hverandre gjennom ROS på den virtuelle maskinen for at systemet skal fungere. De er satt opp på følgende måte:

Kompaï K3

Kompaï fungerer som ROS-master, og har sitt eget nettverk hvor all ROS-kommunikasjon foregår. Alle komponentene som skal interagere med systemet er konfigurert til dette nettverket med statisk IP. Ruteren er integrert i robotens kabinett, og på bakgrunn av at robotens funksjoner avhenger av fri bevegelse i rommet, medfører dette at trådløs tilkobling er nødvendig. Kompaï er kun utstyrt med et nettbrett som brukergrensesnitt, og egner seg dårlig til å programmeres på. Den virtuelle maskinen på stasjonær PC benyttes for styring av ROS-masteren.

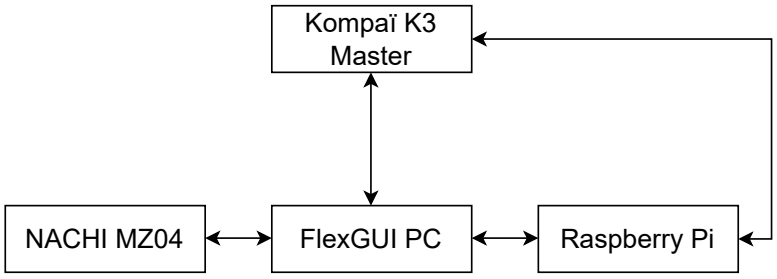
Virtuell maskin

Den virtuelle maskinen er en ROS-slave. Den er konfigurert med dobbel nettverkskonfigurasjon, slik at den er på samme nett som ROS-master og NACHI. I den virtuelle maskinens nettverksinnstillinger er det satt opp en nettverksbro, slik at maskinen er koblet opp til begge nettverkene. NACHI er koblet med ethernet til labnettet, og avhenger av den virtuelle maskinen for kom-

munikasjon med ROS-master. Brukergrensesnittet i FlexGUI er lokalisert på den virtuelle maskinen, og herfra er alle nodene tilgjengelige.

NACHI MZ04
NACHI fungerer som ROS-slave. FlexGUI er utviklet i samarbeid med NACHI, og integrasjonen med NACHI MZ04 er derfor tilrettelagt. For å koble MZ04 til en maskin må roboten og maskinen være koblet til samme nettverk. Deretter benyttes ftp til å gå inn på robotens filsystem og endre konfigurasjonsfilen. Linjen som definerer IP-adressen må stemme overens med maskinens IP adresse.

Raspberry Pi, slave
Raspberry Pi-en er koblet opp til samme nettverket som ROS-masteren, og fungerer som slave. For å gjøre dette ble nettverksinnstillingene endret med `sudoedit /etc/netplan/50-cloud-init.yaml` samt at to linjer ble lagt til bashrc-filen: export ROS_MASTER_URI = http://192.168.X.X:X og export ROS_IP=192.168.X.X



Figur 4.1: Diagram for kommunikasjon mellom enhetene

4.1.1 Noder

NACHI MZ04: MZ04

MZ04 er en node med god integrasjon i FlexGUI. Det medfører at de variabeltypene som er lagt fram i tabell 3.11 (seksjon 3.5.4) er lett tilgjengelig som topics fra grensesnittet. Det er kun heltallsvariabler som kan skrives av FlexGUI til NACHI. Heltallsvariablene brukes av SLIM-programkoden til å kommunisere med brukergrensesnittet i FlexGUI, som forklart i tabell 3.14 (seksjon 3.6.1).

Raspberry Pi: raspi_gateway

raspi_gateway er navnet på noden som kjøres fra RPi. Programmet den kjører er todelt, der den ene delen henter verdier fra mikrokontrolleren med bruk av BLE og den andre publiserer verdiene videre til ROS og FlexGUI. Den består

av tre publishers som sender til de ulike topicene nextMuffin, nextCoffee og servingData. Topicene brukes videre av robotene. Det står mer opp dette i **??**

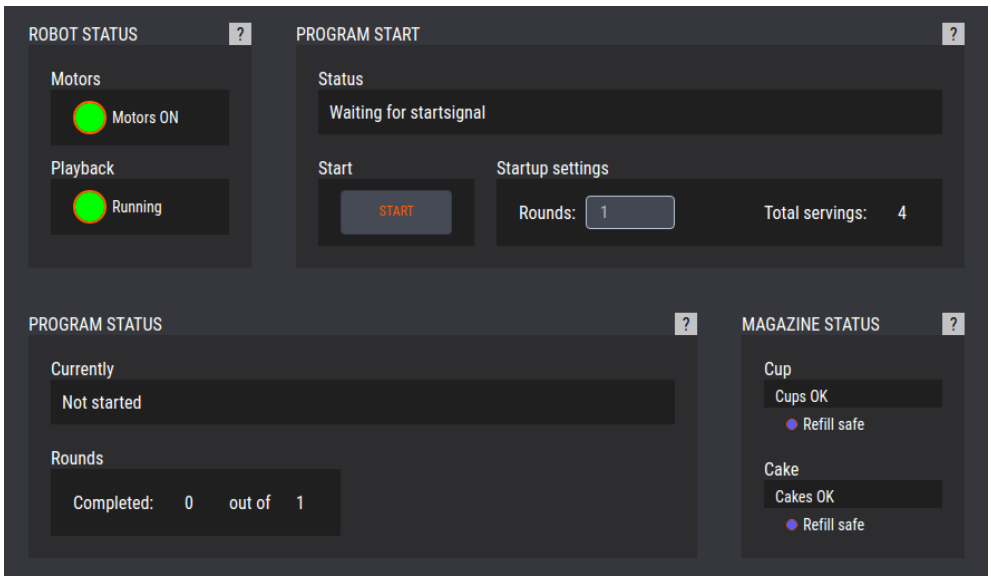
Kompaï K3: kompai_controller

Noden kompai_controller inneholder flere servicer. Disse servicene forteller Kompaï at den skal gå til ulike posisjoner på NHL

4.2 FlexGUI brukergrensesnitt

Brukergrensesnittet i FlexGUI er det sentrale grensesnittet for styring av systemet. Her knyttes systemets komponenter sammen til et helhetlig system.

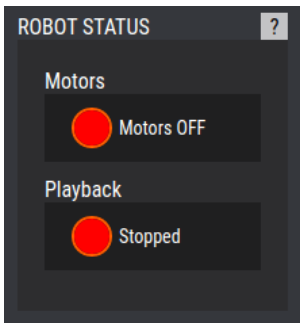
4.2.1 Grafisk grensesnitt



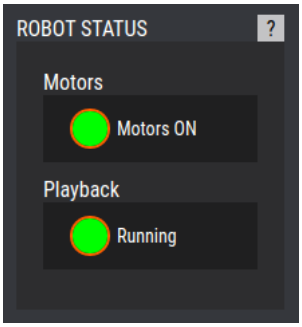
Figur 4.2: Skjermdump av brukergrensesnittet i FlexGUI.

Robot Status

Under robotstatus vises status for motor og playback. Disse baserer seg på de standard utgangssignalene Motors energized og Running in auto. De er enten på eller av. Når de er av vil de se ut som i figur 4.3a. Når de er på vil de se ut som i figur 4.3b.



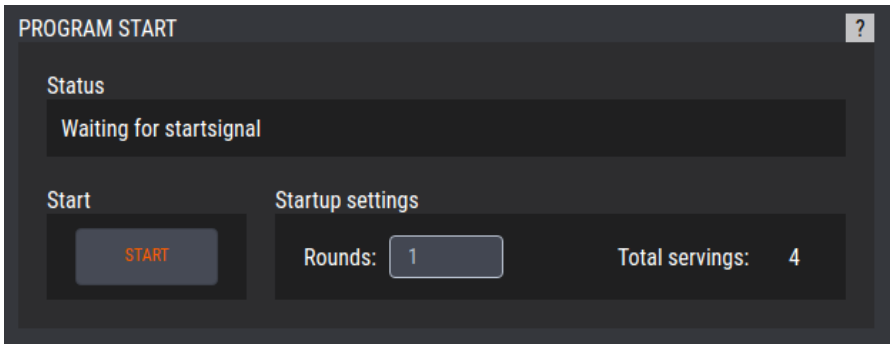
(a) Motor av og playback stoppet.



(b) Motor på og playback startet.

Program Start

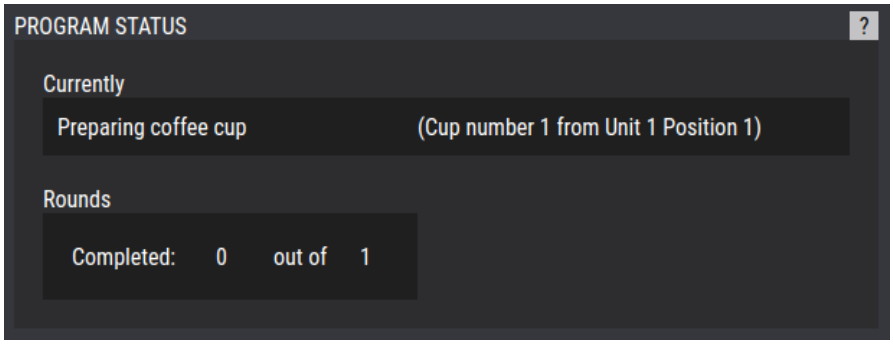
Under programstartruten er status og styring for programstart. Statusruten vil vise om programmet er startet, eller venter på startsignal. Om det kreves nullstilling av systemet, eller om systemet er i teach mode, vil dette vises. Startknappen vil kun være tilgjengelig når NACHI er startet i auto modus, og det ikke er noen feil i systemet. Startinnstillingene kan kun fylles ut før startsignalet er mottatt. I disse innstillingene kan antall runder settes opp. Ved flere runder enn to, vil magasinene måtte fylles opp underveis. Figur 4.4 viser ruten slik den ser ut når start er tilgjengelig.



Figur 4.4: Rute for programstart.

Program Status

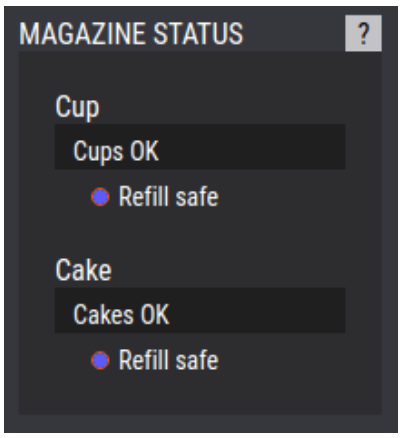
I ruten for programstatus (figur 4.5) står det informasjon om prosessen. Under Currently vises programmets nåværende status. Hvis programmet stoppes vil teksten forbli den samme, og vise hvor programmet vil gjenopptas. Ved nullstilling av programmet vil teksten informere om at en nullstilling er foretatt. Når roboten er i ferd med å plukke/plassere kopp eller kake, vil det vises hvilken posisjon det plukkes fra/plasseres til. Under runder vil antall fullførte runder stå, samt hvor mange runder som skal kjøres.



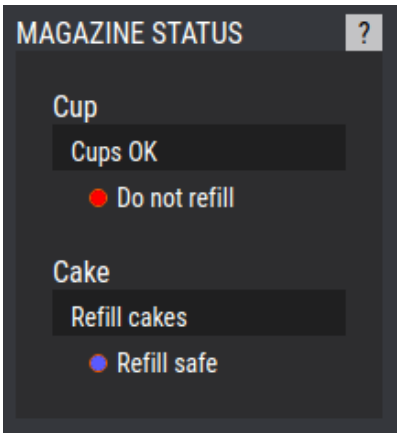
Figur 4.5: Rute for Program Status

Magazine Status

I ruten for magasinstatus står det informasjon om statusen til koppmagasinet og kakemagasinet. Det vil vise om det er tomt for kopper eller kaker. Videre vil det stå om det er trygt å fylle på magasinene. Det avhenger av hvilken del av programmet NACHI er i. Hvis den er i ferd med å plukke kopp, vil det stå at det ikke skal utføres påfyll av kopper. Det samme gjelder for kaker når roboten er i ferd med å plukke opp kaker.



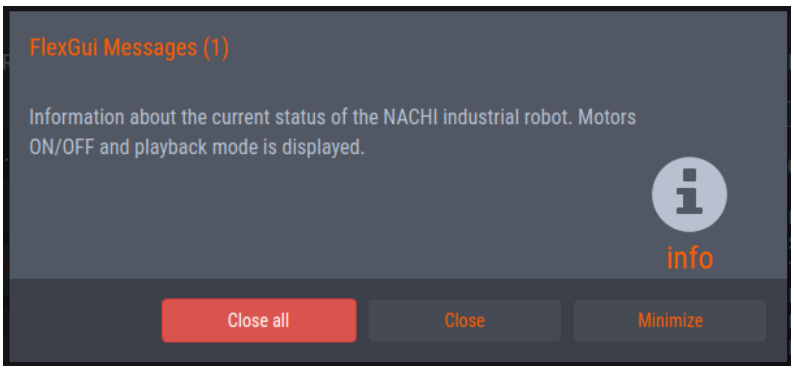
(a) Magasiner ok, påfyll mulig.



(b) Kakemagasin tomt, påfyll av koppmagasin ikke trygt.

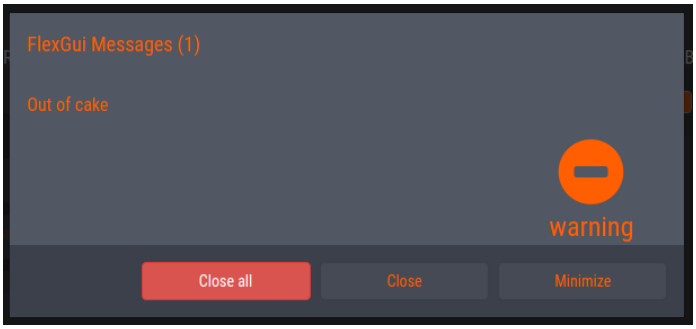
Informasjon og feilsøking

Alle rutene er utstyrt med en informasjonsknapp (klikkbar rute med spørsmålstegn). Om denne trykkes vil informasjon om ruten som ble trykket, dukke opp i en popup. Popup med informasjon når informasjonsknappen på Robot Status trykkes, er vist i figur 4.7.

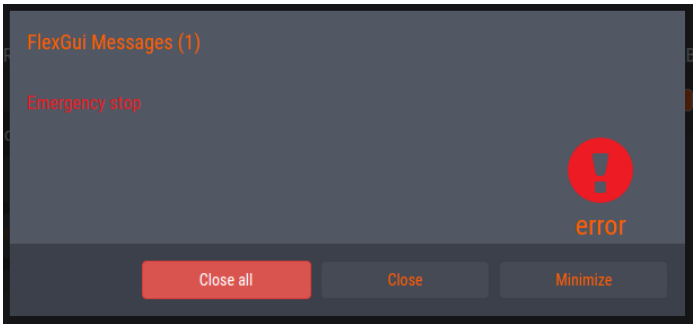


Figur 4.7: Informasjonsknapp popup.

Det er også implementert popupmeldinger som varsler om tomme magasiner eller nødstopp. Dersom kopp- eller kakemagasinet er tomt, vil en advarsel som i figur 4.8 dukke opp. Dersom nødstopp er aktiv, vil en feilmelding dukke opp, som i figur 4.9.



Figur 4.8: Advarsel om tomt kakemagasin.



Figur 4.9: Feilmelding om nødstopp.

4.2.2 Script

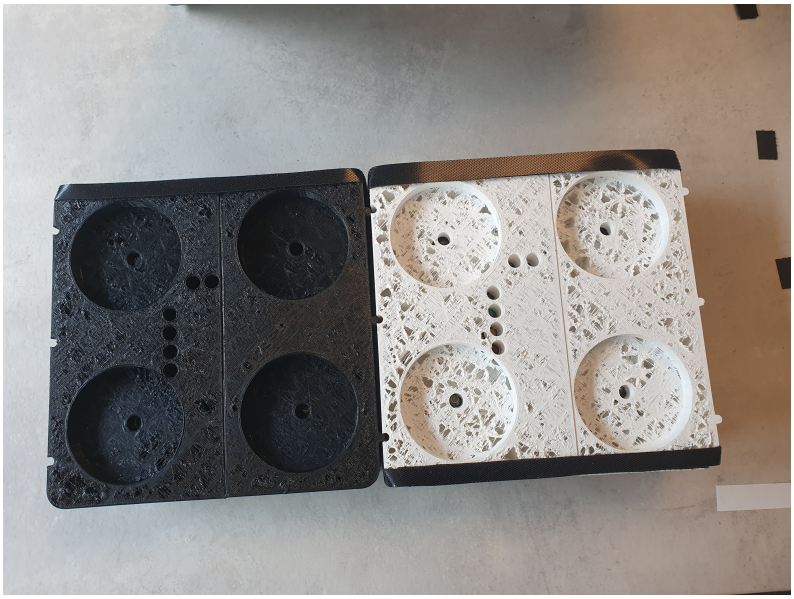
FlexGUI benytter JavaScript for scripting. Det er en init fil som kjører når brukergrensesnittet åpnes, som inneholder alle nødvendige funksjoner. OnChange scriptene er koblet til topics og brukes blant annet til å oppdatere statusstrenger, eller oversette magasindata fra sensorsystemet til neste posisjon som skal gripes.

Kapittel 5

Resultat

5.1 Sensorsystem

5.1.1 3D-modeller

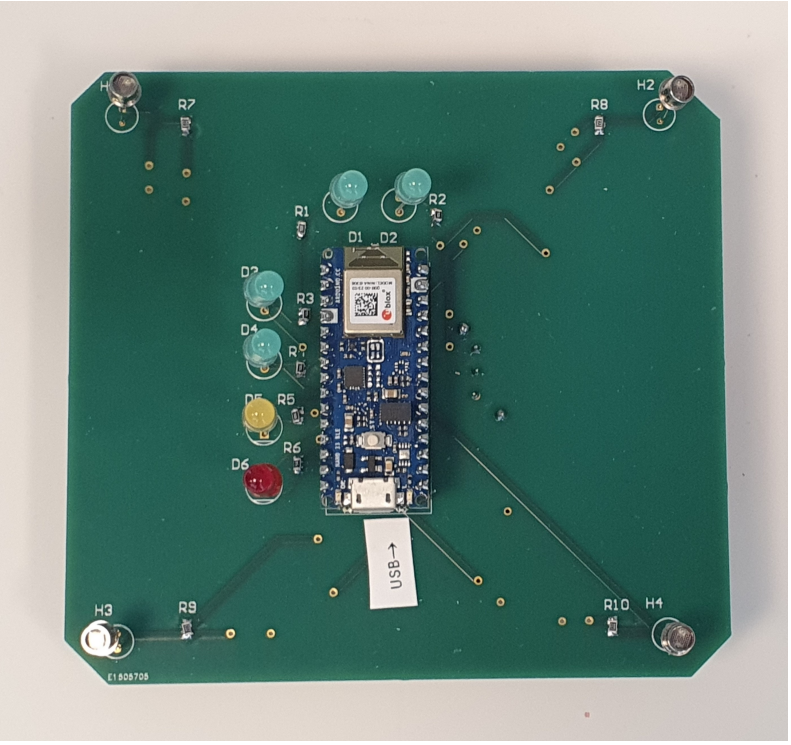


Figur 5.1: 3D-modellene av kakeholderne

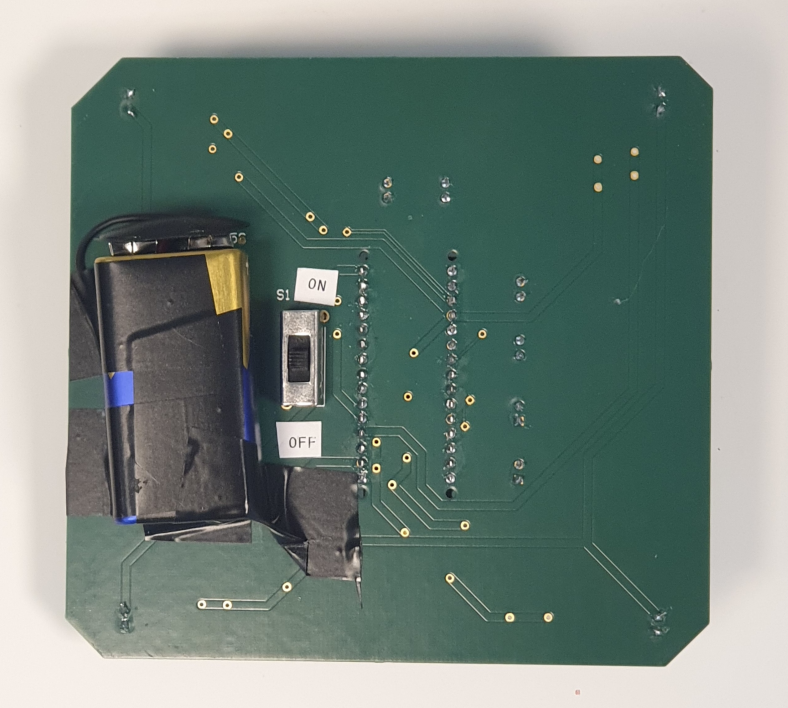
Figur 5.2: 3D-modellene av koppholderne

Figur 5.3: 3D-modellen av serveringsbrettet

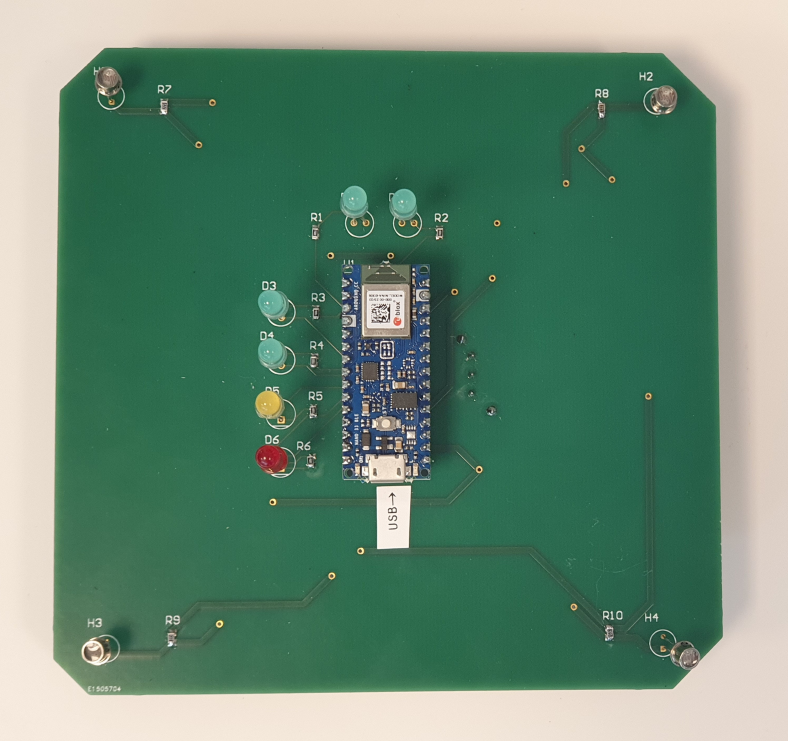
5.1.2 PCB-er



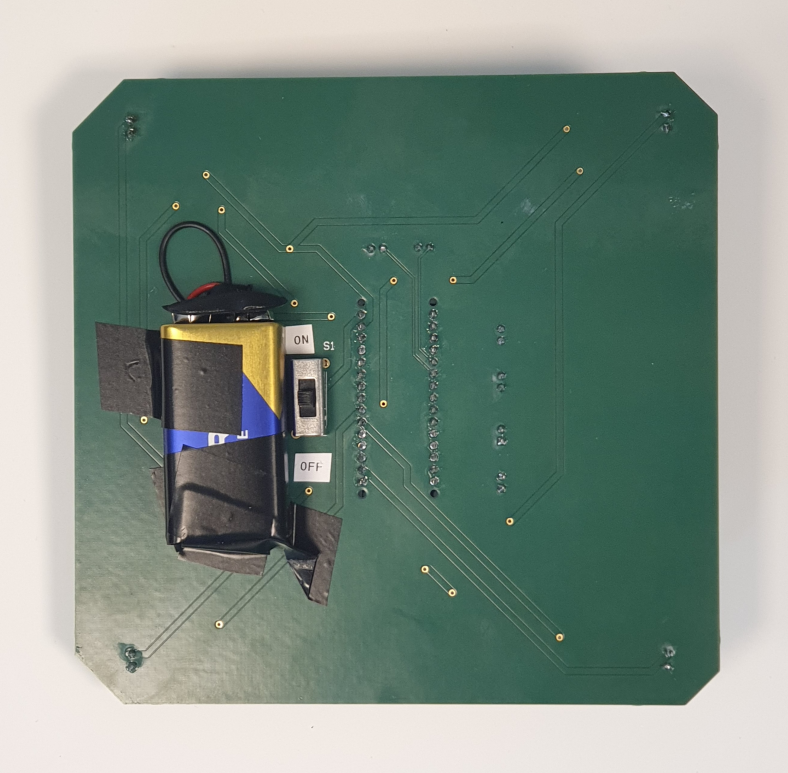
Figur 5.4: Forsiden av kakeholder PCB-en



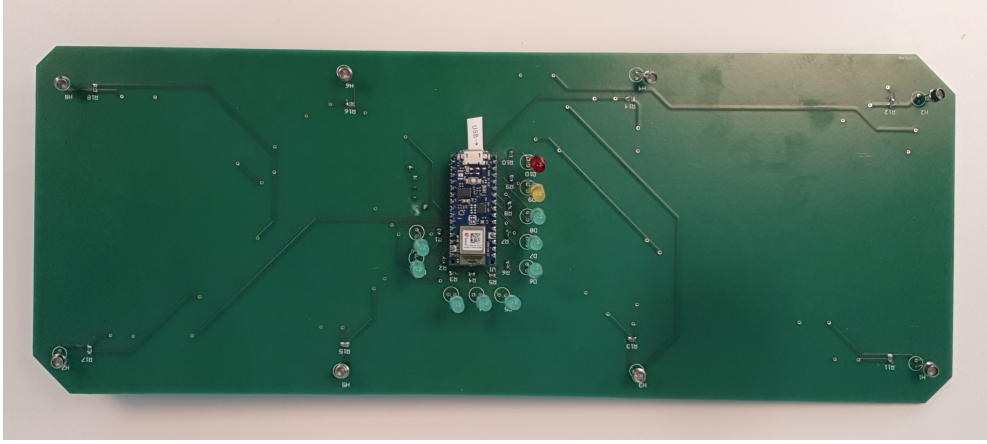
Figur 5.5: Baksiden av kakeholder PCB-en



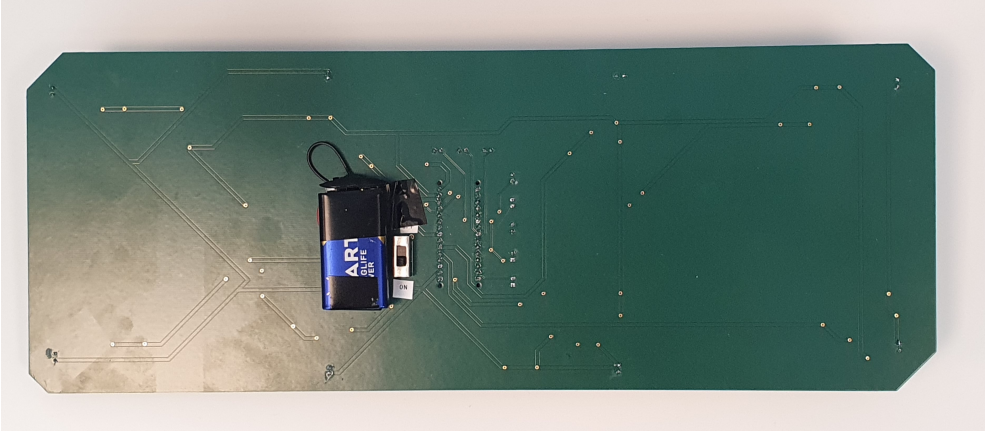
Figur 5.6: Forsiden av koppholder PCB-en



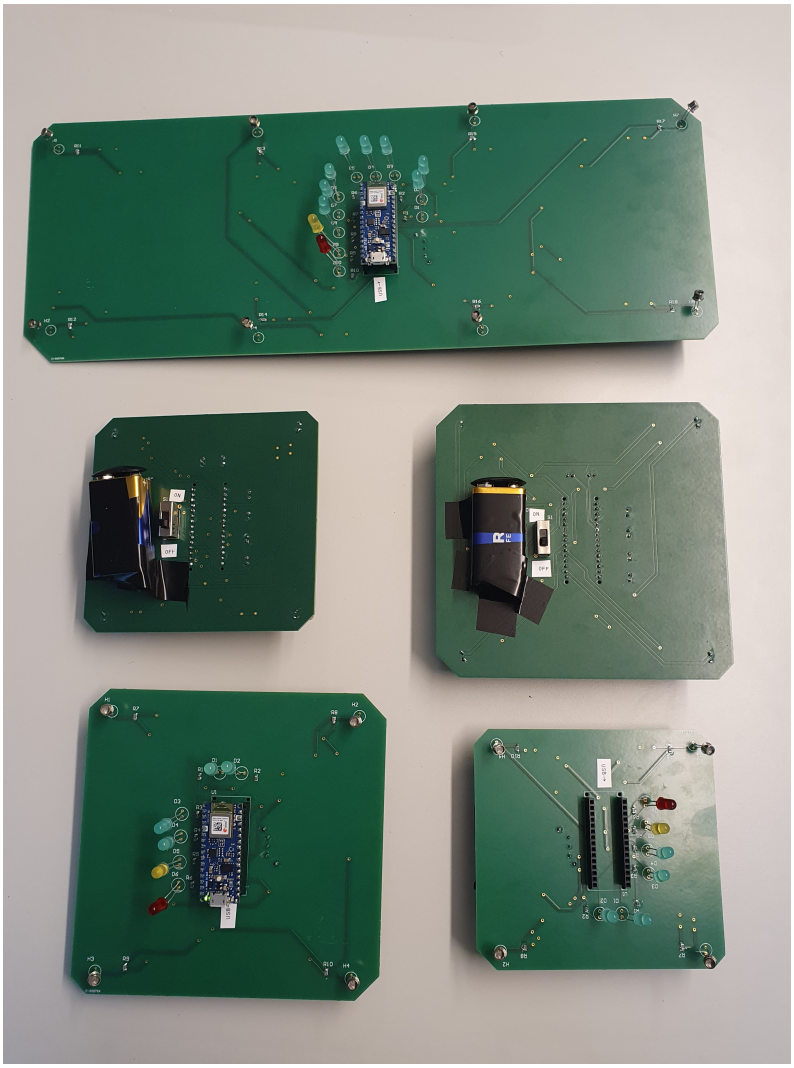
Figur 5.7: Baksiden av koppholder PCB-en



Figur 5.8: Forsiden av serveringsbrett PCB-en

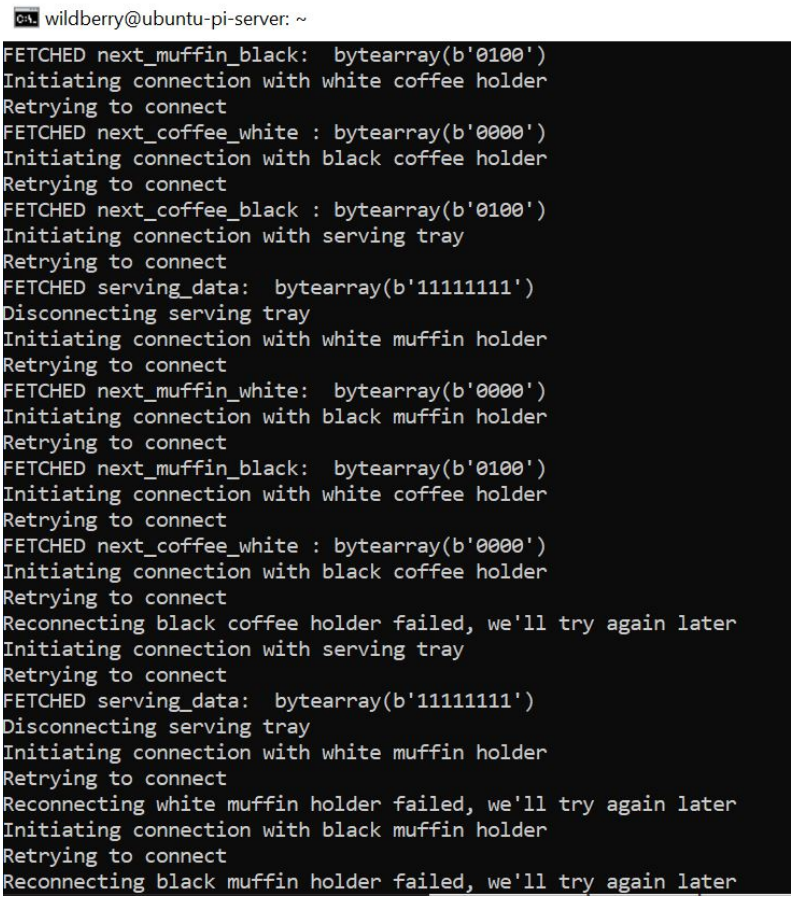


Figur 5.9: Baksiden av serveringsbrett PCB-en

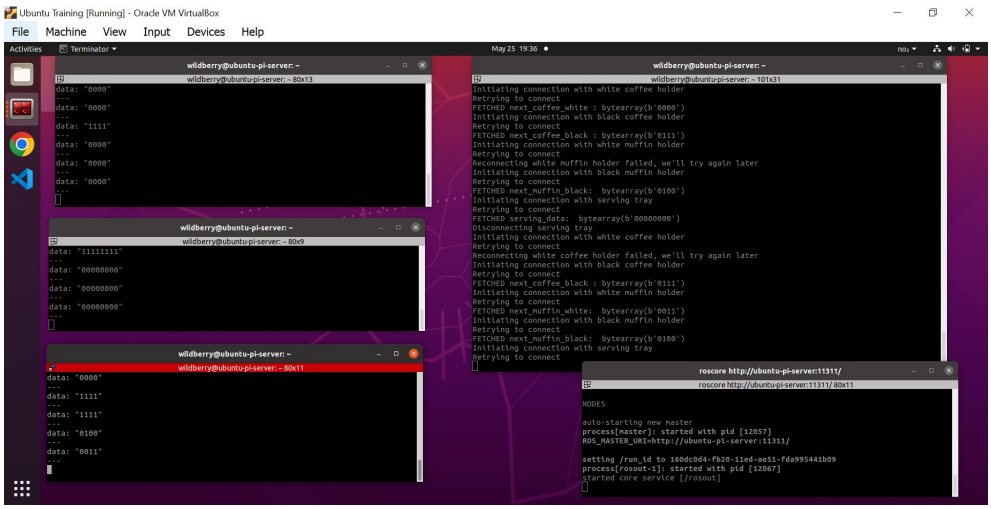


Figur 5.10: Ferdige PCB-er

5.1.3 Programvare



Figur 5.11: RPi-en kobler seg til enhetene på rundgang



Figur 5.12: RPi-en tilkoblet ROS

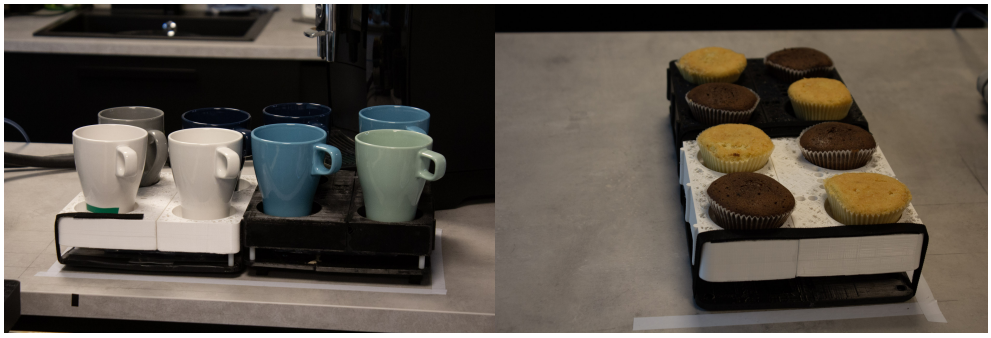
5.2 NACHI MZ04: Tilberedning av kaffe og kake

Systemet er implementert i NHL, med alle tilhørende komponenter. Eksterne enheter som pneumatikk og koblinger er integrert i kjøkkenbenken så godt det lar seg gjøres, slik at systemet er å anse som en del av kjøkkenet.

Figur 5.13: Foto av helhetlig system

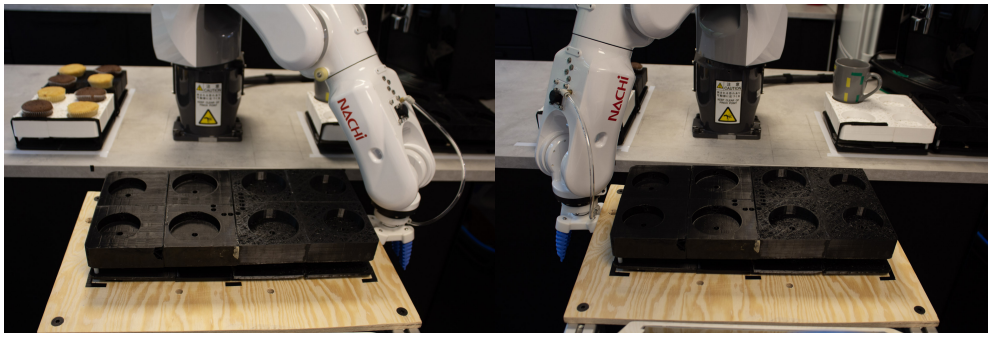
Kakemagasinene tar kaker med en maks diameter på 6 cm, og en høyde på maks 4 cm. Kaffemagasinet tar kopper med en maks diameter på 7 cm, og bør ikke være høyere enn 10 cm for best pålitelighet. Hanken på koppen kan strekke seg maksimalt 4 cm fra koppens ytterste radius. Systemet er utformet

til å fungere optimalt med kopper med nøyaktig 10 cm høyde, og en øvre diameter på 8 cm. Bunnen bør ikke være smalere enn 5 cm i diameter.



(a) Foto av fullt koppmagasin **(b)** Foto av fullt kakemagasin

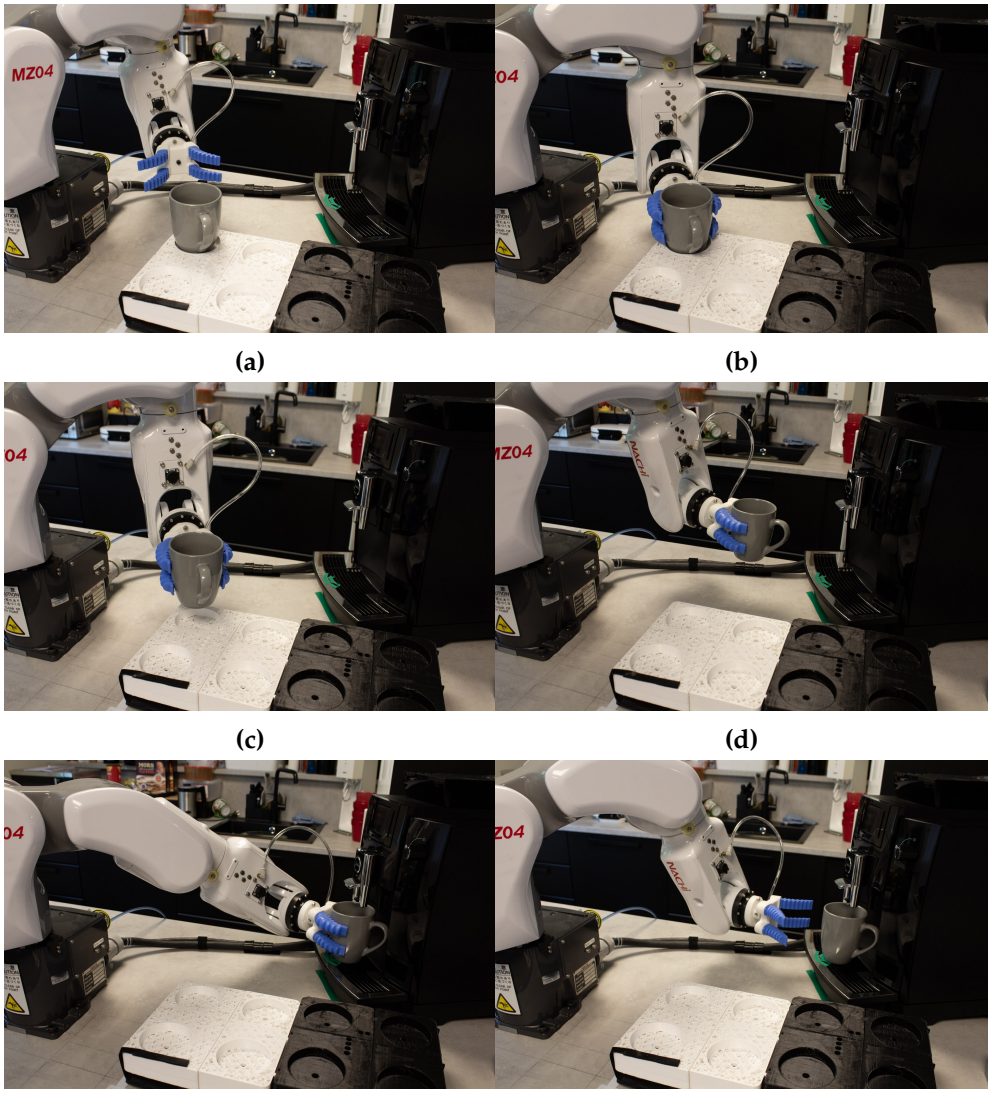
På bakgrunn av utfordringer med Kompaïs posisjonering er det lagt til et program som justerer brettet. Kompaï snur seg under navigering, og baksiden vris så den står skjevt. NACHI kompenserer for dette ved å justere brettet fra hver side som vist i figur 5.15. Denne prosessen tar ca. 1 minutt, men gjennomføres kun én gang på starten av hver runde.



(a) **(b)**

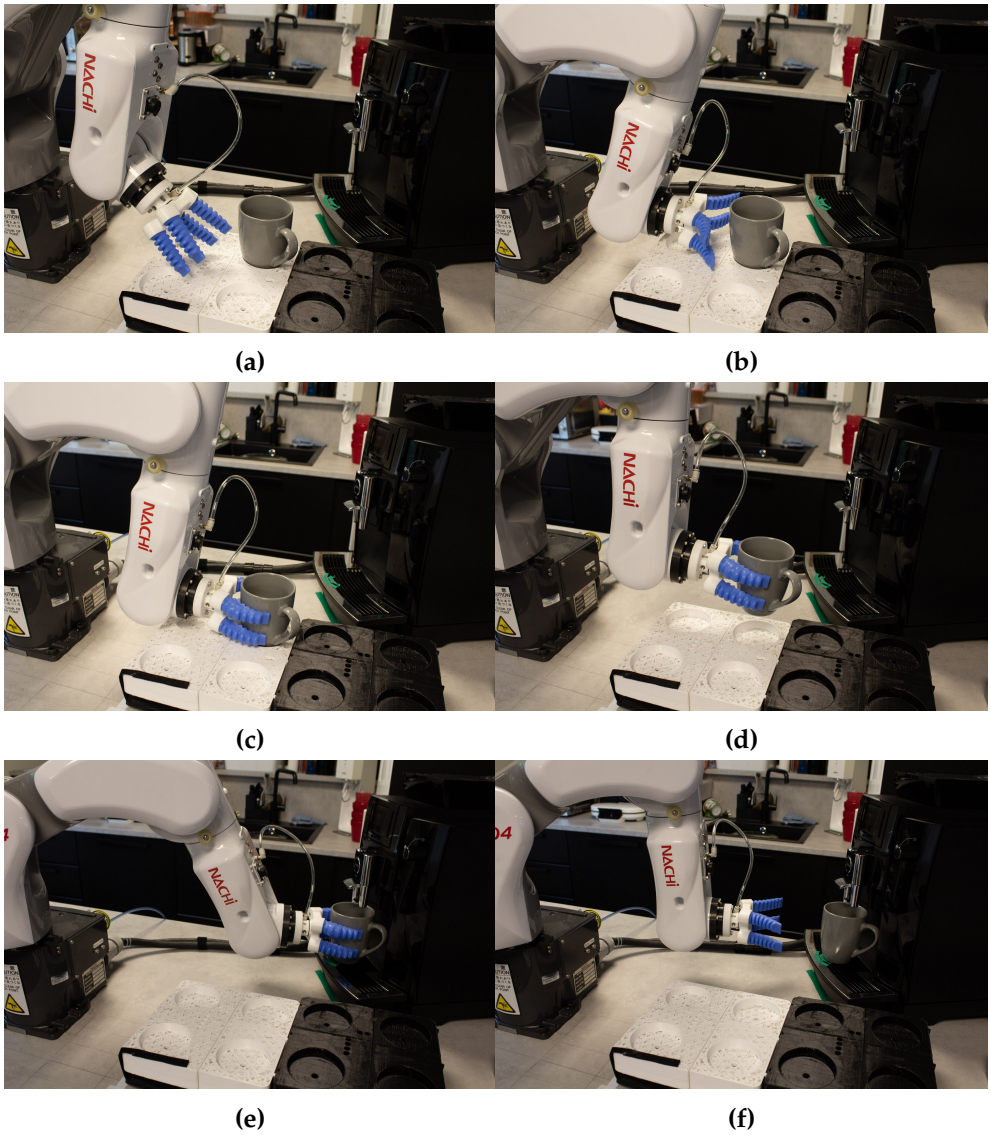
Figur 5.15: Justering av brettet til Kompaï.

I figur 5.16 plukker NACHI opp koppen fra første posisjon. I denne posisjonen må griperen komme fra siden. Den blir så plassert i kaffemaskinen. Fordi griperen plukker koppen fra siden, må den også plassere den i kaffemaskinen fra siden, slik at den vil stå med samme orientering som en kopp plassert fra fronten. For en kopp som blir grepet fra siden vil det ta 15 sekunder å gripe koppen, og 35 sekunder å plassere den i kaffemaskinen. Dette er maksimal tid for å gripe kopp og plassere i kaffemaskin, da det opereres i de ytre parametrene av robotens akser.



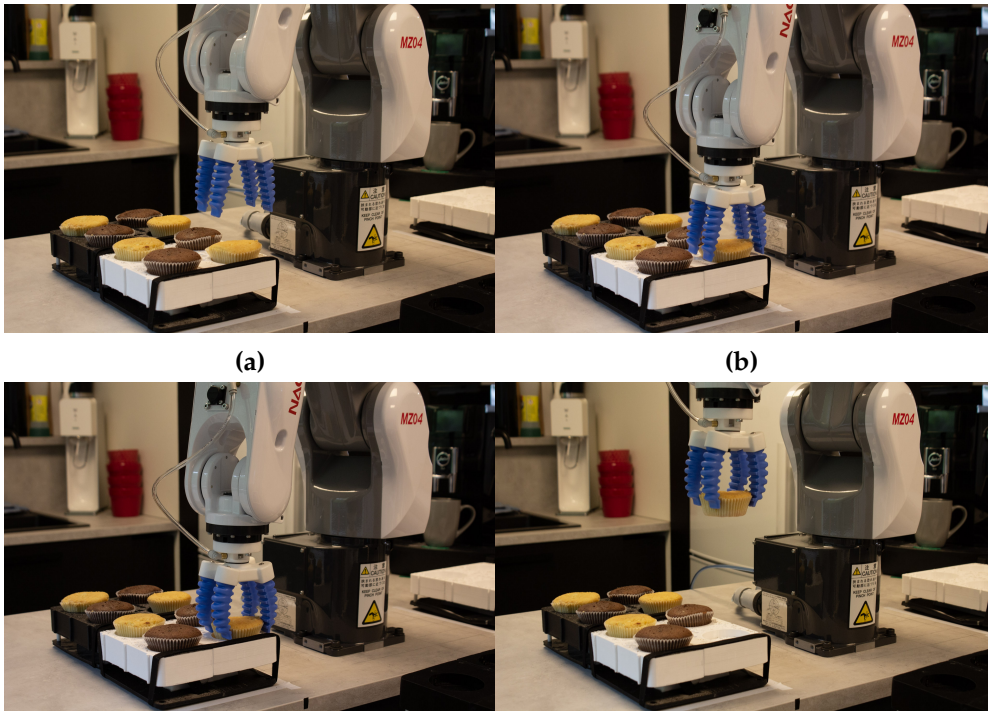
Figur 5.16: Plukking og plassering av kopp (fra siden).

I figur 5.17 blir koppen i tredje posisjon plukket. Denne blir plukket fra fronten, og plassert i kaffemaskinen. Legg merke til at koppenes posisjon i kaffemaskinen er tilsvarende i figur **??** og figur **??**.



Figur 5.17: Plukking og plassering av kopp (fra front).

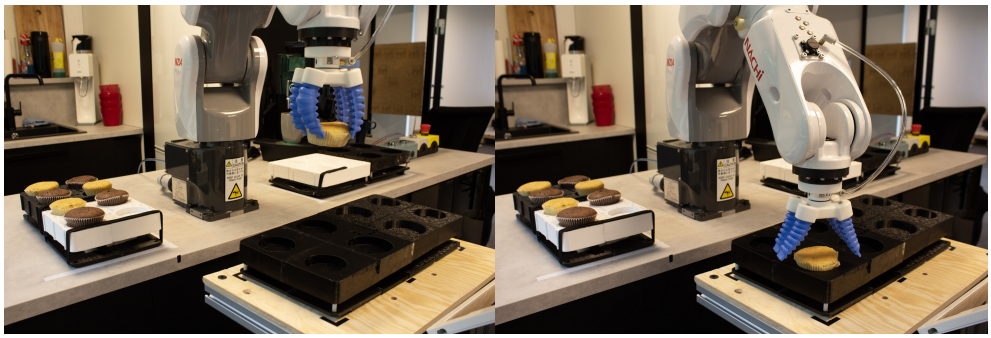
Etter å ha plassert koppen i kaffemaskinen blir maskinen skrudd på ved at griperen plasserer seg foran panelet på maskinen, lukker griperen, og drar en finger over knappen. Det tar 20 sekunder å utføre. Deretter snur roboten seg til andre siden og plukker kaken som i figur 5.18. Kaken blir plassert rett på serveringsbrettet som i figur 5.19. Prosessen tar 25-35 sekunder å utføre, avhengig av hvilken kakeposisjon det plukkes fra.



(a) (b)

(c) (d)

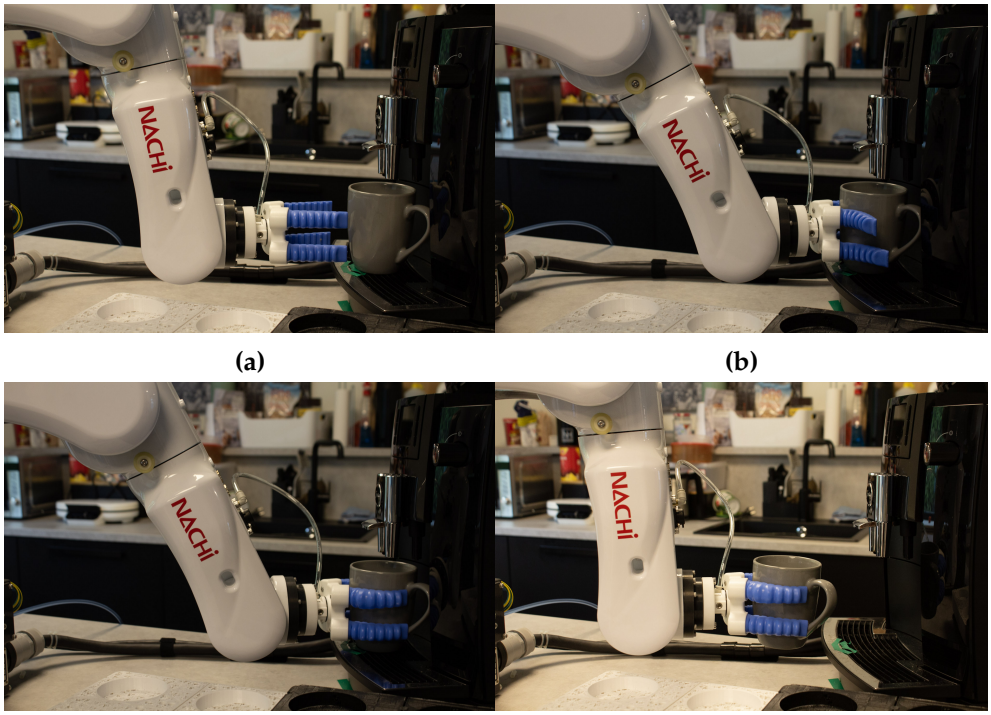
Figur 5.18: Plukking av kake fra kakeholder.



(a) (b)

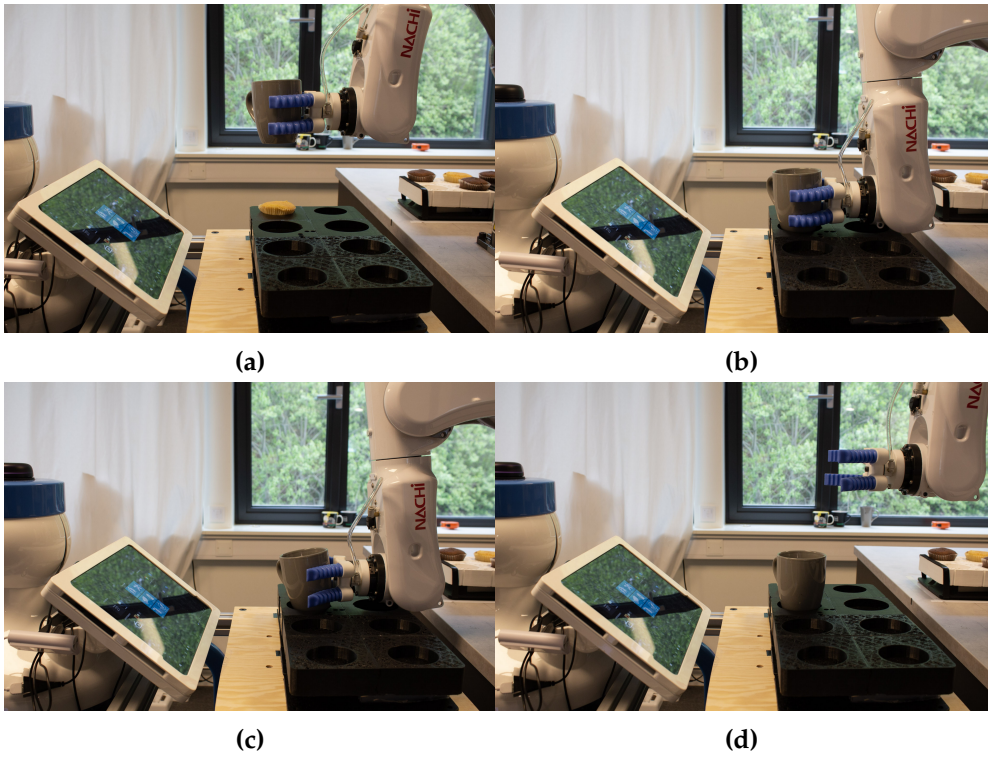
Figur 5.19: Plassering av kake på serveringsbrett.

I figur 5.20 blir koppen hentet fra kaffemaskinen. Det vil da ha gått ca. et minutt siden knappen ble trykket. Kaffemaskinen tar 45 sekunder fra knappen trykkes til koppen med kaffe er klar. Det vil ta 35 sekunder å frakte koppen fra kaffemaskinen til serveringsbrettet. Fordi koppene blir plassert i kaffemaskinen med samme posisjon og orientering, er denne prosessen lik uansett hvilken magasinposisjon koppen har utgangspunkt i.



Figur 5.20: Henting av full kopp fra kaffemaskin.

I figur 5.21 blir koppen plassert på serveringsbrettet. Når griperen går fra lukket til åpen vil det dytte koppen litt bakover. Ved for lett kopp kan dette medføre at koppen velter. Men er koppen av standard vekt vil den stå stabilt når griperen trekker seg sakte tilbake. Roboten vil deretter trekke seg oppover, og snu seg tilbake til startposisjon. Prosessen som er beskrevet her, vil gjenta seg til sammen fire ganger, før serveringsbrettet er klart.

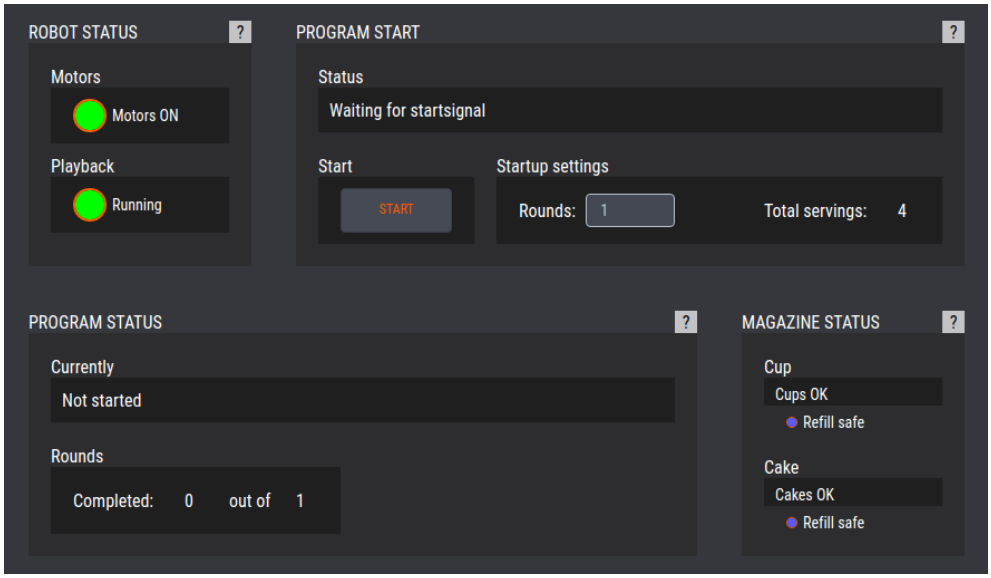


Figur 5.21: Plassering av kopp på serveringsbrett.

5.3 Brukergrensesnitt

Brukergrensesnittet består av kontrollpanelet på kjøkkenbenken ved siden av NACHI 5.22, og en FlexGUI-skjerm 5.23. Kontrollpanelet gjør at den viktigste robotstyringen kan gjøres umiddelbart med direkte tilkobling til kontrollenheten. Brukergrensesnittet på FlexGUI tilbyr overvåkning over systemet, og en kontrollert start som forholder seg til viktige variabler i systemet.

Figur 5.22: Kontrollpanel med 4 brytere (fra venstre: Start, Stopp, Reset, Nødstopp)



Figur 5.23: Skjermdump av brukergrensesnittet i FlexGUI.

5.4 Brukermanual

Det er utviklet en brukermanual

5.5 Kompaï

Kompaï funger som ønsket, ved at servicroboten nå kan styres fra FlexGUI, til å gå til ønsket posisjon på kommando.

Kapittel 6

Videre arbeid

6.1 Stabilitet

6.1.1 Bluetooth og programvare

Å bruke to RPi-er istedet for en ville, forbedret BLE-oppsettet betraktelig. Da er det ikke lenger en RPi som må koble til og fra bluetooth, men to RPi-er som kan være tilkoblet og lese fra henholdsvis tre og to mikrokontrollere på de ulike enhetene. Verdiene ville kunne sendes uten merkbare forsinkelser. For å gjøre dette vil man eventuelt trenge en BLE-adapter samt å klone et microSD-kort. Fem enheter overstiger antallet antennen på RPi og en helt decent kode klarer å holde på til enhver tid. På grunn av det å få signalene igjennom ble prioritert over å spare strøm, vil man igjen kunne satse på strømsparingstiltak dersom man vet at BLE-tilkoblingen er god. Slik kan man finne en balanse mellom pålitelighet og det å spare strøm. En svakhet i systemet er at det ikke lenger vil funke, dersom noen andre enheter enn RPi-en kobler seg på enheten. På den andre siden er det greit, ettersom det bare er tenkt at en enhet skal koble seg på uansett. For å gjøre den trådløse dataoverføringen mer sikker kan man kryptere dataen og begrense hva slags enheter som kan koble seg på f. eks. ved at den andre enheten må oppgi en kode. Andre forbedringer man kunne gjort i koden er å gjøre det om til et bibliotek slik at det er ryddigere å lese. For å samhandle best mulig med NACHI må det testes mer. Basert på tilbakemeldinger derfra, ville RPi-koden optimaliserers for dette formålet.

6.1.2 Internettilkobling

Internettilkoblingen er ustabil når det kobles to nettverk til samme maskin. Dette er uheldig når alle robotene og hele systemet kommuniserer sammen via FlexGUI. For å unngå dette skulle man ideelt sett hatt alle enhetene tilkoblet samme nettverk.

6.1.3 Posisjonering av serveringsbrett

Med løsningen som er nå er det NACHI som retter opp serveringsbrettet på Kompaï. Dette kunne blitt fikset innad i Kompaï sin programvare.

6.1.4 Testing

Ytterligere testing av systemet bør foretas, da det var lite tid til å gjennomføre dette grundig.

6.2 Finpussing

PCB: Beskyttelseskretser slik at mikrokontrolleren ikke blir ødelagt hvis man ved et uhell kobler feil pinne på 5V, fjerne koblingen med RESET istedet for å bøye pinnebeina. Bytte til en boost omformer som gjør om fra 3 V til 5 V på Vin. På den andre siden, ville det krevd en del strøm. Så det å bruke eller lage en egen 9 V batteriholder kunne også vært mulig. På grunn av at høyden på 3D-modellen bommet litt, blir komponentbeinene som er loddet på litt lave. En svakhet med through-hole komponenter som står over PCB-en er at det kan komme en knekk på beina, som fører til at spenningen over dem blir feil eller at beina krysser og lager en short i kretsen. For å rette opp dette kunne man isolert dem med varmepistol

3d-modeller: Fikse høyden i modellene istedet for å bruke antistatiske svamper, nylonspacere og borrelås. Å printe dele på nytt på den gode måten, ville også gjort modellene både sterkere og mer estetiske. Det å dekke modellene med kontaktplast ville også hjulpet på det estetiske og beskytte mot både vann og kaffe. Legge til små detaljer som lokk som skal være over batteriet men ha åpning til knappen og gjennomsiktig beskyttelse over LDR-ene og kanskje LED-ene med tanke på søling av væsker. Dersom modellene skulle kommet på markedet, skulle modellene vært laget av eller dekket med food safe plast(belegg).

6.3 Tilleggsfunksjoner

6.3.1 Legge krem på muffinsene

Som en del av forprosjektet ble det planlagt å implementere en løsning for å påføre krem på kakene. Det var imidlertid kuttet fra oppgaven, på bakgrunn av at det ikke ble definert som nødvendig funksjonalitet for prosjektet

Kapittel 7

Konklusjon

NACHI utfører tilberedningen som planlagt. Alle PCB-ene funker som ønsket. 3D-modellene sitter sammen og funker til formålet. RPi-en bruker BLE-kommunikasjon til å hente verdiene på mikrokontrollerne og sende videre til FlexGUI. Systemets helhet styres fra FlexGUI hvor alt er satt sammen. Systemet fungerer godt nok til å foreta en demonstrasjon, men det er begrenset med kunnskap om systemets pålitelighet over tid.

Bibliografi

[1] A. Edgar-Lund, *Test: Ultimaker 3- Den koster flesk, men du verden så bra den skriver*, Sist besøkt: 27. mai 2023, 2017. adresse: %7Bhttps://www.tek.no/test/i/y3K3Kr/ultimaker-3%7D.

[2] C. S. Nutan Jaeger, *The Best Polycarbonate 3D Printers in 2023*, Sist besøkt: 27. mai 2023, 2022. adresse: %7Bhttps://all3dp.com/2/polycarbonate-pc-3d-printing-all-you-need-to-know/%7D.

[3] Arduino, *Nano 33 BLE Pinout*, Sist besøkt: 27. mai 2023, 2021. adresse: %7Bhttps://content.arduino.cc/assets/Pinout-NANOble_latest.pdf%7D.

[4] N. semiconductor, *nRF52840 Product Specification*, Sist besøkt: 27. mai 2023, 2019. adresse: %7Bhttps://infocenter.nordicsemi.com/pdf/nRF52840_PS_v1.1.pdf%7D.

[5] N. semiconductor, *Welcome to the nRF Connect SDK!* Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/index.html%7D.

[6] Arduino, *Arduino Nano 33 BLE*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://store.arduino.cc/products/arduino-nano-33-ble%7D.

[7] Arduino, *Arduino® Nano 33 BLE datasheet*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://docs.arduino.cc/resources/datasheets/ABX00030-datasheet.pdf%7D.

[8] Arduino, *Nano boards that can be powered directly with 3.3 V*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://support.arduino.cc/hc/en-us/articles/360014735580-Nano-boards-that-can-be-powered-directly-with-3-3-V%7D.

[9] Arduino, *Arduino docs: Nano 33 BLE*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://docs.arduino.cc/hardware/nano-33-ble%7D.

[10] Arduino, *Reset your board*, Sist besøkt: 27. mai 2023, 2022. adresse: %7Bhttps://support.arduino.cc/hc/en-us/articles/5779192727068-Reset-your-board%7D.

[11] Arduino, *Arduino Nano 33 IoT*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://store.arduino.cc/products/arduino-nano-33-iot%7D.

[12] G. Halfacree, *Raspberry Pi Pico W Bluetooth Support Is Just Around the Corner, for Both C/C++ and MicroPython*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://www.hackster.io/news/raspberry-pi-pico-w-bluetooth-support-is-just-around-the-corner-for-both-c-c-and-micropython-592c42d1170a%7D.

[13] M. Electronics, *HLMP-4740*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://no.mouser.com/ProductDetail/Broadcom-Avago/HLMP-4740?qs=jT9z6tsiFNlbkh2eYMYS%2Fw%3D%3D%7D.

[14] M. Electronics, *HLMP-4719*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://no.mouser.com/ProductDetail/Broadcom-Avago/HLMP-4719?qs=jT9z6tsiFNkjZPkCwarxTw%3D%3D%7D.

[15] M. Electronics, *HLMP-4700*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://no.mouser.com/ProductDetail/Broadcom-Avago/HLMP-4700?qs=jT9z6tsiFNlLY0m7YkBilg%3D%3D%7D.

[16] M. Electronics, *NSL-06S53*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://no.mouser.com/ProductDetail/Advanced-Photonix/NSL-06S53?qs=Znm5pLBrcALY7Q9olzqeHw%5C%3D%5C%3D%7D.

[17] M. Electronics, *MHS12304*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://no.mouser.com/ProductDetail/TE-Connectivity-PB/MHS12304?qs=x%5C%2FgbKjZ2T%5C%2FMAxqlZIkgJKQ%5C%3D%5C%3D%7D.

[18] C. Ohlson, *VARTA CR2450 litiumbatteri 3 V*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://www.clasohlson.com/no/VARTA-CR2450-litiumbatteri-3-V/p/32-2239%7D.

[19] renata batteries, *Battery holders*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://www.renata.com/en/products/battery-holders/%7D.

[20] C. Specialists, *How to Calculate Resistor Value for LED Lighting*, Sist besøkt: 27. mai 2023, 2012. adresse: %7Bhttps://www.circuitspecialists.com/blogs/news/how-to-determine-resistor-value-for-led-lighting%7D.

[21] M. Thill, *Choosing a Voltage Divider Resistor for a Light Dependent Resistor*, Sist besøkt: 27. mai 2023, 2017. adresse: %7Bhttps://markusthill.github.io/electronics/choosing-a-voltage-divider-resistor-for-a-ldr/%7D.

[22] D. International, *Zigbee vs. Bluetooth: Choosing the Right Protocol for Your IoT Application*, Sist besøkt: 27. mai 2023, 2021. adresse: %7Bhttps://www.digi.com/blog/post/zigbee-vs-bluetooth-choosing-the-right-protocol%7D.

[23] E. Nesbo, *What Is BLE (Bluetooth Low Energy) and How Does It Work?* Sist besøkt: 27. mai 2023, 2021. adresse: %7Bhttps://www.makeuseof.com/what-is-ble-bluetooth-low-energy/%7D.

[24] J. Marcell, *How Bluetooth Technology Uses Adaptive Frequency Hopping to Overcome Packet Interference*, Sist besøkt: 27. mai 2023, 2020. adresse: %7Bhttps://www.bluetooth.com/blog/how-bluetooth-technology-uses-adaptive-frequency-hopping-to-overcome-packet-interference/%7D.

[25] A. G. I. Mohammed, *The basics of Bluetooth Low Energy (BLE)*, Sist besøkt: 27. mai 2023, 2016. adresse: %7Bhttps://www.edn.com/the-basics-of-bluetooth-low-energy-ble/%7D.

[26] metageek, *ZigBee and Wi-Fi Coexistence*, Sist besøkt: 27. mai 2023, Ukjent. adresse: %7Bhttps://www.metageek.com/training/resources/zigbee-wifi-coexistence/%7D.

[27] C. S. Alliance, *Zigbee FAQ*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://csa-iot.org/all-solutions/zigbee/zigbee-faq/%7D.

[28] R. Yadav, *How to choose between Zigbee and BLE mesh for your IoT application?* Sist besøkt: 27. mai 2023, 2019. adresse: %7Bhttps://www.linkedin.com/pulse/how-choose-between-zigbee-ble-mesh-your-iot-rahul-yadav%7D.

[29] M. Bloechl, *Power Consumption IoT: Bluetooth Zigbee*, Sist besøkt: 27. mai 2023, 2015. adresse: %7Bhttps://www.link-labs.com/blog/bluetooth-zigbee-comparison%7D.

[30] 1NCE, *Licensed vs. unlicensed radio technology*, Sist besøkt: 27. mai 2023, Ukjent. adresse: %7Bhttps://1nce.com/en-eu/resources/news-insights/blog/licensed-vs-unlicensed-radio-technology%7D.

[31] Lovdata, *Forskrift om generelle tillatelser til bruk av frekvenser (fribruksforskriften)*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://lovdata.no/dokument/SF/forskrift/2012-01-19-77/KAPITTEL_3#KAPITTEL_3%7D.

[32] 8. L. N. Guide, *Hardware*, Sist besøkt: 27. mai 2023, 2021. adresse: %7Bhttps://development.libelium.com/868-lp-networking-guide/hardware%7D.

[33] G. M. Djuknic, *Far and Near Fields*, Sist besøkt: 27. mai 2023, 2003. adresse: %7Bhttps://www.wikidata.org/wiki/Q13405516#/media/File:FarNearFields-USP-4998112-1.svg%7D.

[34] Arduino, *ArduinoBLE*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://reference.arduino.cc/reference/en/libraries/arduinoble/ble.central/%7D.

[35] O. Robotics, *rosserial*, Sist besøkt: 27. mai 2023, 2018. adresse: %7Bhttp://wiki.ros.org/rosserial%7D.

[36] micro-ROS, *Supported Hardware*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://micro.ros.org/docs/overview/hardware/%7D.

[37] A. Ezquerro, *ROS2 Tutorials 8: How to communicate between ROS1 ROS2 with ros1$_b$ridge*, Sist besøkt: 27. mai 2023, 2019. adresse: %7Bhttps://www.theconstructsim.com/how-to-communicate-between-ros1-ros2-with-ros1_bridge/%7D.

[38] A. T. LLC, *Using BLE Devices with a Raspberry Pi*, Sist besøkt: 27. mai 2023, Ukjent. adresse: %7Bhttps://www.argenox.com/library/bluetooth-low-energy/using-raspberry-pi-ble/%7D.

[39] C. Ltd, *How to install Ubuntu Server on your Raspberry Pi*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://ubuntu.com/tutorials/how-to-install-ubuntu-on-your-raspberry-pi#1-overview%7D.

[40] O. robotics, *Ubuntu install of ROS Noetic*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttp://wiki.ros.org/noetic/Installation/Ubuntu%7D.

[41] D. McKay, *How to Set Up Bluetooth on Linux*, Sist besøkt: 27. mai 2023, 2022. adresse: %7Bhttps://www.howtogeek.com/829360/how-to-set-up-bluetooth-on-linux/%7D.

[42] ask Ubuntu, *Bluetooth not working on Raspberry Pi (Ubuntu 20.04)*, Sist besøkt: 27. mai 2023, 2021. adresse: %7Bhttps://askubuntu.com/questions/1246723/bluetooth-not-working-on-raspberry-pi-ubuntu-20-04%7D.

[43] S. E. U.
bibinitperiod Linux, *Bluetoothctl: No default controller available, despite being unblocked*, Sist besøkt: 27. mai 2023, 2014. adresse: %7Bhttps://unix.stackexchange.com/questions/169931/bluetoothctl-no-default-controller-available-despite-being-unblocked%7D.

[44] M. Ballard, *Connect to a Bluetooth Device via the Terminal*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://www.baeldung.com/linux/bluetooth-via-terminal%7D.

[45] B. Solomon, *Async IO in Python: A Complete Walkthrough*, Sist besøkt: 27. mai 2023, 2019. adresse: %7Bhttps://realpython.com/async-io-python/%7D.

[46] H. Blidh, *bleak*, Sist besøkt: 27. mai 2023, 2023. adresse: %7Bhttps://bleak.readthedocs.io/en/latest/%7D.

[47] A. Anwar, *Part 2: 7 Simple Steps to Create and Build Your First ROS Package*, Sist besøkt: 27. mai 2023, 2017. adresse: %7Bhttps://medium.com/swlh/7-simple-steps-to-create-and-build-our-first-ros-package-7e3080d36faa%7D.

[48] O. Robotics, *Creating a ROS Package*, Sist besøkt: 27. mai 2023, 2013. adresse: %7Bhttp://wiki.ros.org/catkin/Tutorials/CreatingPackage%7D.

[49] O. Robotics, *Using numpy with rospy*, Sist besøkt: 27. mai 2023, 2019. adresse: %7Bhttp://wiki.ros.org/rospy_tutorials/Tutorials/numpy%7D.

[50] NACHI-FUJIKOSHI CORP., *Manipulator Instruction Manual MZ04-01/MZ04E-01 [CFD] MZ04D-01/MZ04DE-01 [CFD]*, 8. utg., 1-1-1, Fujikoshihonmachi, Toyama City, Japan 930-8511.

[51] Wegard GmbH, *SoftGripping Product Catalog 2022*, Kirchenheide 18, 22395 Hamburg, Germany. adresse: https://soft-gripping.com/assets/downloads/SoftGripping_Catalogue.pdf.

[52] SMC CORPORATION, *5-Port Solenoid Valve*, Akihabara UDX 15F, 4-14-1, Sotokanda, Chiyoda-ku, Tokyo 101-0021 JAPAN. adresse: https://www.smcpneumatics.com/pdfs/SY.New.pdf.

[53] SMC CORPORATION, *Operation manual Digital pressure switch for energy-saving control ejector*, Akihabara UDX 15F, 4-14-1, Sotokanda, Chiyoda-ku, Tokyo 101-0021 JAPAN. adresse: https://www.smcworld.com/assets/manual/en-jp/files/ZK2-E.pdf.

[54] SMC CORPORATION, *3 Port Solenoid Valve Direct Operated Poppet Type*, Akihabara UDX 15F, 4-14-1, Sotokanda, Chiyoda-ku, Tokyo 101-0021 JAPAN. adresse: https://www.smcpneumatics.com/pdfs/VT307.pdf.

[55] NACHI-FUJIKOSHI CORP., *CFD/CFDL/CCZ Controller Instruction Manual Option (I/O Connection)*, 5. utg., 1-1-1, Fujikoshihonmachi, Toyama City, Japan 930-8511.

[56] NACHI-FUJIKOSHI CORP., *FD/CFD Controller Instruction Manual Robot Monitoring Unit RMU20-20/30/40*, 10. utg., 1-1-1, Fujikoshihonmachi, Toyama City, Japan 930-8511.

[57] J. Carette, *What are the Present Safety Requirements for Collaborative Robots?* Sist besøkt: 28. mai 2023, 2014. adresse: https://blog.robotiq.com/bid/70004/What-are-the-Present-Safety-Requirements-for-Collaborative-Robots.

[58] Kompaï Robotics, *User manual Kompaï-CARE*.

Vedlegg A

Vedlegg

- Standardavtale
- Plakat
- Brukermanual
- Sensorsystem
- ROS kode

NTNU

Norges teknisk-naturvitenskapelige universitet

Fastsatt av prorektor for utdanning 10.12.2020

STANDARDAVTALE

om utføring av studentoppgave i samarbeid med ekstern virksomhet

Avtalen er ufravikelig for studentoppgaver (heretter oppgave) ved NTNU som utføres i samarbeid med ekstern virksomhet.

Forklaring av begrep

Opphavsrett
Er den rett som den som skaper et åndsverk har til å fremstille eksemplar av åndsverket og gjøre det tilgjengelig for allmennheten. Et åndsverk kan være et litterært, vitenskapelig eller kunstnerisk verk. En studentoppgave vil være et åndsverk.

Eiendomsrett til resultater
Betyr at den som eier resultatene bestemmer over disse. Utgangspunktet er at studenten eier resultatene fra sitt studentarbeid. Studenten kan også overføre eiendomsretten til den eksterne virksomheten.

Bruksrett til resultater
Den som eier resultatene kan gi andre en rett til å bruke resultatene, f.eks. at studenten gir NTNU og den eksterne virksomheten rett til å bruke resultatene fra studentoppgaven i deres virksomhet.

Prosjektbakgrunn
Det partene i avtalen har med seg inn i prosjektet, dvs. som vedkommende eier eller har rettigheter til fra før og som brukes i det videre arbeidet med studentoppgaven. Dette kan også være materiale som tredjepersoner (som ikke er part i avtalen) har rettigheter til.

Utsatt offentliggjøring
Betyr at oppgaven ikke blir tilgjengelig for allmennheten før etter en viss tid, f.eks. før etter tre år. Da vil det kun være veileder ved NTNU, sensorene og den eksterne virksomheten som har tilgang til studentarbeidet de tre første årene etter at studentarbeidet er innlevert.

KM

AW

R.K.

1. Avtaleparter

Norges teknisk-naturvitenskapelige universitet (NTNU) Institutt: Institutt for teknisk kybernetikk
Veileder ved NTNU: ***Sigurd Gossé*** Sigurd.gosse@gmail.com 91 76 61 06
Ekstern virksomhet: ***PPM Robotics AS*** Ekstern virksomhet sin kontaktperson, e-post og tlf.: ***Trygve Thomessen*** tth@ppm.no 92 24 21 89
Student: ***Renate Heierstad Klemetsdal*** Fødselsdato: 2000-12-26
Student: ***Karoline Margrethe Mørkeng*** Fødselsdato: 1998-08-23
Student: **Astrid Meen Wold** Fødselsdato: 2000-08-01

Partene har ansvar for å klarere eventuelle immaterielle rettigheter som studenten, NTNU, den eksterne eller tredjeperson (som ikke er part i avtalen) har til prosjektbakgrunn før bruk i forbindelse med utførelse av oppgaven. Eierskap til prosjektbakgrunn skal fremgå av eget vedlegg til avtalen der dette kan ha betydning for utførelse av oppgaven.

2. Utførelse av oppgave

Studenten skal utføre: (sett kryss)

Masteroppgave	
Bachelor-oppgave	**X**
Prosjektoppgave	
Annen oppgave	

Startdato: **2023-01-09**
Sluttdato: **2023-05-22**

Oppgavens arbeidstittel er: ***Fremtidens sykehjem med service-roboter og informasjonsteknologi – Oppgave 2 Robotisert tilberedning av kaffe og kaker***

Ansvarlig veileder ved NTNU har det overordnede faglige ansvaret for utforming og godkjenning av prosjektbeskrivelse og studentens læring.

3. Ekstern virksomhet sine plikter

Ekstern virksomhet skal stille med en kontaktperson som har nødvendig faglig kompetanse til å gi studenten tilstrekkelig veiledning i samarbeid med veileder ved NTNU. Ekstern kontaktperson fremgår i punkt 1.

Formålet med oppgaven er studentarbeid. Oppgaven utføres som ledd i studiet. Studenten skal ikke motta lønn eller lignende godtgjørelse fra den eksterne for studentarbeidet. Utgifter knyttet til gjennomføring av oppgaven skal dekkes av den eksterne. Aktuelle utgifter kan for eksempel være reiser, materialer for bygging av prototyp, innkjøp av prøver, tester på lab, kjemikalier. Studenten skal klarere dekning av utgifter med ekstern virksomhet på forhånd.

Ekstern virksomhet skal dekke følgende utgifter til utførelse av oppgaven: ***Kostnader leie av laboratorium samt nødvendig laboratorieutstyr, materialer / forbruksartikler, faglig konsultering og eventuelle reiser i forbindelse med gjennomføring av oppgaven.***

Dekning av utgifter til annet enn det som er oppført her avgjøres av den eksterne underveis i arbeidet.

4. Studentens rettigheter

Studenten har opphavsrett til oppgaven[1]. Alle resultater av oppgaven, skapt av studenten alene gjennom arbeidet med oppgaven, eies av studenten med de begrensninger som følger av punkt 5, 6 og 7 nedenfor. Eiendomsretten til resultatene overføres til ekstern virksomhet hvis punkt 5 b er avkrysset eller for tilfelle som i punkt 6 (overføring ved patenterbare oppfinnelser).

I henhold til lov om opphavsrett til åndsverk beholder alltid studenten de ideelle rettigheter til eget åndsverk, dvs. retten til navngivelse og vern mot krenkende bruk.

Studenten har rett til å inngå egen avtale med NTNU om publisering av sin oppgave i NTNUs institusjonelle arkiv på Internett (NTNU Open). Studenten har også rett til å publisere oppgaven eller deler av den i andre sammenhenger dersom det ikke i denne avtalen er avtalt begrensninger i adgangen til å publisere, jf. punkt 8.

5. Den eksterne virksomheten sine rettigheter

Der oppgaven bygger på, eller videreutvikler materiale og/eller metoder (prosjektbakgrunn) som eies av den eksterne, eies prosjektbakgrunnen fortsatt av den eksterne. Hvis studenten skal utnytte resultater som inkluderer den eksterne sin prosjektbakgrunn, forutsetter dette at det er inngått egen avtale om dette mellom studenten og den eksterne virksomheten.

[1] Jf. Lov om opphavsrett til åndsverk mv. av 15.06.2018 § 1

RJK

KM

AVV

Alternativ a) (sett kryss) Hovedregel

	Ekstern virksomhet skal ha bruksrett til resultatene av oppgaven

Dette innebærer at ekstern virksomhet skal ha rett til å benytte resultatene av oppgaven i egen virksomhet. Retten er ikke-eksklusiv.

Alternativ b) (sett kryss) Unntak

X	***Ekstern virksomhet skal ha eiendomsretten til resultatene av oppgaven og studentens bidrag i ekstern virksomhet sitt prosjekt***

Begrunnelse for at ekstern virksomhet har behov for å få overført eiendomsrett til resultatene:

Oppgavene gjennomføres som en del av PPMs strategiske utvikling mot markedet for service-roboter innen sykehjem. PPMs investorer har lagt betydelige investeringer inn i dette, og krever full tilgang til mulig videre utvikling av bedriften forretningsvirksomhet, med bakgrunn i de resultatene som kommer ut av prosjektet.

Studentene har på sin side, mulighet til vitenskapelig publisering av resultatene, så fremt dette er forelagt PPM til godkjenning i forkant, og referanse til PPM Robotics AS nevnes under "Acknowledgement". Studentene kan også benytte de generiske resultatene fra prosjektet i forbindelse med videre studier.

6. Godtgjøring ved patenterbare oppfinnelser

Dersom studenten i forbindelse med utførelsen av oppgaven har nådd frem til en patenterbar oppfinnelse, enten alene eller sammen med andre, kan den eksterne kreve retten til oppfinnelsen overført til seg. Dette forutsetter at utnyttelsen av oppfinnelsen faller inn under den eksterne sitt virksomhetsområde. I så fall har studenten krav på rimelig godtgjøring. Godtgjøringen skal fastsettes i samsvar med arbeidstakeroppfinnelsesloven § 7. Fristbestemmelsene i § 7 gis tilsvarende anvendelse.

7. NTNU sine rettigheter

De innleverte filer av oppgaven med vedlegg, som er nødvendig for sensur og arkivering ved NTNU, tilhører NTNU. NTNU får en vederlagsfri bruksrett til resultatene av oppgaven, inkludert vedlegg til denne, og kan benytte dette til undervisnings- og forskningsformål med de eventuelle begrensninger som fremgår i punkt 8.

8. Utsatt offentliggjøring

Hovedregelen er at studentoppgaver skal være offentlige.

Sett kryss

X	***Oppgaven skal være offentlig***

I særlige tilfeller kan partene bli enige om at hele eller deler av oppgaven skal være undergitt utsatt offentliggjøring i maksimalt tre år. Hvis oppgaven unntas fra offentliggjøring, vil den kun være tilgjengelig for student, ekstern virksomhet og veileder i denne perioden. Sensurkomiteen vil ha tilgang til oppgaven i forbindelse med sensur. Student, veileder og sensorer har taushetsplikt om innhold som er unntatt offentliggjøring.

Oppgaven skal være underlagt utsatt offentliggjøring i (sett kryss hvis dette er aktuelt):

Sett kryss		Sett dato
	ett år	
	to år	
	tre år	

Behovet for utsatt offentliggjøring er begrunnet ut fra følgende:

Dersom partene, etter at oppgaven er ferdig, blir enig om at det ikke er behov for utsatt offentliggjøring, kan dette endres. I så fall skal dette avtales skriftlig.

Vedlegg til oppgaven kan unntas ut over tre år etter forespørsel fra ekstern virksomhet. NTNU (ved instituttet) og student skal godta dette hvis den eksterne har saklig grunn for å be om at et eller flere vedlegg unntas. Ekstern virksomhet må sende forespørsel før oppgaven leveres.

De delene av oppgaven som ikke er undergitt utsatt offentliggjøring, kan publiseres i NTNUs institusjonelle arkiv, jf. punkt 4, siste avsnitt. Selv om oppgaven er undergitt utsatt offentliggjøring, skal ekstern virksomhet legge til rette for at studenten kan benytte hele eller deler av oppgaven i forbindelse med jobbsøknader samt videreføring i et master- eller doktorgradsarbeid.

Generelt

Denne avtalen skal ha gyldighet foran andre avtaler som er eller blir opprettet mellom to av partene som er nevnt ovenfor. Dersom student og ekstern virksomhet skal inngå avtale om konfidensialitet om det som studenten får kjennskap til i eller gjennom den eksterne virksomheten, kan NTNUs standardmal for konfidensialitetsavtale benyttes.

Den eksterne sin egen konfidensialitetsavtale, eventuell konfidensialitetsavtale den eksterne har inngått i samarbeidprosjekter, kan også brukes forutsatt at den ikke inneholder punkter i motstrid med denne avtalen (om rettigheter, offentliggjøring mm). Dersom det likevel viser seg at det er motstrid, skal NTNUs standardavtale om utføring av studentoppgave gå foran. Eventuell avtale om konfidensialitet skal vedlegges denne avtalen.

KM RK AW

Eventuell uenighet som følge av denne avtalen skal søkes løst ved forhandlinger. Hvis dette ikke fører frem, er partene enige om at tvisten avgjøres ved voldgift i henhold til norsk lov. Tvisten avgjøres av sorenskriveren ved Sør-Trøndelag tingrett eller den han/hun oppnevner.

Denne avtale er signert i fire eksemplarer hvor partene skal ha hvert sitt eksemplar. Avtalen er gyldig når den er underskrevet av NTNU v/instituttleder.

Signaturer:

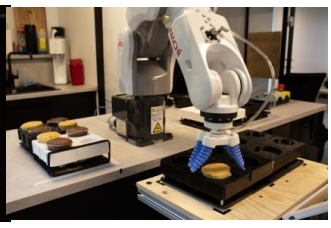
Instituttleder: Dato:
Veileder ved NTNU: Dato:
Ekstern virksomhet: Dato: 2023-03-01
Student: Kardine M. Mørkeng Dato: 01.03.2023
Student: Renate H. Klemetsdal Dato: 01.03.2023
Student: Astrid Meen Vold Dato: 01.03.2023

Fremtidens eldreomsorg:

Robotisert tilberedning av kaffe og kake

Mangelen på pleiere i eldreomsorgen er et stadig økende problem i Norge. I tillegg går landet en eldrebølge i møte. Pleierne er kritiske for de eldres velvære, og det er kritisk at pleierne får bruke tiden sin på det som teller mest. Det er mange oppgaver som tar av pleierne verdifulle tid, som likesågodt kunne vært utført av roboter. For eksempel tilberedning og servering av kaffe og kake. På bakgrunn av dette er det utviklet et system for *automatisert tilberedning av kaffe og kake*.

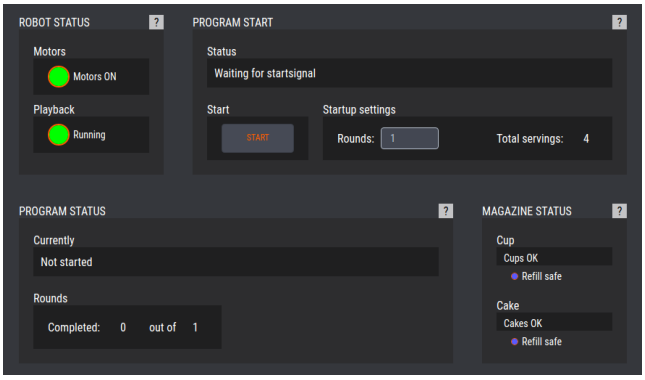
En industrirobot, NACHI MZ04, står montert på en kjøkkenbenk. Den er utstyrt med en pneumatisk fingergriper og er i stand til å gripe både kopper og kaker. Roboten plasserer kopper i kaffemaskinen og plasserer kake på et serveringsbrett. Når kaffekoppen er fylt, blir den plassert på serverings-brettet. Serveringsbrettet er montert på serviceroboten, Kompaï K3. Den er I stand til å navigere på laben.



Smart magasinering

For å øke systemets fleksibilitet i påfyll av magasiner er det utviklet et sensorsystem for smart magasinering. Sensorene vet til enhver tid statusen for kopper og kaker i magasinene. Sensordataen prosesseres av en mikrokontroller i hver av magasinene. Det er to magasiner for kaffe og to for kake, til sammen er magasinstørrelsen åtte.

Brukergrensesnitt



I brukergrensesnittet knyttes systemet sammen. Her blir data fra alle enhetene hentet inn gjennom ROS. Status for magasin, robot og prosess kan leses av i brukergrensesnittet. Det kan bestemmes antall serveringer som skal utføres ved å bestemme antall runder. Det er også utviklet en brukermanual som gjør operasjon av systemet lett, selv for de uten faglig kunnskap innen programmering og robotikk.

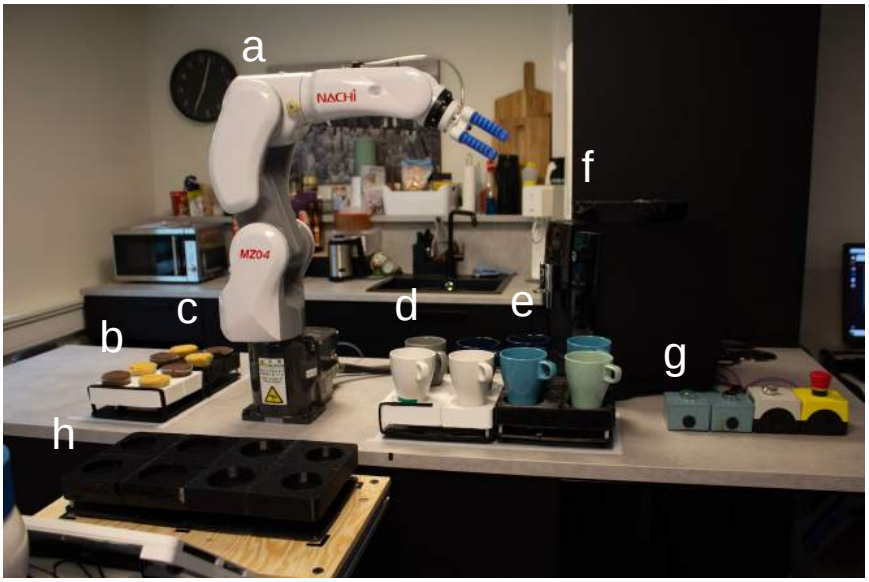
Brukermanual

System for tilberedning og servering

> **Viktig**
>
> Les hele dette dokumentet før du opererer systemet

Innhold

1 Komponenter



Figur 1: Oppsett av komponenter på kjøkkenbenken.

Med unntak av serveringsbrettet på serviceroboten Kompaï, er alle komponentene på kjøkkenbenken. De er plassert slik som i figur 1.

Figur 1	Navn	Beskrivelse
a	NACHI MZ04	Industriroboten som utfører tilberedningen
b	Kakemagasin 1	Magasin for kaker (Unit 2)
c	Kakemagasin 2	Magasin for kaker (Unit 1)
d	Koppmagasin 1	Magasin for kopper (Unit 1)
e	Koppmagasin 2	Magasin for kopper (Unit 2)
f	Kaffemaskin	En standard knappestyrt kaffemaskin
g	Kontrollpanel	Kontrollpanel med fire brytere for kontroll av roboten
h	Serveringsbrett	Serveringsbrett festet på baksiden av Kompaï

Tabell 1: Oversikt over komponenter på benken fra figur 1

1.1 NACHI MZ04

Kontrollenhet
Kontrollenheten står under kjøkkenbenken, og må skrus på før operasjon av systemet er mulig.

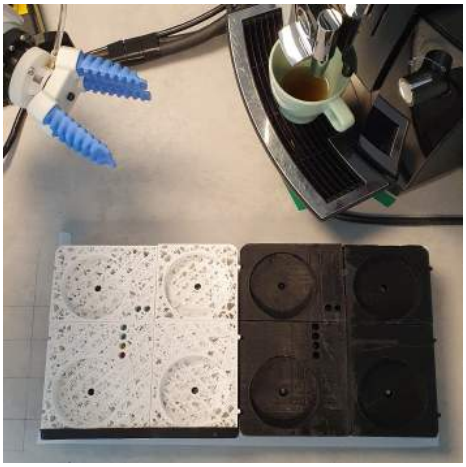
Håndkontroller
Håndkontrolleren brukes for å lese av statusen til roboten, spesifikt feilmeldinger.

1.2 Magasiner

Det er magasiner for kopper, kaker og kaffe. Kaffen består av magasin for vann og kaffebønner. Alle magasinene må være påslått og ha tilstrekkelig med innhold for at systemet skal kunne ta dem i bruk.

1.2.1 Kopper

Koppholderne står til høyre for industriroboten, foran kaffemaskinen. Den hvite holderen er enhet 1 (Unit 1), mens den svarte er enhet 2 (Unit 2). Strømbryteren er plassert på undersiden.

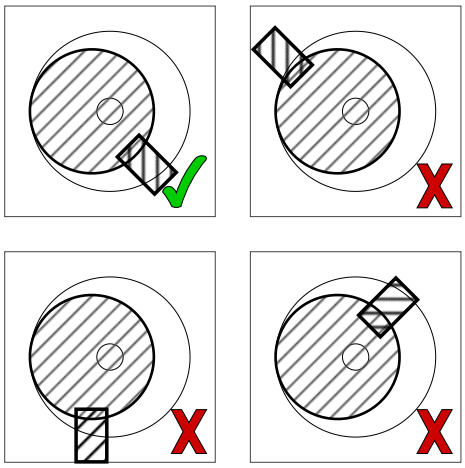


(a)

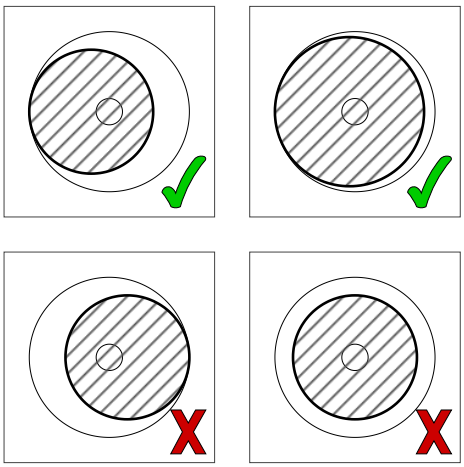
(b)

Figur 2: Fotografi av koppholdere og strømbryter

Når koppene plasseres i koppholderne, er det viktig at de plasseres med hanken slik som illustrert i figur 3a. Videre skal koppene plasseres i hullet slik at de står som i figur 3b. Dersom koppens radius utvider seg og avviker mye fra bunnens areal, skal koppen helst plasseres slik at den største radiusen ikke går for lang ut over kanten. Koppene kan ha en maksimal høyde på 10,5 cm, men kan ikke være lavere enn 9 cm. Videre må den nedre diameteren få plass i koppholderen, og den øvre diameteren ikke overstige 8,5 cm.



(a) Vinkelen til koppens hank i koppmagasinet.



(b) Plassering av kopp i koppmagasinet.

Figur 3

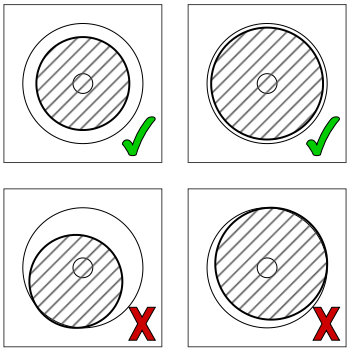
1.2.2 Kaker

Kakeholderne står til venstre for industriroboten. Det samme prinsippet gjelder hvor den hvite holderen er enhet 1 (Unit 1), mens den svarte er enhet 2 (Unit 2). Videre er strømbryteren på samme plass som på koppholderne.

(a) (b)

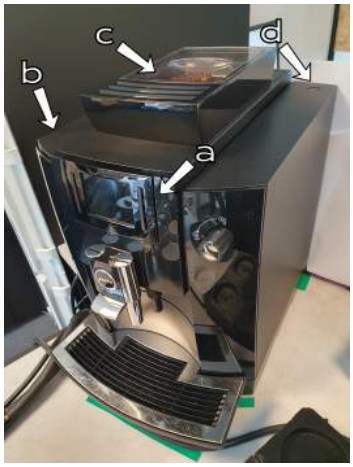
Figur 4: Fotografi av kakeholdere og strømbryter

Når kaker plasseres i magasinet skal de sentreres i midten av hullene. Kaker som er under 2 cm kan ikke gripes. Videre må de passe inn i hullene på holderen. Unngå å plassere kaker som avviker mye hullenes form.



Figur 5: Hvordan kakene skal plasseres i kakeholderne (sett ovenfra).

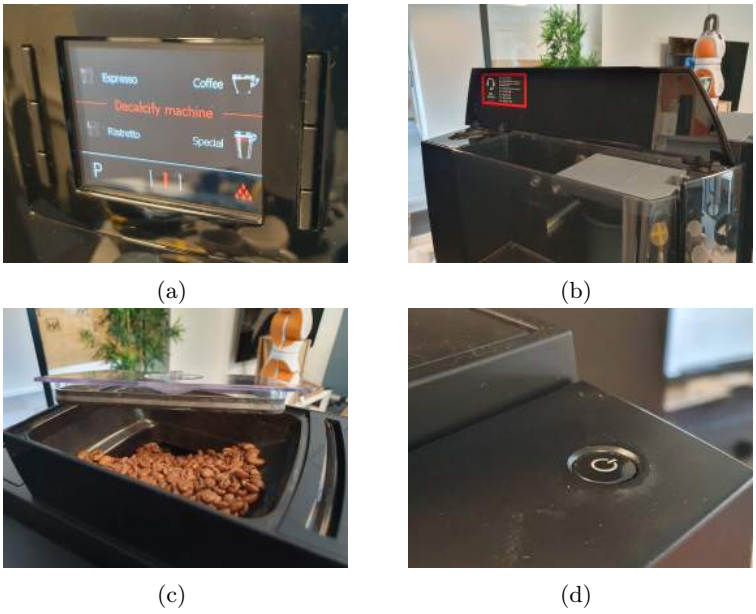
1.2.3 Kaffe



Figur 6: Kaffemaskinen. a) Skjerm, b) Vannmagasin, c) Kaffebønner, d) Bryter)

Kaffemaskinen har magasin for vann (figur 7b) og for kaffebønner (figur 7c). Begge magasinene må ha tilstrekkelig med innhold for at det skal være mulig for maskinen å produsere kaffe. Videre må kaffemaskinen være koblet til strøm og påslått. Den skrus på med knappen på toppen (figur 7d). Kaffemaskinen krever også jevnlig vedlikehold for å fungere. Kaffegrut må tømmes væske etter rens, og kaffegrut etter hver kopp. Dette avfallet må tømmes jevnlig, da kaffemaskinen ikke fungerer ellers.

Ved behov for vedlikehold eller fylling av magasin, vil skjermen (figur 7a) vise en melding om hva som må gjøres. Manuell operasjon av kaffemaskinen er mulig ved å benytte knappene på siden av skjermen. Det er tillatt å operere kaffemaskinen manuelt når systemet ikke kjører. Kaffemaskinen har ingen elektronisk integrasjon med de andre systemkomponentene, og vil dermed fungere uavhengig av resten av systemet. Det vil også bety at systemet ikke har noen måte å kommunisere med kaffemaskinen på, og den krever manuell oppfølging.



(a) (b)

(c) (d)

Figur 7: Modellering av systemets oppsett i NHL (modellert i FDonDesk).

1.3 Kontrollpanel

Figur 8: Kontrollpanel med 4 brytere (fra venstre: Start, Stopp, Reset, Nødstopp)

Roboten er utstyrt med et kontrollpanel med fire brytere i form av trykknapper. Disse kan benyttes for direkte styring av roboten. Kontrollpanelet er avbildet i figur 8. Knappenes funksjon er beskrevet i tabell 2.

Navn	Funksjon (når trykket)
START	Dersom motoren er av, vil et knappetrykk skru på motorene. Dersom motorene er på, vil et knappetrykk starte hovedprogrammet.
STOPP	Dersom roboten kjører, vil et knappetrykk stoppe roboten. Hvis roboten ikke kjører, vil ikke knappen gjøre noe. Denne knappen har ingen påvirkning på motoren.
RESET	Et enkelt knappetrykk vil nullstille eventuelle feilmeldinger. Hvis knappen holdes inne i to sekunder, vil programsekvensen nullstilles. Når programsekvensen er nullstilt, vil roboten begynne fra toppen av hovedprogrammet neste gang START trykkes.
NØDSTOPP	NØDSTOPP-knappen stopper roboten umiddelbart, og igangsetter systemets sikkerhetsfunksjoner.

Tabell 2: Oversikt over kontrollpanelets funksjoner

2 Oppstart

Før oppstart må systemets komponenter være koblet til strøm og påslått. I figur 3 er en oversikt over komponentenes krav for strømtilførsel. De som trenger ekstern strømtilgang må kobles med støpsel i stikkontakt. For de komponentene som har egen strømbryter må operatør forsikre seg om at disse er påslått.

Komponent	Ekstern strømtilgang	Strømbryter
NACHI Kontrollenhet	Ja	Ja
NACHI Robot Monitoring Unit	Ja	Nei
Kompressor	Ja	Ja
Kaffemaskin	Ja	Ja
Kakemagasin 1	Nei	Ja
Kakemagasin 2	Nei	Ja
Koppmagasin 1	Nei	Ja
Koppmagasin 2	Nei	Ja

Tabell 3: Oversikt over nødvendig strømtilførsel for systemet

(a) (b)

Figur 9: Bryter på kontrollenhet og håndkontroller.

Når alle enhetene er koblet til strøm og påslått, forsikre deg om at både kontrollenheten og håndkontrolleren står i auto-modus, slik som i figur 9.

Både kake- og koppmagasinene har strømbryter på undersiden. NACHIs kontrollenhet skrus på med bryteren på baksiden av kabinettet. Kompressoren har bryter på toppen. Når alle enhetene er klare, og systemet er i auto, kan start-knappen trykkes.

Trykk på startknappen for å starte motorene. Når motorene er på, trykk på startknappen igjen for å kjøre programmet. Dersom programmet ble avsluttet før den ble kjørt ferdig, vil programmet gjenopptas der den ble stoppet.

Roboten kan når som helst stoppes med stopp-knappen. Det startes igjen med start-knappen. Programmet vil da gjenopptas der det ble stoppet.

Dersom programmet skal starte fra begynnelsen av, når den allerede er påbegynt, kan reset-knappen benyttes. Hold inne reset-knappen i to sekunder for å nullstille programmet. Hvis NACHI er langt unna startposisjonen, må den stilles i manuell-modus, og jogges til en posisjon for startposisjonen kan nås uten kollisjoner.

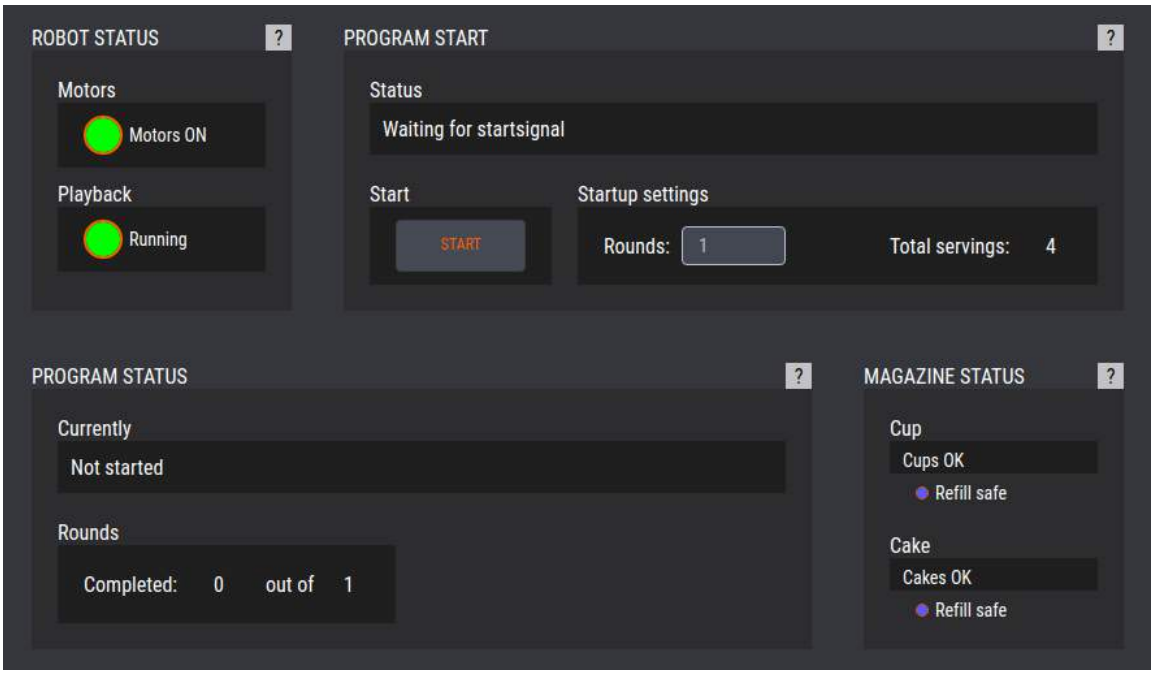
i

Advarsler

- Ikke stopp programmet mens griperen er tilført vakuum eller lufttrykk.
- Om det skulle forekomme strømbrudd mens griperen er tilført trykkluft, fjern umiddelbart enheter fra griperen.
- Ved oppstart etter strømbrudd, vil programmet fortsette der det ble avsluttet. Hvis griperen var aktiv da strømbruddet skjedde, må programmet nullstilles.

3 Operasjon

Brukergrensesnittet kjøres gjennom Google Chrome i Ubuntu. Den er kun tilgjengelig når serveren kjører. Der kan status for roboten observeres, og programmet kan styres. Trykk på spørsmålstegnene i brukergrensesnittet for å få informasjon om rutene. Se figur 10 for bilde av brukergrensesnittet.



Figur 10: Skjermdump av brukergrensesnittet i FlexGUI.

Før programmet kan startes, må motorene være på og programmet kjøres i playback modus. For å starte programmet trykker du på startknappen. Om en nullstilling er nødvendig vil dette stå i statusfeltet i startruten. Før start trykkes kan antall runder spesifiseres. Én runde består av en servering av fire kopper kaffe og fire kaker.

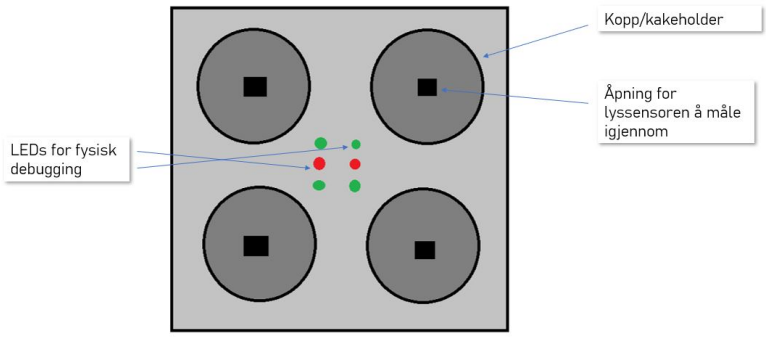
Dersom det er tomt for kopper eller kaker, vil det stå i ruten for magasinstatus og et popup-vindu vil dukke opp. I denne ruten vil det også stå når det egner seg å fylle magasinene. Ved påfyll av magasiner, stopp roboten først.

> **Advarsel**
>
> Stopp alltid roboten før det foretas påfyll av magasiner!

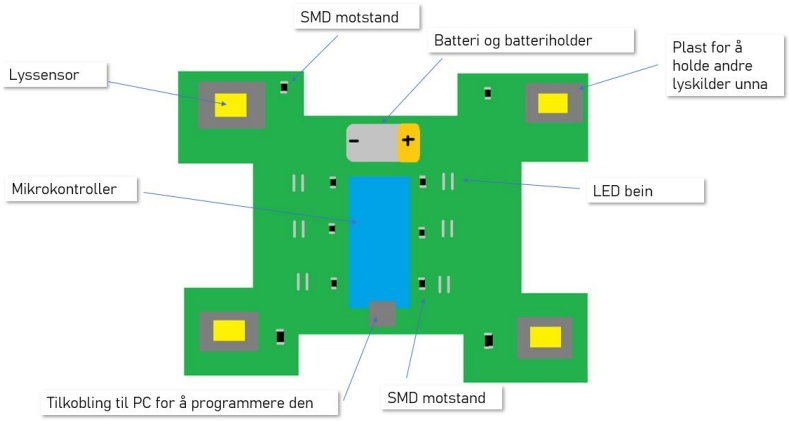
Utenom statusen for magasiner er det mulig å lese av følgende informasjon:

- Om robotens motorer er av eller på (Motors).
- Om roboten kjører i playback modus, eller om den er stoppet (Playback).
- Hvorvidt systemet venter på startsignal, er startet, er i teach modus eller trenger nullstilling (Program Start Status).
- Hvilken del av programmet roboten kjører, eventuelt hvilken del som ble kjørt sist eller om systemet er blitt nullstilt (Program Status Currently).
- Hvor mange runder som ble valgt i startinnstillingene før start ble trykket, og hvor mange runder som er ferdig kjørt hittil (Program Status Rounds).

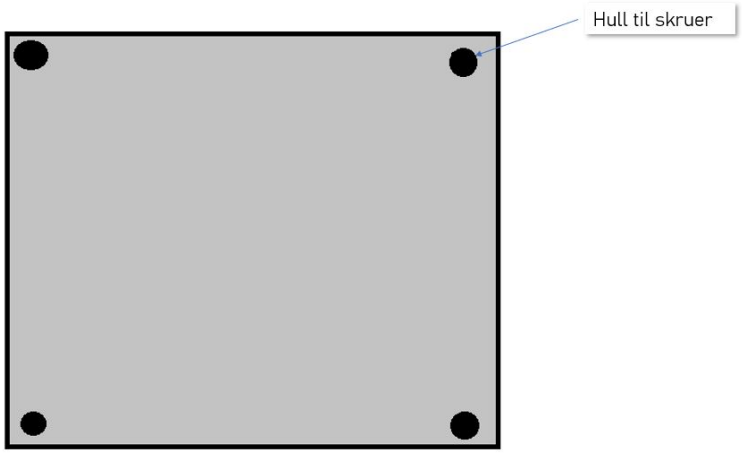
A.0.1 Konseptskisser



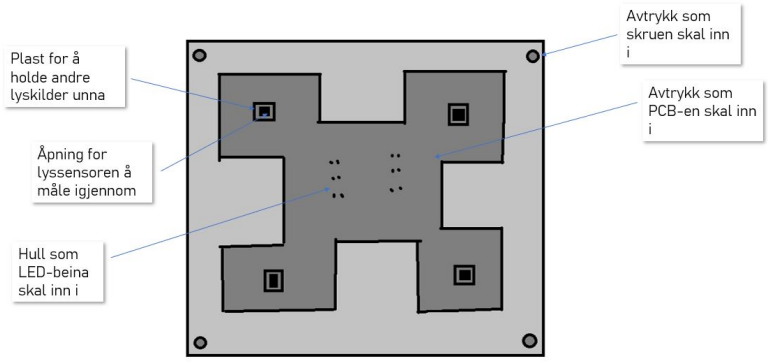
Figur A.1: Øverste lag av forsiden



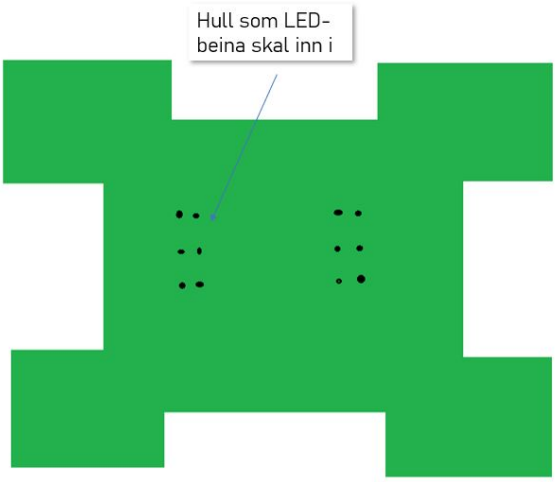
Figur A.2: Midterste lag av forsiden



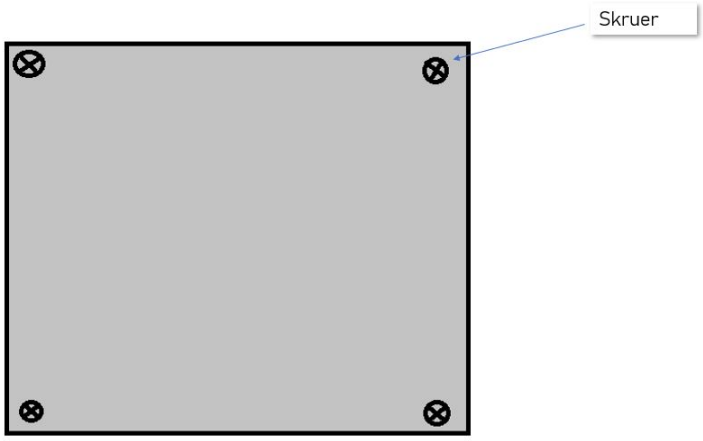
Figur A.3: Nederste lag av forsiden



Figur A.4: Øverste lag av baksiden

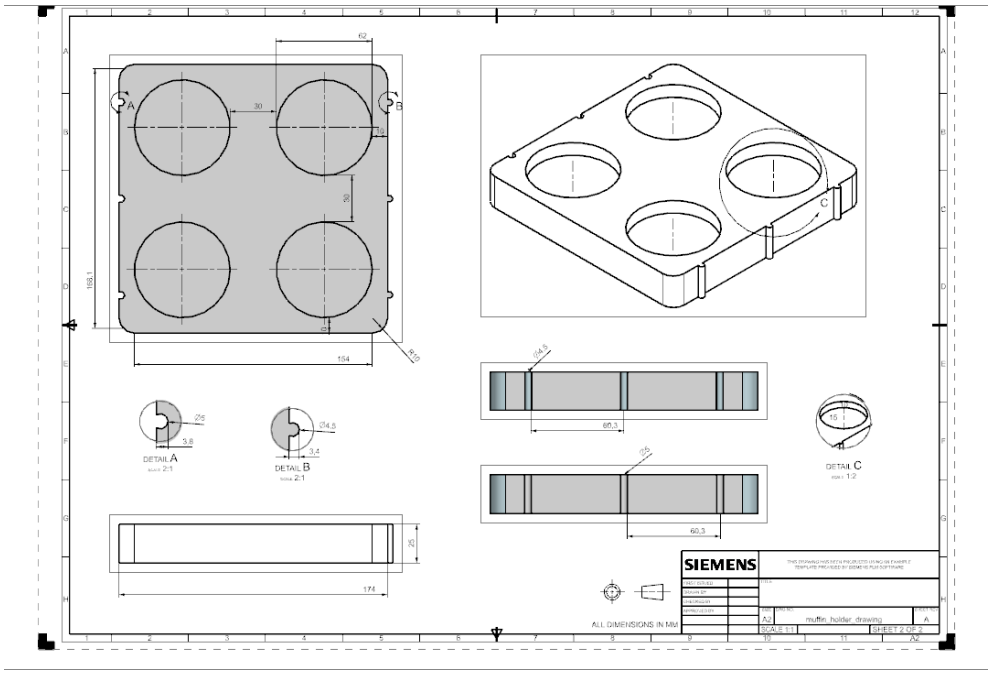


Figur A.5: Midterste lag av baksiden

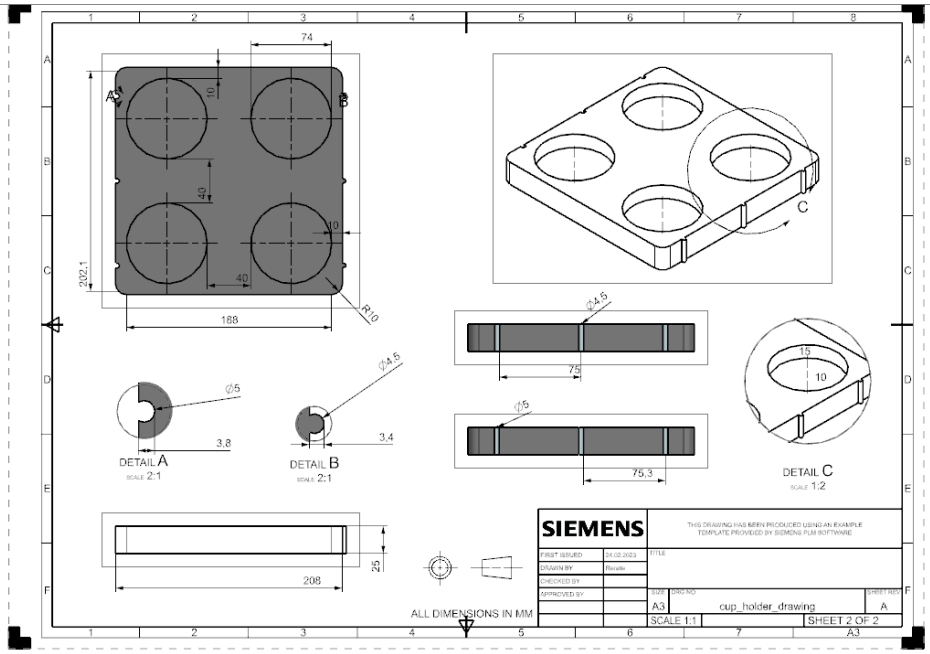


Figur A.6: Nederste lag av baksiden

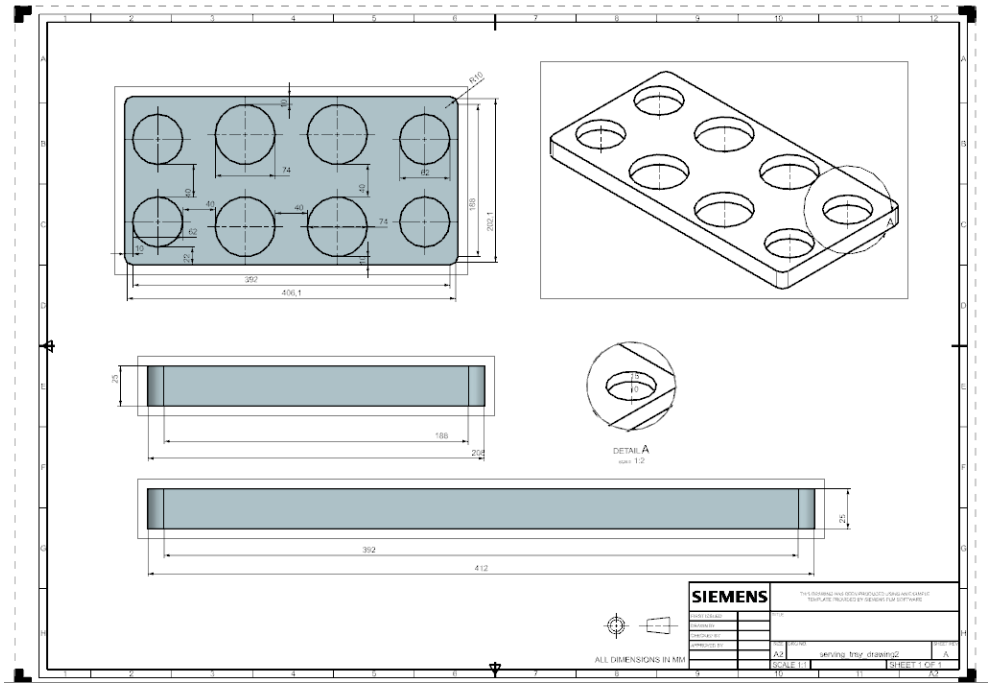
A.0.2 3D-modell tegninger



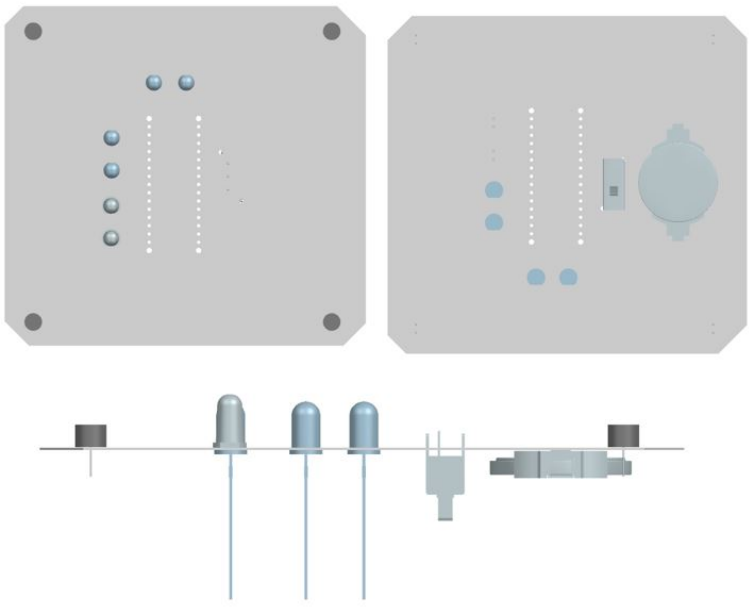
Figur A.7: Dimensjoner for kakeholder



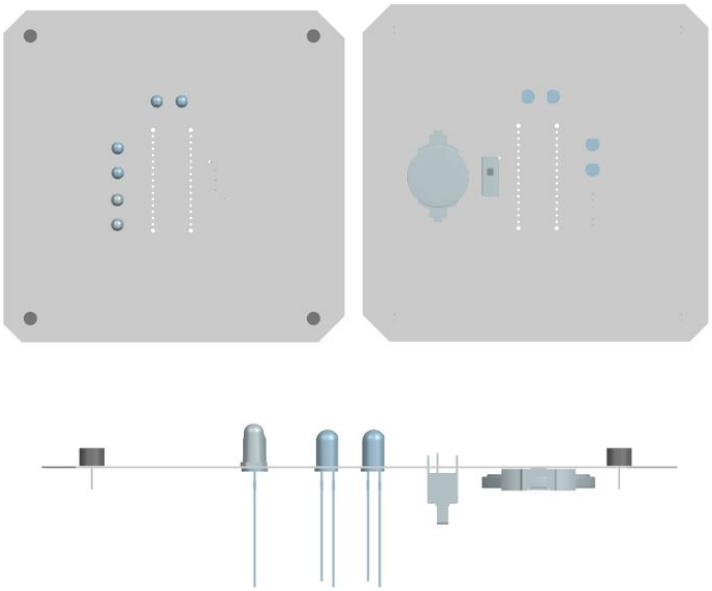
Figur A.8: Dimensjoner for koppholder



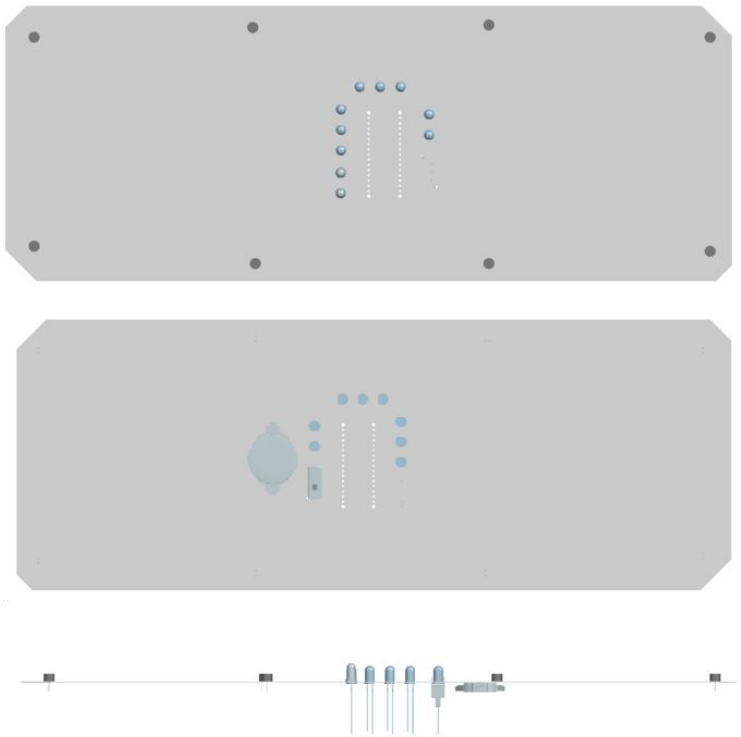
Figur A.9: Dimensjoner for serveringsbrett



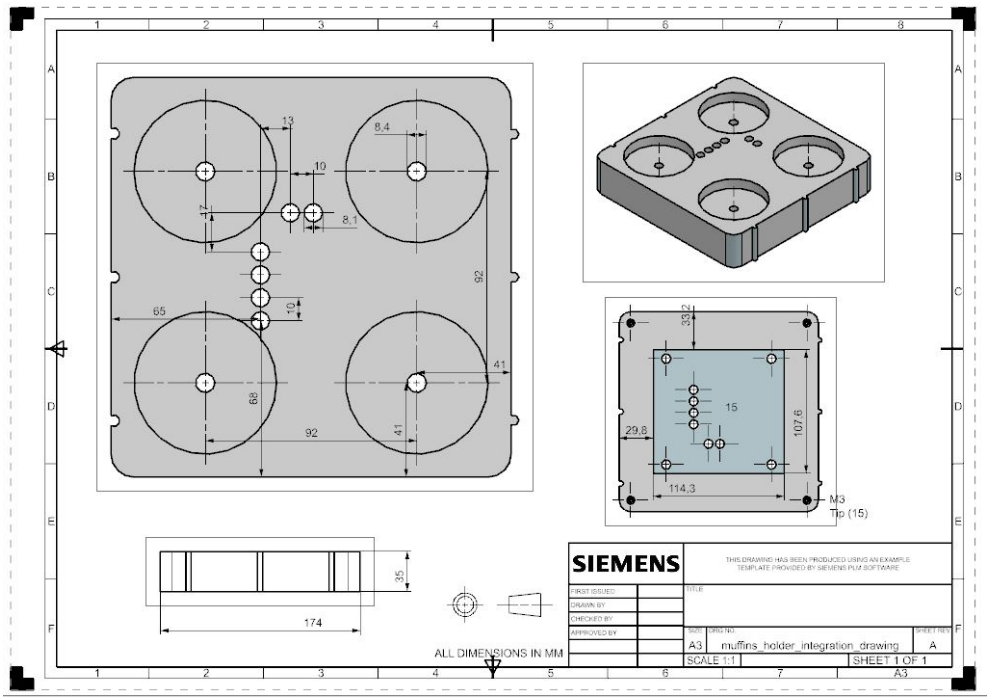
Figur A.10: Kakeholder PCB-en generert som 3D-modell i NX



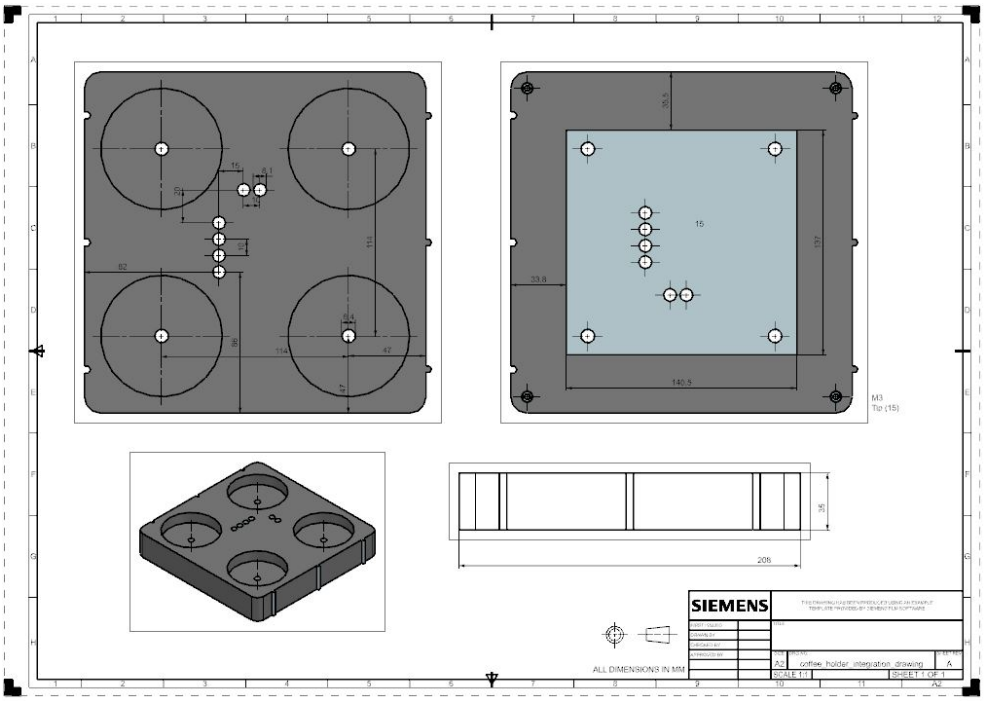
Figur A.11: Koppholder PCB-en generert som 3D-modell i NX



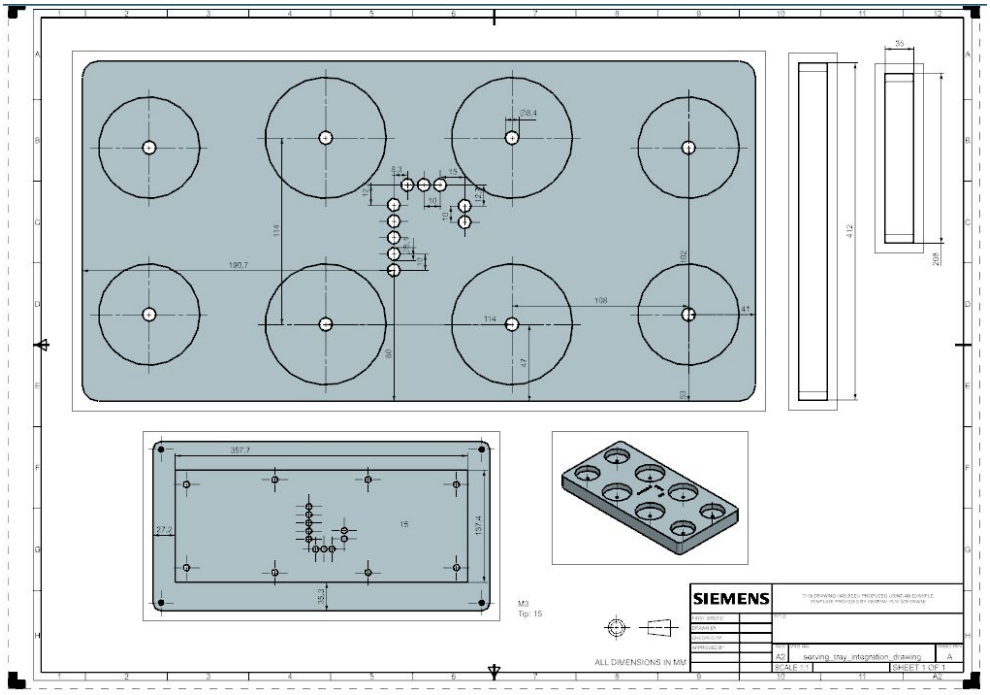
Figur A.12: Serveringsbrett PCB-en generert som 3D-modell i NX



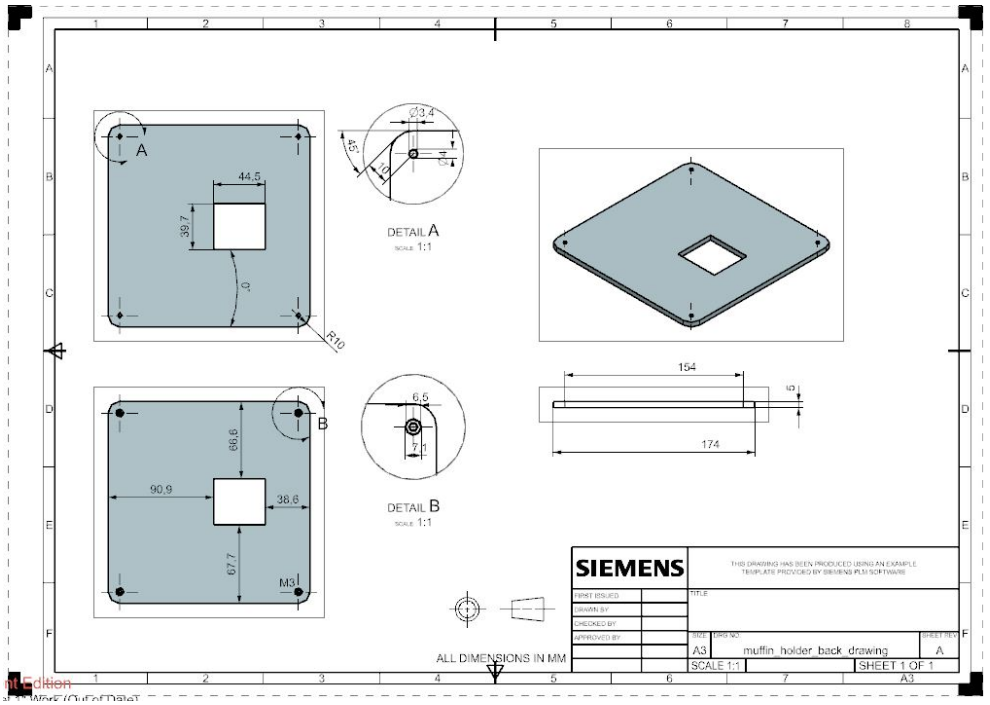
Figur A.13: Forsiden av kakeholderen



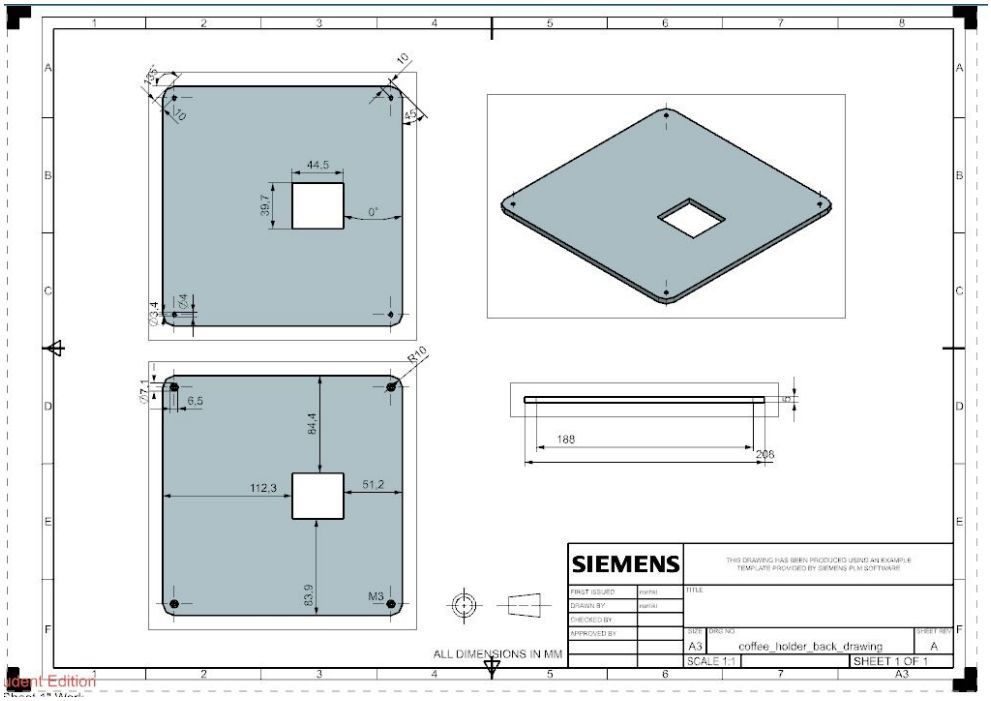
Figur A.14: Forsiden av koppholderen



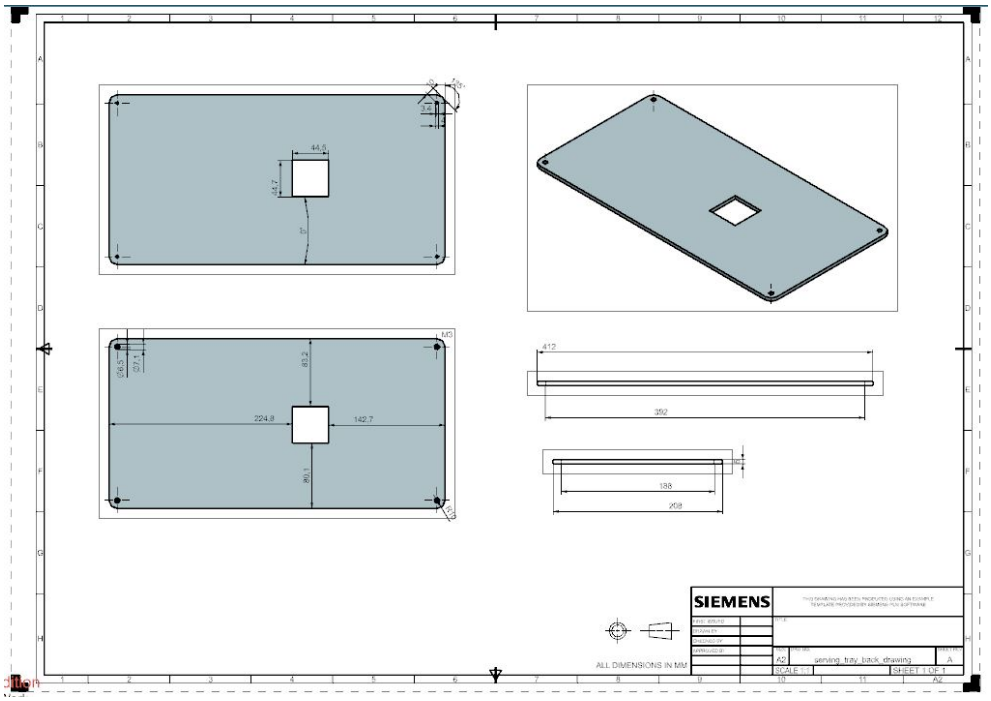
Figur A.15: Forsiden av serveringsbrettet



Figur A.16: Baksiden av kakeholderen



Figur A.17: Baksiden av koppholderen



Figur A.18: Baksiden av serveringsbrettet

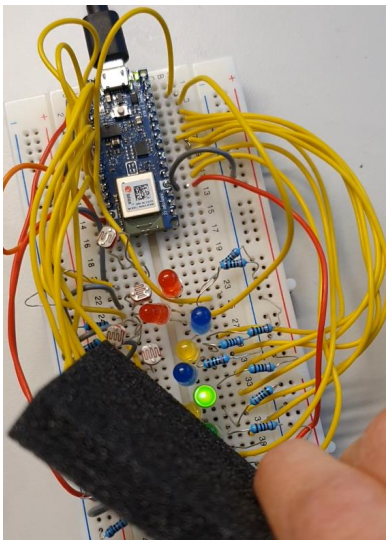
A.0.3 Oppkobling og koding før PCB-design

Bildedokumentasjon testrapport:

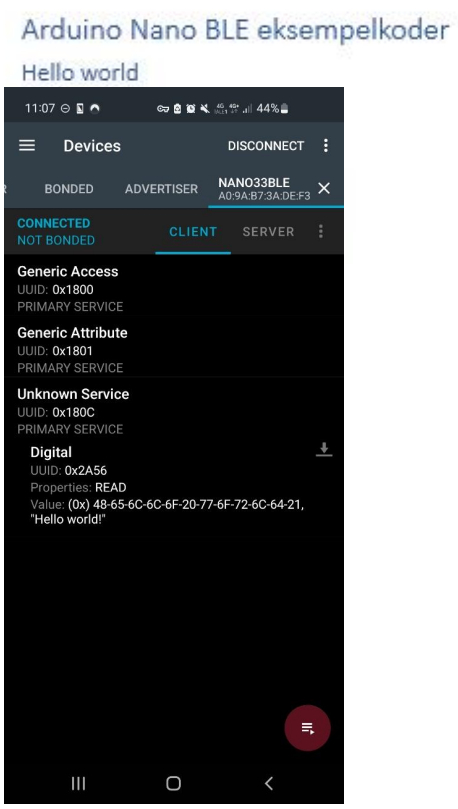
Programmering og koblinger før PCB-design

Pinnetest

LED-ene er digitale utganger, mens LDR-ene er analoge innganger. Mikrokontrolleren får spenning fra USB-kabelen mens alle pinnene får spenning fra 3V3-pinnen

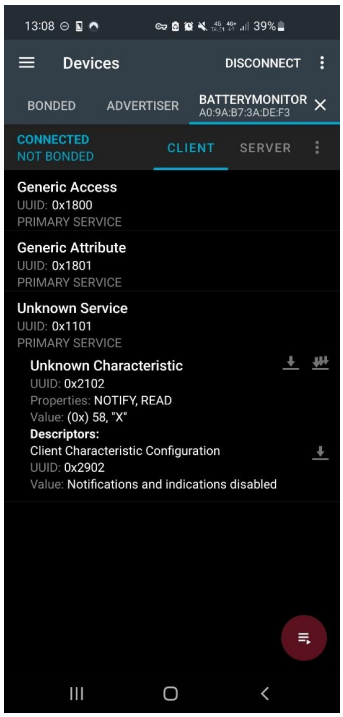


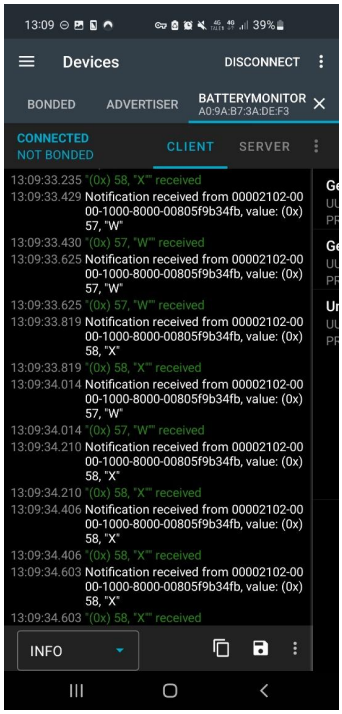
20230413_105430.mp4

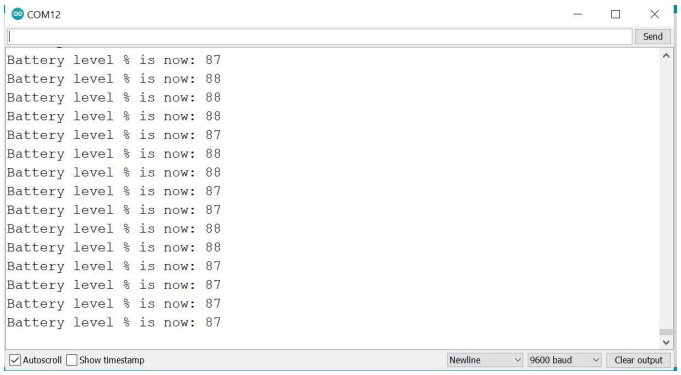


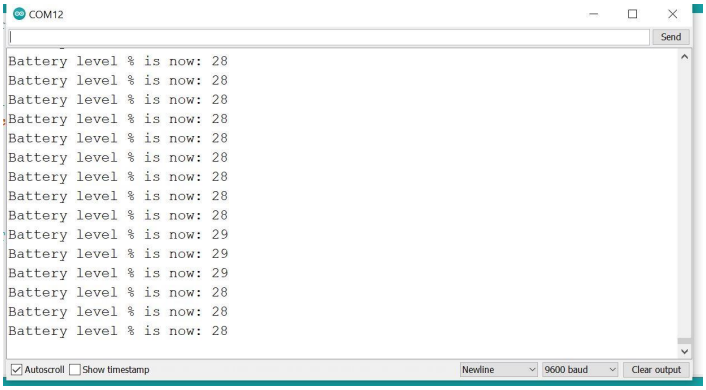
Batteritesten

(egentlig bare voltmåling av analog pinne)

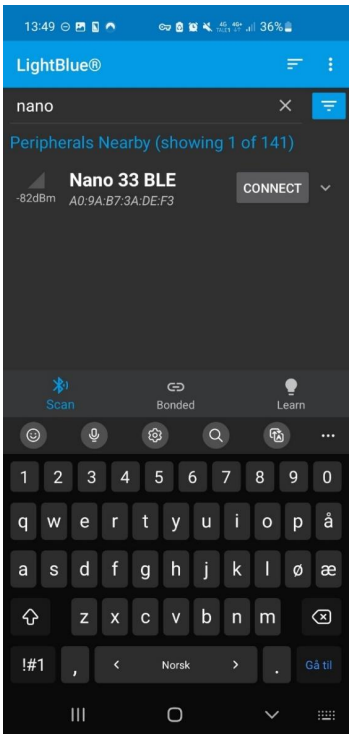


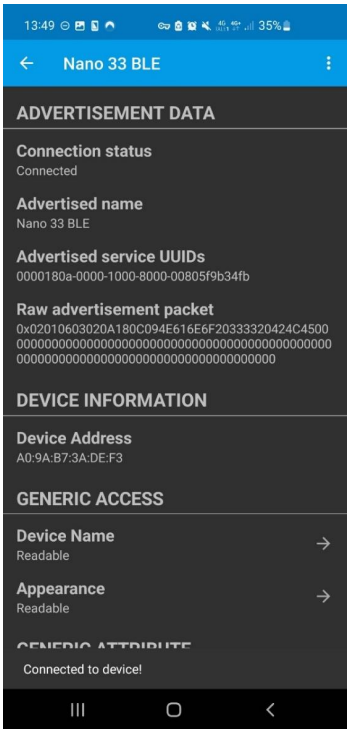


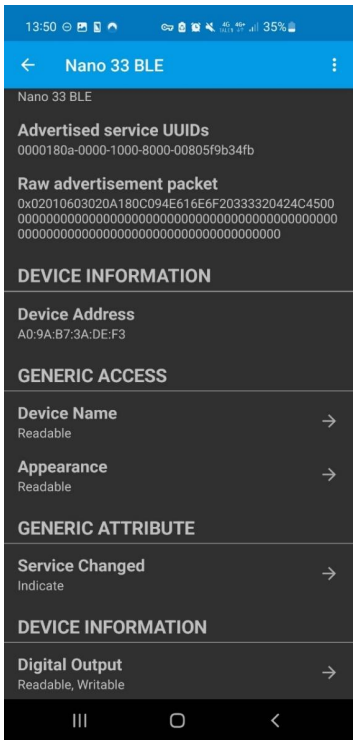


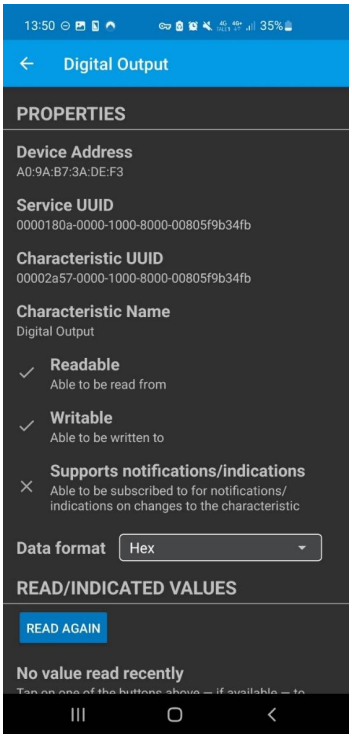


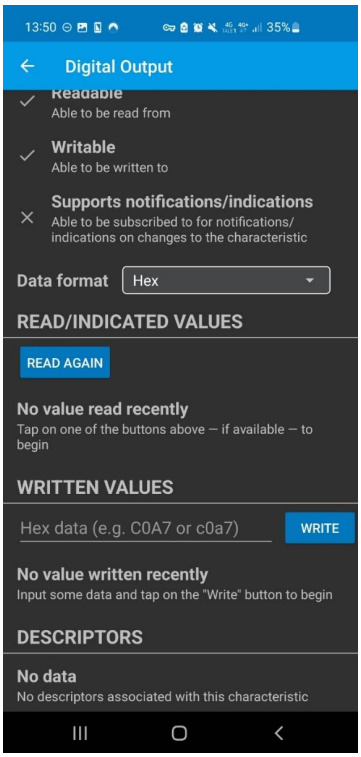
Styring av RGB LED

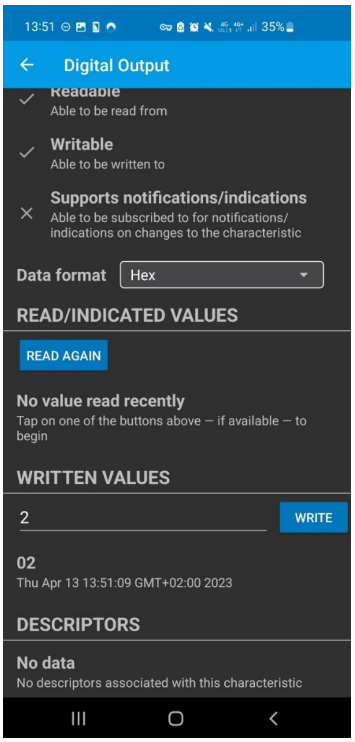


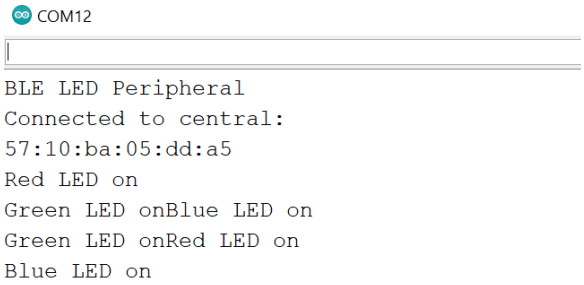


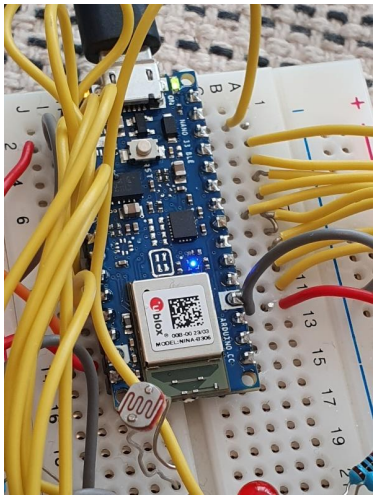


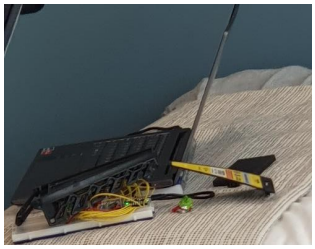


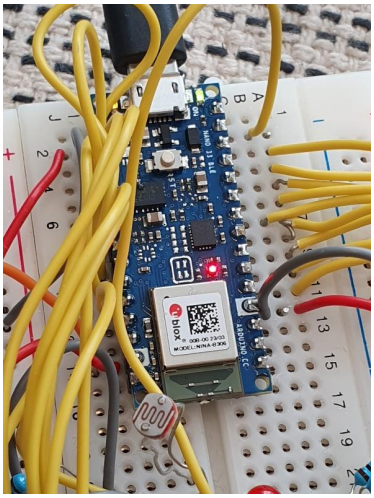

13:50
Digital Output
Readable
Able to be read from
Writable
Able to be written to
Supports notifications/indications
Able to be subscribed to for notifications/ indications on changes to the characteristic
Data format
Hex
READ/INDICATED VALUES
READ AGAIN
No value read recently
Tap on one of the buttons above — if available — to begin
WRITTEN VALUES
Hex data (e.g. C0A7 or c0a7)
WRITE
No value written recently
Input some data and tap on the "Write" button to begin
DESCRIPTORS
No data
No descriptors associated with this characteristic

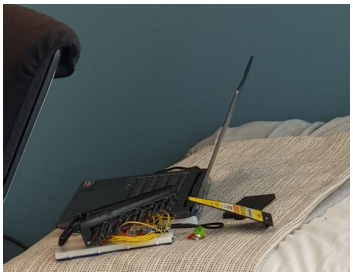

13:51
Digital Output
Readable
Able to be read from
Writable
Able to be written to
Supports notifications/indications
Able to be subscribed to for notifications/ indications on changes to the characteristic
Data format
Hex
READ/INDICATED VALUES
READ AGAIN
No value read recently
Tap on one of the buttons above — if available — to begin
WRITTEN VALUES
2
WRITE
02
Thu Apr 13 13:51:09 GMT+02:00 2023
DESCRIPTORS
No data
No descriptors associated with this characteristic

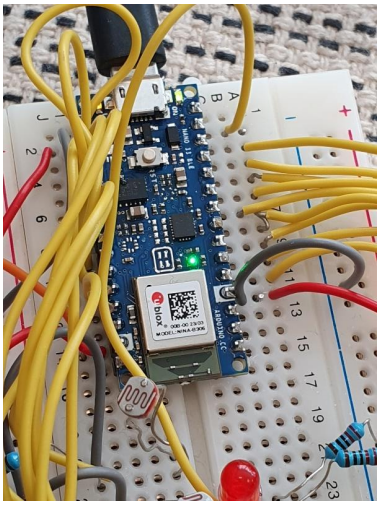

COM12
BLE LED Peripheral
Connected to central:
57:10:ba:05:dd:a5
Red LED on
Green LED onBlue LED on
Green LED onRed LED on
Blue LED on



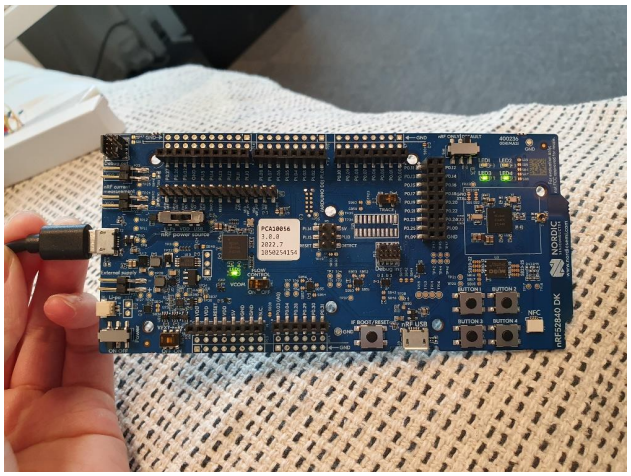


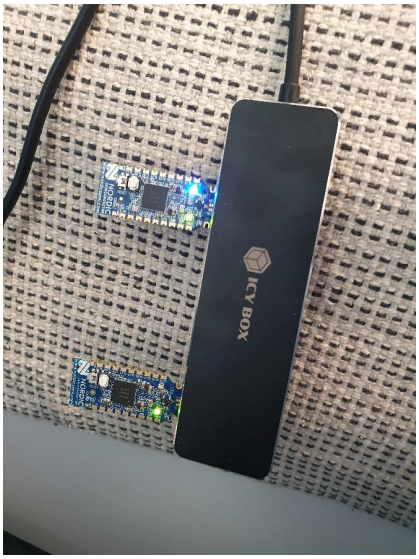






Zigbee-testen





Bluetooth-tester

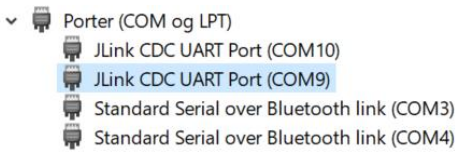
Blinkende lys

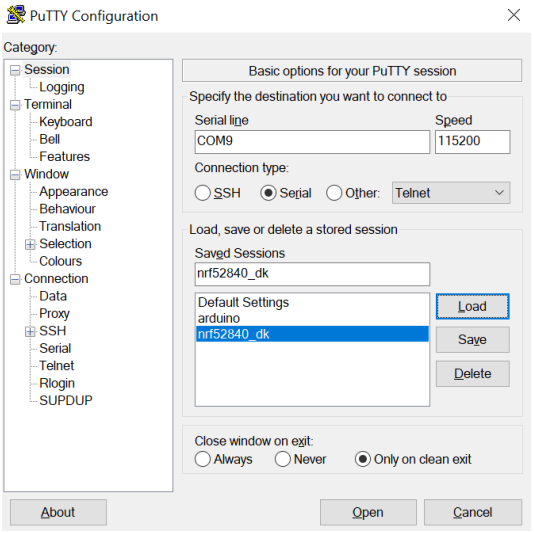
To tester med blinkende lys ble utført. Dette ble gjort med to ulike apper, disse står i parentes

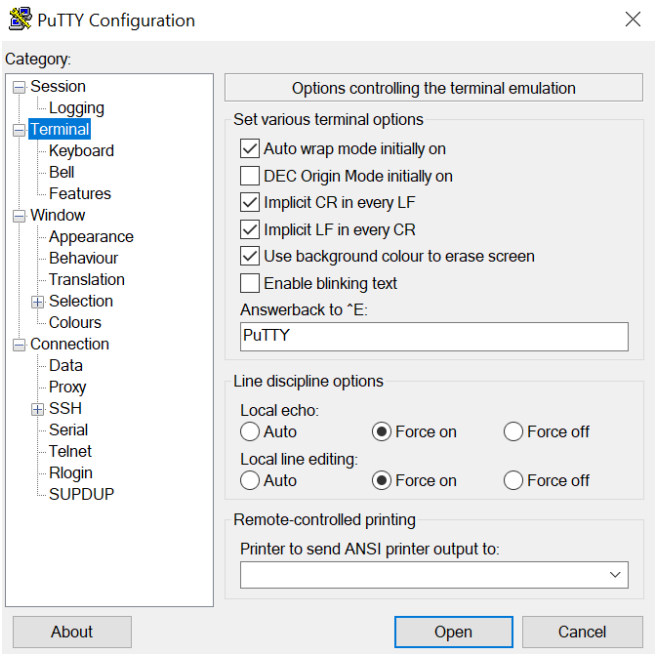
- Peripheral LBS (nRF Connect)
- ST BLE Sensor Demo (ST BLE sensor)

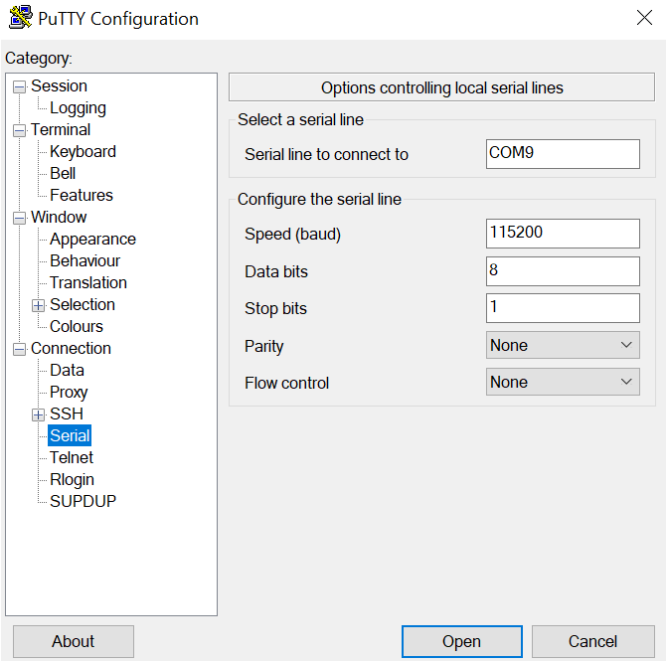
UART

(trenger ikke egen UART kabel, bare vanlig ledning mellom PC og mikrokontroller)



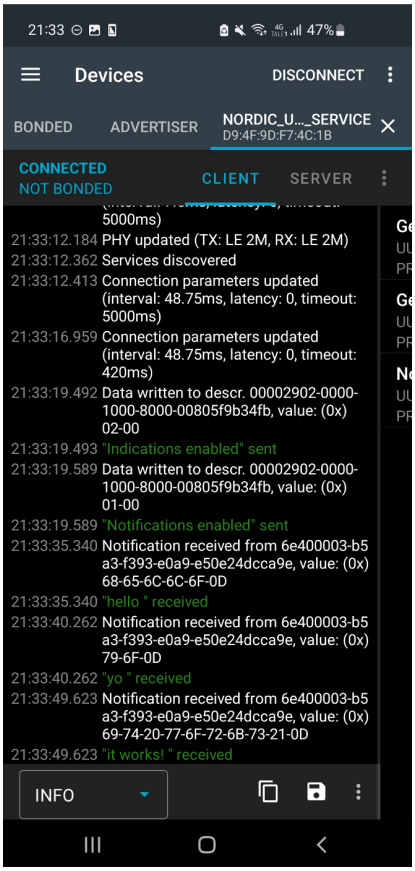




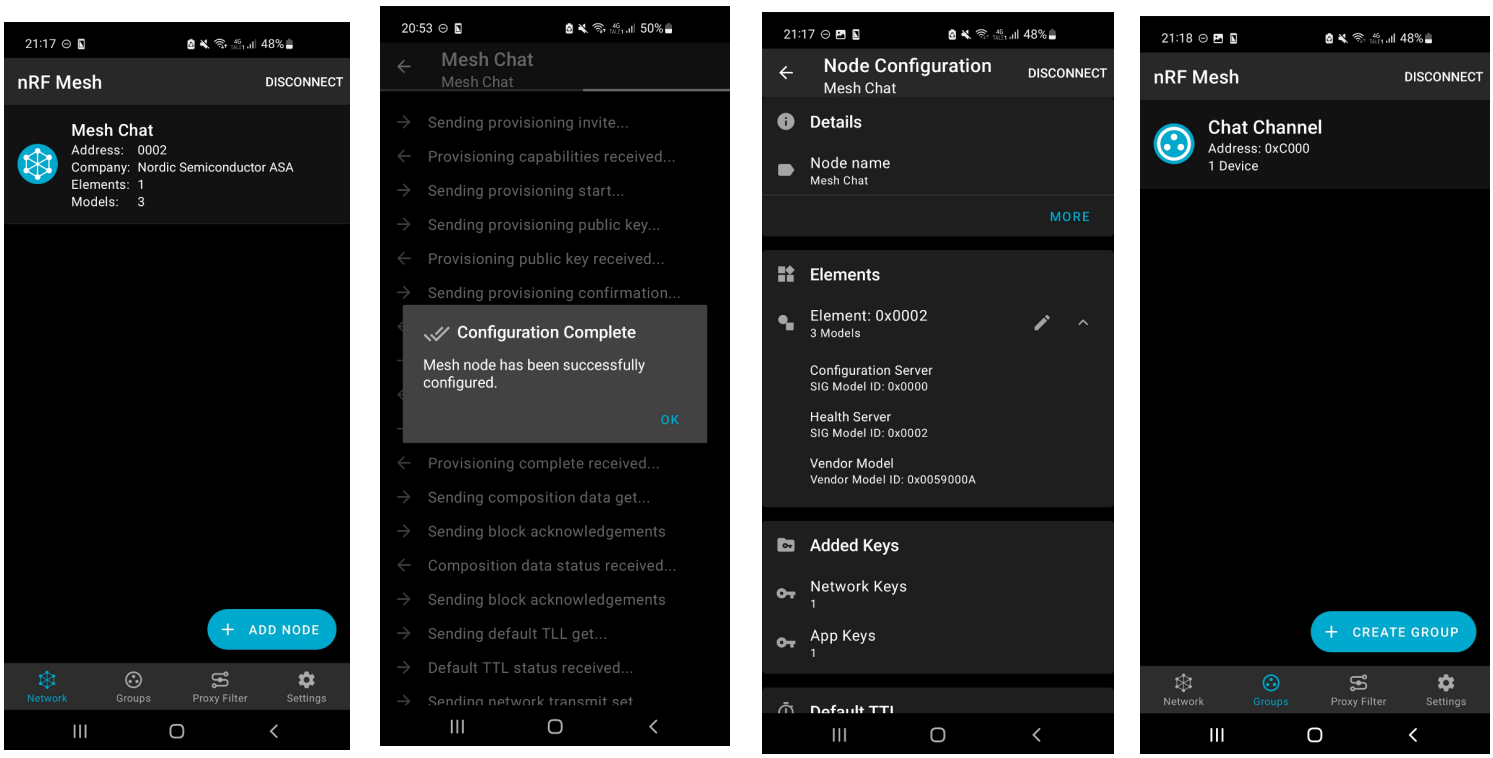


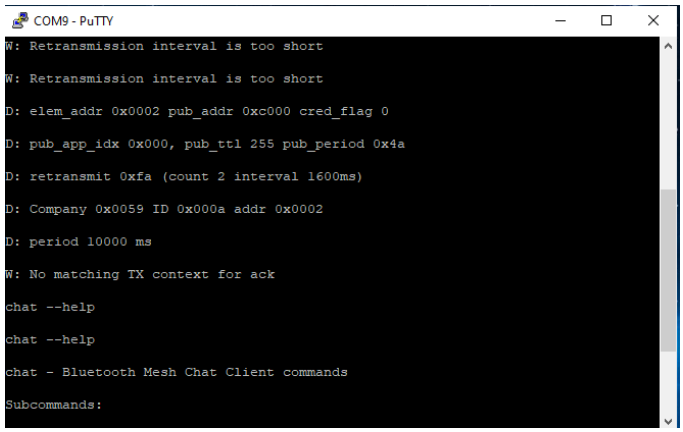
Trykk ctrl + space når terminalen åpnes

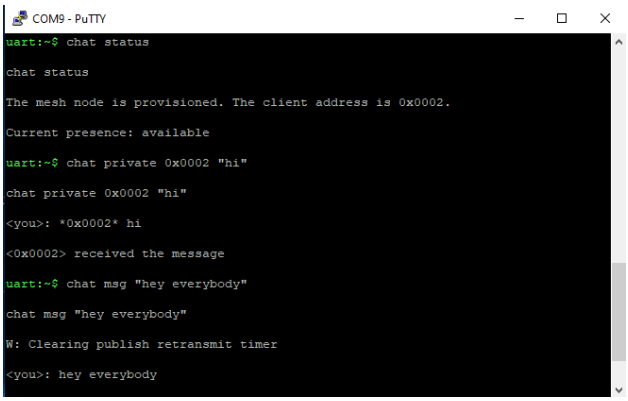

COM9 - PuTTY
*** Bo
^@
hello
yo
it works!

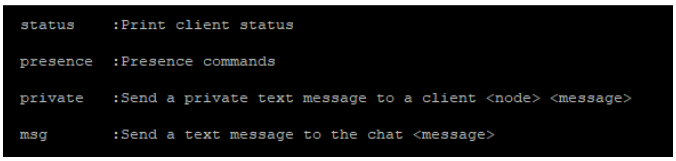

21:33
47%
Devices
DISCONNECT
BONDED
ADVERTISER
NORDIC_U..._SERVICE
D9:4F:9D:F7:4C:1B
CONNECTED
NOT BONDED
CLIENT
SERVER
5000ms)
21:33:12.184 PHY updated (TX: LE 2M, RX: LE 2M)
21:33:12.362 Services discovered
21:33:12.413 Connection parameters updated (interval: 48.75ms, latency: 0, timeout: 5000ms)
21:33:16.959 Connection parameters updated (interval: 48.75ms, latency: 0, timeout: 420ms)
21:33:19.492 Data written to descr. 00002902-0000-1000-8000-00805f9b34fb, value: (0x) 02-00
21:33:19.493 "Indications enabled" sent
21:33:19.589 Data written to descr. 00002902-0000-1000-8000-00805f9b34fb, value: (0x) 01-00
21:33:19.589 "Notifications enabled" sent
21:33:35.340 Notification received from 6e400003-b5a3-f393-e0a9-e50e24dcca9e, value: (0x) 68-65-6C-6C-6F-0D
21:33:35.340 "hello " received
21:33:40.262 Notification received from 6e400003-b5a3-f393-e0a9-e50e24dcca9e, value: (0x) 79-6F-0D
21:33:40.262 "yo " received
21:33:49.623 Notification received from 6e400003-b5a3-f393-e0a9-e50e24dcca9e, value: (0x) 69-74-20-77-6F-72-6B-73-21-0D
21:33:49.623 "it works! " received
INFO

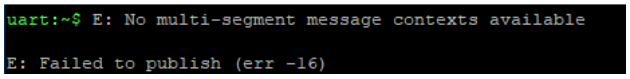
Mesh chat

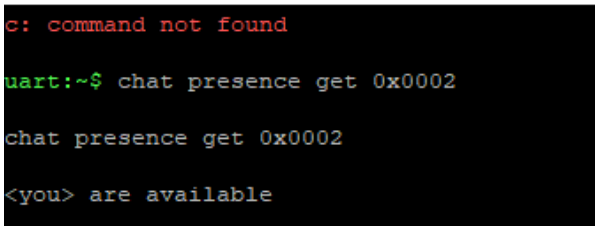

21:17
48%
nRF Mesh
DISCONNECT
Mesh Chat
Address: 0002
Company: Nordic Semiconductor ASA
Elements: 1
Models: 3
ADD NODE
Network
Groups
Proxy Filter
Settings
20:53
50%
Mesh Chat
Mesh Chat
Sending provisioning invite...
Provisioning capabilities received...
Sending provisioning start...
Sending provisioning public key...
Provisioning public key received...
Sending provisioning confirmation...
Configuration Complete
Mesh node has been successfully configured.
OK
Provisioning complete received...
Sending composition data get...
Sending block acknowledgements
Composition data status received...
Sending block acknowledgements
Sending default TLL get...
Default TTL status received...
21:17
48%
Node Configuration
Mesh Chat
DISCONNECT
Details
Node name
Mesh Chat
MORE
Elements
Element: 0x0002
3 Models
Configuration Server
SIG Model ID: 0x0000
Health Server
SIG Model ID: 0x0002
Vendor Model
Vendor Model ID: 0x0059000A
Added Keys
Network Keys
1
App Keys
1
21:18
48%
nRF Mesh
DISCONNECT
Chat Channel
Address: 0xC000
1 Device
CREATE GROUP
Network
Groups
Proxy Filter
Settings


COM9 - PuTTY
W: Retransmission interval is too short
W: Retransmission interval is too short
D: elem_addr 0x0002 pub_addr 0xc000 cred_flag 0
D: pub_app_idx 0x000, pub_ttl 255 pub_period 0x4a
D: retransmit 0xfa (count 2 interval 1600ms)
D: Company 0x0059 ID 0x000a addr 0x0002
D: period 10000 ms
W: No matching TX context for ack
chat --help
chat --help
chat - Bluetooth Mesh Chat Client commands
Subcommands:


COM9 - PuTTY
uart:~$ chat status
chat status
The mesh node is provisioned. The client address is 0x0002.
Current presence: available
uart:~$ chat private 0x0002 "hi"
chat private 0x0002 "hi"
<you>: *0x0002* hi
<0x0002> received the message
uart:~$ chat msg "hey everybody"
chat msg "hey everybody"
W: Clearing publish retransmit timer
<you>: hey everybody


status :Print client status
presence :Presence commands
private :Send a private text message to a client <node> <message>
msg :Send a text message to the chat <message>


uart:~$ E: No multi-segment message contexts available
E: Failed to publish (err -16)


c: command not found
uart:~$ chat presence get 0x0002
chat presence get 0x0002
<you> are available

Førdemo-program

(dette programmet er mer et NACHI-gripeprogram bare med 8 objekter, så en kombinasjon av de to programmene)

pcb_nano_eight_ble

```
#include <ArduinoBLE.h>

//om man skal fylle inn hele koden eller kun de 16 som endres?
//definisjon av BLE-service og -karakteristikker
BLEService objectManager("145A");
BLEStringCharacteristic nextObject("5B8F", BLERead | BLENotify, 7); //7 er størrelsen på strengen


//LED-indikatorer for fysisk debugging
//disse trenger kun være 0 eller 1 og er digitale utganger
#define ledStatusPin1 2
#define ledStatusPin2 3
#define ledStatusPin3 4
#define ledStatusPin4 5
#define ledStatusPin5 6
#define ledStatusPin6 7
#define ledStatusPin7 8
#define ledStatusPin8 9
#define noObjectErrorPin 10
#define wirelessErrorPin 11

//objekter dvs. kaffe eller kaker
//disse er analoge innganger, for kodet tilpasning til miljøet
#define lightObject1 A0
#define lightObject2 A1
#define lightObject3 A2
#define lightObject4 A3
#define lightObject5 A4
#define lightObject6 A5
#define lightObject7 A6
#define lightObject8 A7

int ledStatus1;
int ledStatus2;
int ledStatus3;
```

```
int ledStatus4;
int ledStatus5;
int ledStatus6;
int ledStatus7;
int ledStatus8;
int noObjectError;
int wirelessError;

void setup() {

  //serial monitor, UART
  Serial.begin(9600);
  while (!Serial);

  //retning på pinner
  pinMode(ledStatusPin1, OUTPUT);
  pinMode(ledStatusPin2, OUTPUT);
  pinMode(ledStatusPin3, OUTPUT);
  pinMode(ledStatusPin4, OUTPUT);
  pinMode(ledStatusPin5, OUTPUT);
  pinMode(ledStatusPin6, OUTPUT);
  pinMode(ledStatusPin7, OUTPUT);
  pinMode(ledStatusPin8, OUTPUT);
  pinMode(noObjectErrorPin, OUTPUT);
  pinMode(wirelessErrorPin, OUTPUT);

  pinMode(lightObject1, INPUT);
  pinMode(lightObject2, INPUT);
  pinMode(lightObject3, INPUT);
  pinMode(lightObject4, INPUT);
  pinMode(lightObject5, INPUT);
  pinMode(lightObject6, INPUT);
  pinMode(lightObject7, INPUT);
  pinMode(lightObject8, INPUT);

  //initialverdier for statuskode for LEDs
  ledStatus1 = 0;
  ledStatus2 = 0;
  ledStatus3 = 0;
  ledStatus4 = 0;
  ledStatus5 = 0;
  ledStatus6 = 0;
  ledStatus7 = 0;
  ledStatus8 = 0;
  noObjectError = 0;
  wirelessError = 0;

  //oppstart av BLE
  if (!BLE.begin()) {
    Serial.println("Starting Bluetooth Low Energy failed");
    while (1);
  }

  BLE.setLocalName("Serveringsbrett");
  BLE.setAdvertisedService(objectManager);
  objectManager.addCharacteristic(nextObject);
  BLE.addService(objectManager);
  //trenger jeg initialverdi eller verdi som betyr ingenting
  BLE.advertise();
  Serial.println("Serving tray pheripheral");
```

```
126   //skru av det man ikke bruker for å spare strøm
127   //sjekke om dette ikke forstyrrer andre ting og har virkning
128   //kommentere ut alt av print og unødvendige LEDs i det sammensatte systemet
129   digitalWrite(LED_PWR, LOW);
130   digitalWrite(PIN_ENABLE_SENSORS_3V3, LOW);
131   digitalWrite(PIN_ENABLE_I2C_PULLUP, LOW);
132
133 }
134
135
136 void loop() {
137
138   //verdien som tilsvarer at det er lyst/mørkt altså om det er kaffe eller kake over lyssensoren eller ikke
139   int darkValue = 500;
140
141   BLEDevice central = BLE.central();
```

```
143   if (central) {
144     Serial.print("Connected to central:");
145     Serial.println(central.address());
146     digitalWrite(LED_BUILTIN, HIGH);
147
148     while (central.connected()) {
149       int lightObjectValue1 = analogRead(lightObject1);
150       int lightObjectValue2 = analogRead(lightObject2);
151       int lightObjectValue3 = analogRead(lightObject3);
152       int lightObjectValue4 = analogRead(lightObject4);
153       int lightObjectValue5 = analogRead(lightObject5);
154       int lightObjectValue6 = analogRead(lightObject6);
155       int lightObjectValue7 = analogRead(lightObject7);
156       int lightObjectValue8 = analogRead(lightObject8);
```

```
158     Serial.println(lightObjectValue5);
159
160
161     /*alle LED-ene kobles aktivt høye, status-kodene følger det samme
162       når LDR-verdien er høye er det lyst dvs. at det ikke er noe objekt til stede. Da skal LED-en være av
163       når LDR-verdien er lav er det mørkt dvs. at det er et objekt til stede. Da skal LED-en lyse*/
164
165
166     //objekt 1
167     if (lightObjectValue1 <= darkValue) {
168       digitalWrite(ledStatusPin1, HIGH);
169       ledStatus1 = 1;
170     }


172     else if (lightObjectValue1 > darkValue) {
173       digitalWrite(ledStatusPin1, LOW);
174       ledStatus1 = 0;
175     }
176
177
178     //objekt 2
179     if (lightObjectValue2 <= darkValue) {
180       digitalWrite(ledStatusPin2, HIGH);
181       Serial.println("here");
182       ledStatus2 = 1;
183     }
184
185     else if (lightObjectValue2 > darkValue) {
186       digitalWrite(ledStatusPin2, LOW);
187       ledStatus2 = 0;
```

```
188     }
189
190
191     //objekt 3
192     if (lightObjectValue3 <= darkValue) {
193       digitalWrite(ledStatusPin3, HIGH);
194       ledStatus3 = 1;
195     }
196
197     else if (lightObjectValue3 > darkValue) {
198       digitalWrite(ledStatusPin3, LOW);
199       ledStatus3 = 0;
200     }
```

```
203     //objekt 4
204     if (lightObjectValue4 <= darkValue) {
205       digitalWrite(ledStatusPin4, HIGH);
206       ledStatus4 = 1;
207     }
208
209     else if (lightObjectValue4 > darkValue) {
210       digitalWrite(ledStatusPin4, LOW);
211       ledStatus4 = 0;
212     }
213
214     //objekt 5
215     if (lightObjectValue5 <= darkValue) {
216       digitalWrite(ledStatusPin5, HIGH);
217       ledStatus5 = 1;
218     }
```

```
220     else if (lightObjectValue5 > darkValue) {
221       digitalWrite(ledStatusPin5, LOW);
222       ledStatus5 = 0;
223     }
224
225
226     //objekt 6
227     if (lightObjectValue6 <= darkValue) {
228       digitalWrite(ledStatusPin6, HIGH);
229       ledStatus6 = 1;
230     }
231
232     else if (lightObjectValue6 > darkValue) {
233       digitalWrite(ledStatusPin6, LOW);
234       ledStatus6 = 0;
235     }
```

```
238     //objekt 7
239     if (lightObjectValue7 <= darkValue) {
240       digitalWrite(ledStatusPin7, HIGH);
241       ledStatus7 = 1;
242     }
243
244     else if (lightObjectValue7 > darkValue) {
245       digitalWrite(ledStatusPin7, LOW);
246       ledStatus7 = 0;
247     }
248
249
250     //objekt 8
251     if (lightObjectValue8 <= darkValue) {
252       digitalWrite(ledStatusPin8, HIGH);
253       ledStatus8 = 1;
254     }
```

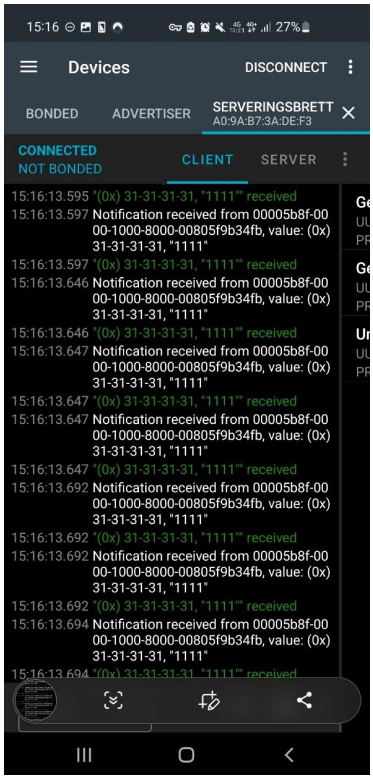
```
256       else if (lightObjectValue8 > darkValue) {
257         digitalWrite(ledStatusPin8, LOW);
258         ledStatus8 = 0;
259       }
260
261
262       //se for meg alle mulighetene koppene kan stå
263       //tror ikke jeg trenger å ta med alle kombinasjonene fordi jeg følger rekkefølgen med if-strukturen min
264
265       /*if-struktur 1 er for at ledene skal vise rett, mens if-struktur
266       2 er for å velge hvilket av objektene som det skal sendes info om*/
267
268
269       if (ledStatus1 == 1) {
270         Serial.println("Pick up object 1");

273         nextObject.writeValue("0000");
274         //og sende bluetooth signal 1, evt. reconnect
275         //vente noen ms, nei den vil sjekke om koppen er der automatisk
276       }
277
278       else if (ledStatus2 == 1) {
279         Serial.println("Pick up object 2");
280
281         nextObject.writeValue("0001");
282         //og sende bluetooth signal 2, evt reconnect
283         //vente noen ms, nei den vil sjekke om koppen er der automatisk
284       }
285
286       else if (ledStatus3 == 1) {
287         Serial.println("Pick up object 3");
288         nextObject.writeValue("0010");
289         //og sende bluetooth signal 3, evt reconnect
290         //vente noen ms, nei den vil sjekke om koppen er der automatisk
```

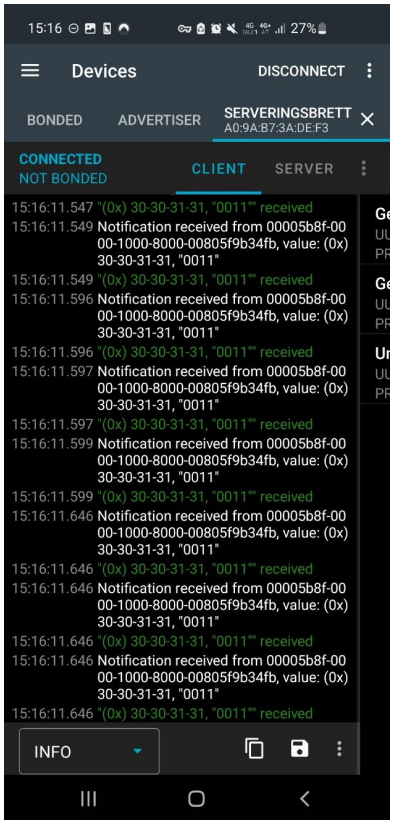
```
292       }
293
294       else if (ledStatus4 == 1) {
295         Serial.println("Pick up object 4");
296         nextObject.writeValue("0011");
297         //og sende bluetooth signal 4, evt. reconnect
298         //vente noen ms, nei den vil sjekke om koppen er der automatisk
299       }
300
301       else if (ledStatus5 == 1) {
302         Serial.println("Pick up object 5");
303         nextObject.writeValue("0100");
304         //og sende bluetooth signal 5, evt. reconnect
305         //vente noen ms, nei den vil sjekke om koppen er der automatisk
306
307       }
309       else if (ledStatus6 == 1) {
310         Serial.println("Pick up object 6");
311         nextObject.writeValue("0101");
312         //og sende bluetooth signal 6, evt. reconnect
313         //vente noen ms, nei den vil sjekke om koppen er der automatisk
314
315       }
316
317       else if (ledStatus7 == 1) {
318         Serial.println("Pick up object 7");
319         nextObject.writeValue("0110");
320         //og sende bluetooth signal 7, evt. reconnect
321         //vente noen ms, nei den vil sjekke om koppen er der automatisk
322
323       }
324
```

```
325       else if (ledStatus8 == 1) {
326         Serial.println("Pick up object 8");
327         //og sende bluetooth signal 8, evt. reconnect
328         nextObject.writeValue("0111");
329         //vente noen ms, nei den vil sjekke om koppen er der automatisk
330
331       }
332
333       else { //det vil si ikke noe kaffe/kaker igjen
334         digitalWrite(noObjectError, HIGH);
335         nextObject.writeValue("1111");
336       }
337     }
338   }
339
340   digitalWrite(LED_BUILTIN, LOW);
341   Serial.print("Disconnected from central: ");
342    Serial.println(central.address());
343
344 }
```

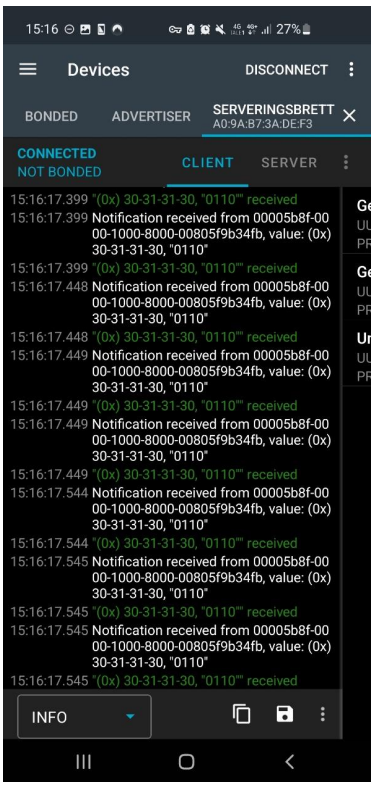

15:16
27%
Devices
DISCONNECT
BONDED
ADVERTISER
SERVERINGSBRETT
A0:9A:B7:3A:DE:F3
CONNECTED
NOT BONDED
CLIENT
SERVER
15:16:26.953 "(0x) 30-30-31-30, "0010"" received
15:16:26.953 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-30, "0010"
15:16:26.953 "(0x) 30-30-31-30, "0010"" received
15:16:27.002 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-30, "0010"
15:16:27.002 "(0x) 30-30-31-30, "0010"" received
15:16:27.004 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-30, "0010"
15:16:27.004 "(0x) 30-30-31-30, "0010"" received
15:16:27.005 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-30, "0010"
15:16:27.005 "(0x) 30-30-31-30, "0010"" received
15:16:27.050 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-30, "0010"
15:16:27.051 "(0x) 30-30-31-30, "0010"" received
15:16:27.051 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-30, "0010"
15:16:27.051 "(0x) 30-30-31-30, "0010"" received
15:16:27.053 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-30, "0010"


15:16
27%
Devices
DISCONNECT
BONDED
ADVERTISER
SERVERINGSBRETT
A0:9A:B7:3A:DE:F3
CONNECTED
NOT BONDED
CLIENT
SERVER
15:16:22.858 "(0x) 30-30-30-30, "0000"" received
15:16:22.859 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-30-30, "0000"
15:16:22.859 "(0x) 30-30-30-30, "0000"" received
15:16:22.905 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-30-30, "0000"
15:16:22.905 "(0x) 30-30-30-30, "0000"" received
15:16:22.907 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-30-30, "0000"
15:16:22.907 "(0x) 30-30-30-30, "0000"" received
15:16:22.907 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-30-30, "0000"
15:16:22.907 "(0x) 30-30-30-30, "0000"" received
15:16:22.954 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-30-30, "0000"
15:16:22.954 "(0x) 30-30-30-30, "0000"" received
15:16:22.955 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-30-30, "0000"
15:16:22.955 "(0x) 30-30-30-30, "0000"" received
15:16:22.955 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-30-30, "0000"
15:16:22.955 "(0x) 30-30-30-30, "0000"" received
INFO

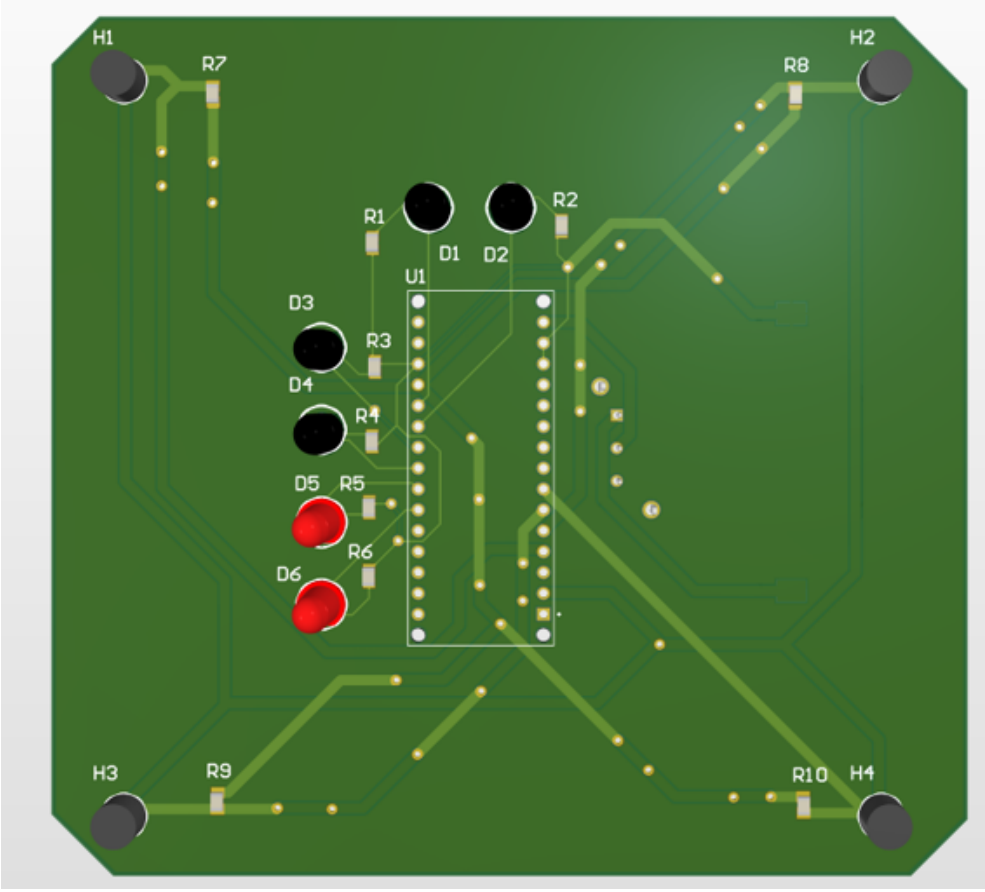

15:16
27%
Devices
DISCONNECT
BONDED
ADVERTISER
SERVERINGSBRETT
A0:9A:B7:3A:DE:F3
CONNECTED
NOT BONDED
CLIENT
SERVER
15:16:13.595 "(0x) 31-31-31-31, "1111"" received
15:16:13.597 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 31-31-31-31, "1111"
15:16:13.597 "(0x) 31-31-31-31, "1111"" received
15:16:13.646 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 31-31-31-31, "1111"
15:16:13.646 "(0x) 31-31-31-31, "1111"" received
15:16:13.647 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 31-31-31-31, "1111"
15:16:13.647 "(0x) 31-31-31-31, "1111"" received
15:16:13.647 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 31-31-31-31, "1111"
15:16:13.647 "(0x) 31-31-31-31, "1111"" received
15:16:13.692 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 31-31-31-31, "1111"
15:16:13.692 "(0x) 31-31-31-31, "1111"" received
15:16:13.692 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 31-31-31-31, "1111"
15:16:13.692 "(0x) 31-31-31-31, "1111"" received
15:16:13.694 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 31-31-31-31, "1111"


15:16
27%
Devices
DISCONNECT
BONDED
ADVERTISER
SERVERINGSBRETT
A0:9A:B7:3A:DE:F3
CONNECTED
NOT BONDED
CLIENT
SERVER
15:16:11.547 "(0x) 30-30-31-31, "0011"" received
15:16:11.549 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-31, "0011"
15:16:11.549 "(0x) 30-30-31-31, "0011"" received
15:16:11.596 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-31, "0011"
15:16:11.596 "(0x) 30-30-31-31, "0011"" received
15:16:11.597 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-31, "0011"
15:16:11.597 "(0x) 30-30-31-31, "0011"" received
15:16:11.599 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-31, "0011"
15:16:11.599 "(0x) 30-30-31-31, "0011"" received
15:16:11.646 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-31, "0011"
15:16:11.646 "(0x) 30-30-31-31, "0011"" received
15:16:11.646 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-31, "0011"
15:16:11.646 "(0x) 30-30-31-31, "0011"" received
15:16:11.646 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-31, "0011"
15:16:11.646 "(0x) 30-30-31-31, "0011"" received
INFO

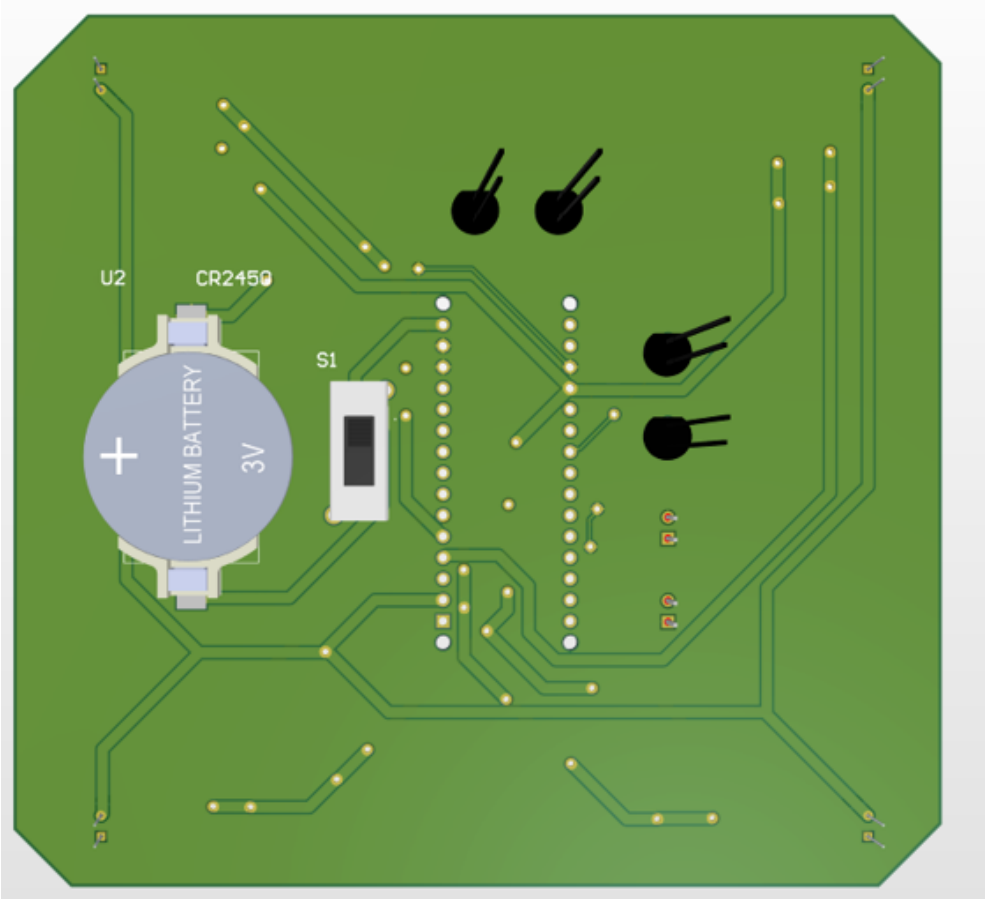

15:16
27%
Devices
DISCONNECT
BONDED
ADVERTISER
SERVERINGSBRETT
A0:9A:B7:3A:DE:F3
CONNECTED
NOT BONDED
CLIENT
SERVER
15:16:06.185 "(0x) 30-30-31-30, "0010"" received
15:16:06.188 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-30, "0010"
15:16:06.188 "(0x) 30-30-31-30, "0010"" received
15:16:06.234 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-30, "0010"
15:16:06.234 "(0x) 30-30-31-30, "0010"" received
15:16:06.234 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-30, "0010"
15:16:06.234 "(0x) 30-30-31-30, "0010"" received
15:16:06.234 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-30, "0010"
15:16:06.234 "(0x) 30-30-31-30, "0010"" received
15:16:06.282 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-30, "0010"
15:16:06.282 "(0x) 30-30-31-30, "0010"" received
15:16:06.282 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-30, "0010"
15:16:06.282 "(0x) 30-30-31-30, "0010"" received
15:16:06.283 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-30-31-30, "0010"
15:16:06.283 "(0x) 30-30-31-30, "0010"" received
INFO


15:16
27%
Devices
DISCONNECT
BONDED
ADVERTISER
SERVERINGSBRETT
A0:9A:B7:3A:DE:F3
CONNECTED
NOT BONDED
CLIENT
SERVER
15:16:17.399 "(0x) 30-31-31-30, "0110"" received
15:16:17.399 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-31-31-30, "0110"
15:16:17.399 "(0x) 30-31-31-30, "0110"" received
15:16:17.448 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-31-31-30, "0110"
15:16:17.448 "(0x) 30-31-31-30, "0110"" received
15:16:17.449 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-31-31-30, "0110"
15:16:17.449 "(0x) 30-31-31-30, "0110"" received
15:16:17.449 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-31-31-30, "0110"
15:16:17.449 "(0x) 30-31-31-30, "0110"" received
15:16:17.544 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-31-31-30, "0110"
15:16:17.544 "(0x) 30-31-31-30, "0110"" received
15:16:17.545 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-31-31-30, "0110"
15:16:17.545 "(0x) 30-31-31-30, "0110"" received
15:16:17.545 Notification received from 00005b8f-0000-1000-8000-00805f9b34fb, value: (0x) 30-31-31-30, "0110"
15:16:17.545 "(0x) 30-31-31-30, "0110"" received
INFO

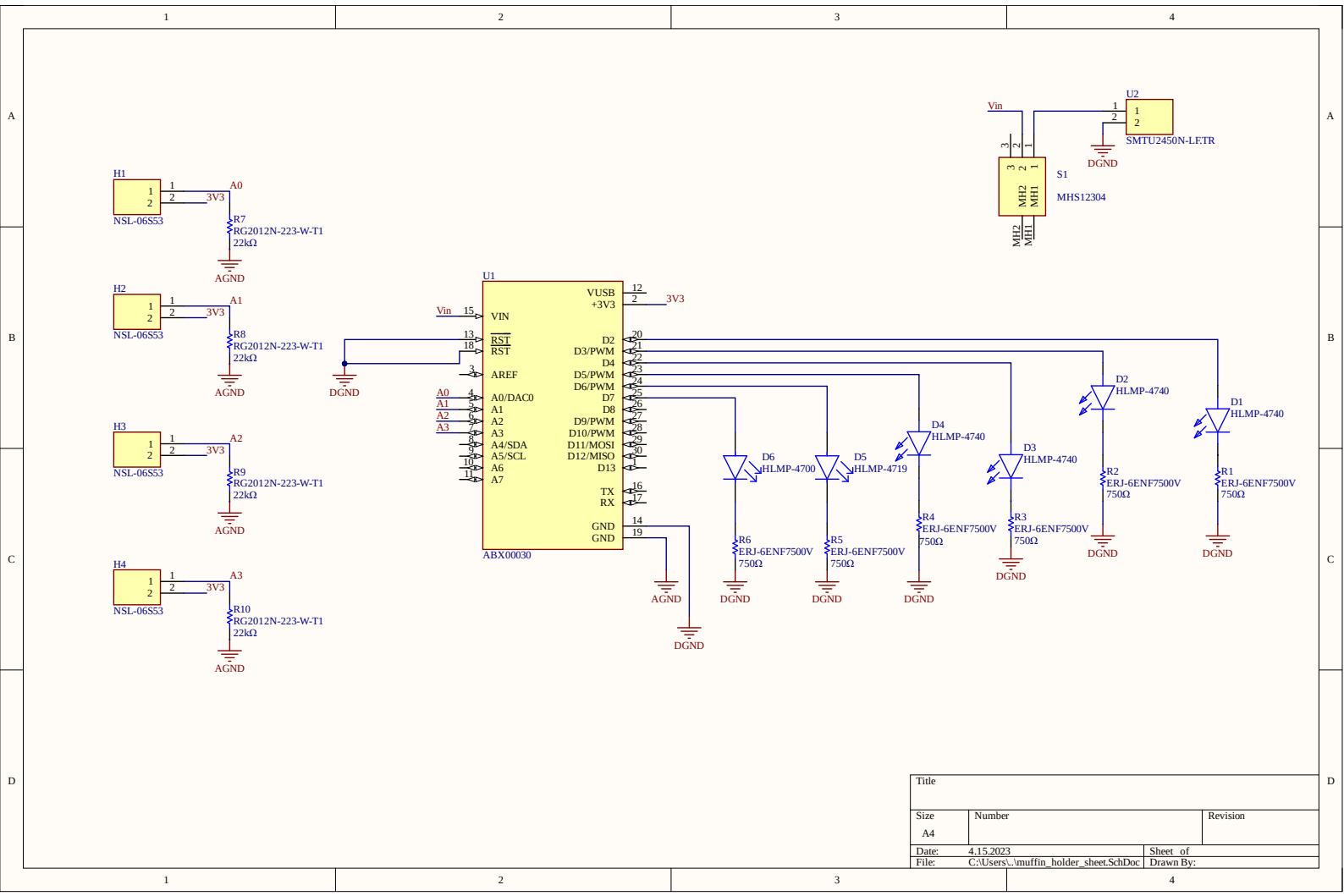
A.0.4 PCB-designfiler

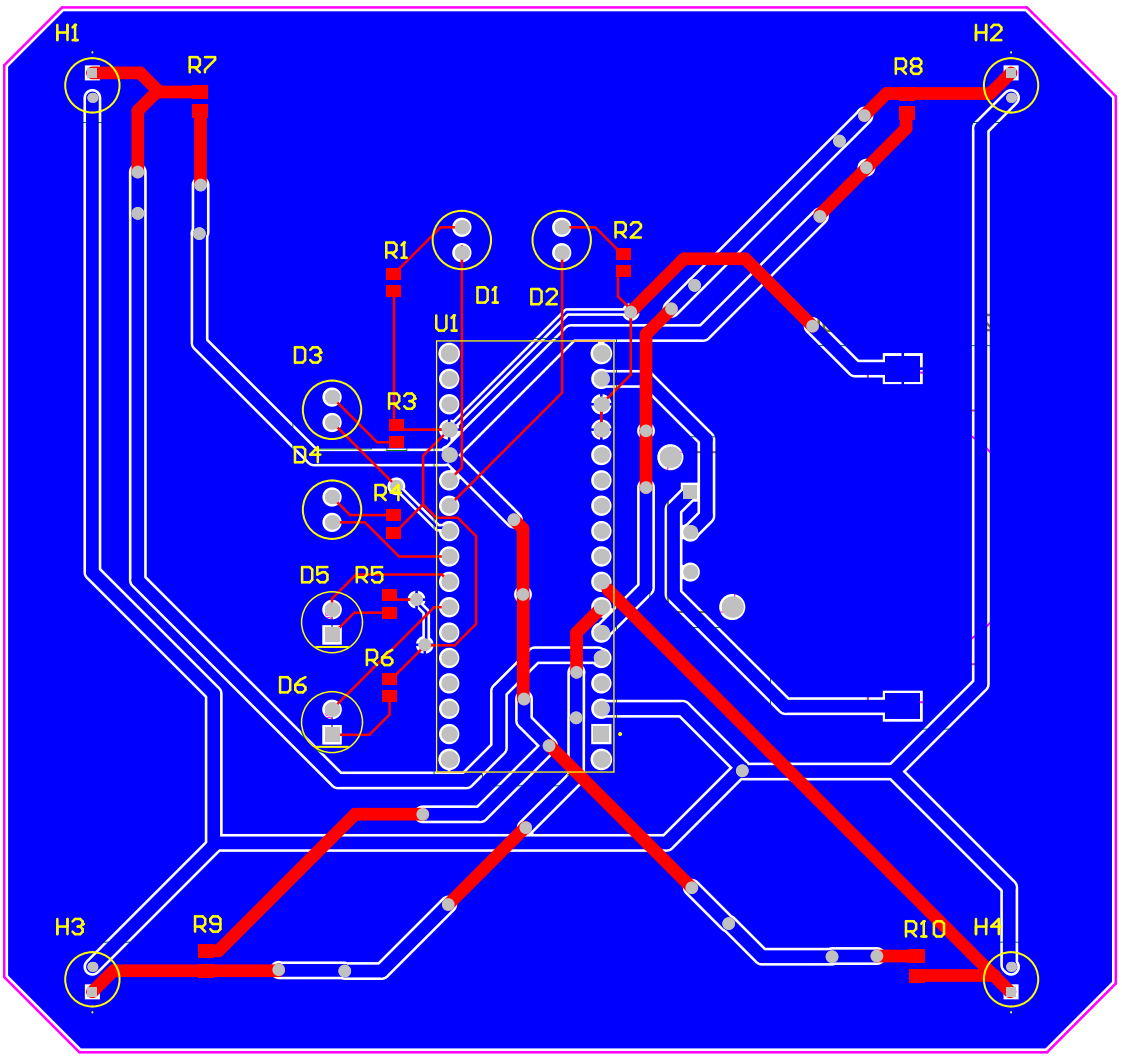


Figur A.19: Kakeholder PCB-ens forside i 3D



Figur A.20: Kakeholder PCB-ens bakside i 3D


H1
NSL-06S53
A0
3V3
R7
RG2012N-223-W-T1
22kΩ
AGND
H2
NSL-06S53
A1
3V3
R8
RG2012N-223-W-T1
22kΩ
AGND
H3
NSL-06S53
A2
3V3
R9
RG2012N-223-W-T1
22kΩ
AGND
H4
NSL-06S53
A3
3V3
R10
RG2012N-223-W-T1
22kΩ
AGND
U1
ABX00030
VIN
RST
AREF
A0/DAC0
A1
A2
A3
A4/SDA
A5/SCL
A6
A7
VUSB
+3V3
D2
D3/PWM
D4
D5/PWM
D6/PWM
D7
D8
D9/PWM
D10/PWM
D11/MOSI
D12/MISO
D13
TX
RX
GND
GND
Vin
3V3
DGND
AGND
U2
SMTU2450N-LF.TR
S1
MHS12304
MH2
MH1
D6
HLMP-4700
D5
HLMP-4719
D4
HLMP-4740
D3
HLMP-4740
D2
HLMP-4740
D1
HLMP-4740
R6
ERJ-6ENF7500V
750Ω
R5
ERJ-6ENF7500V
750Ω
R4
ERJ-6ENF7500V
750Ω
R3
ERJ-6ENF7500V
750Ω
R2
ERJ-6ENF7500V
750Ω
R1
ERJ-6ENF7500V
750Ω
Title
Size
A4
Number
Revision
Date: 4.15.2023
Sheet of
File: C:\Users\..\muffin_holder_sheet.SchDoc
Drawn By:


H1
R7
H2
R8
R1
D1
D2
R2
U1
D3
R3
D4
R4
D5
R5
R6
D6
H3
R9
R10
H4

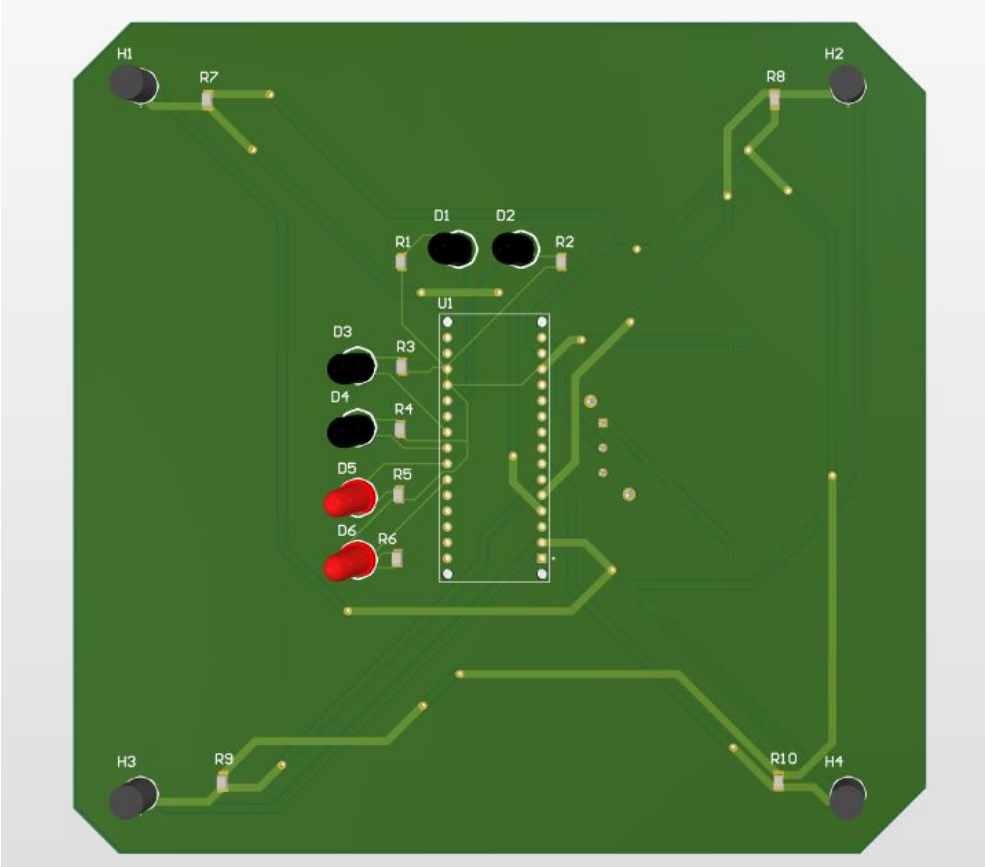
Design Rules Verification Report

Filename : C:\Users\Public\Documents\Altium\Projects\bachelor\muffin_holder_pcb\muffin

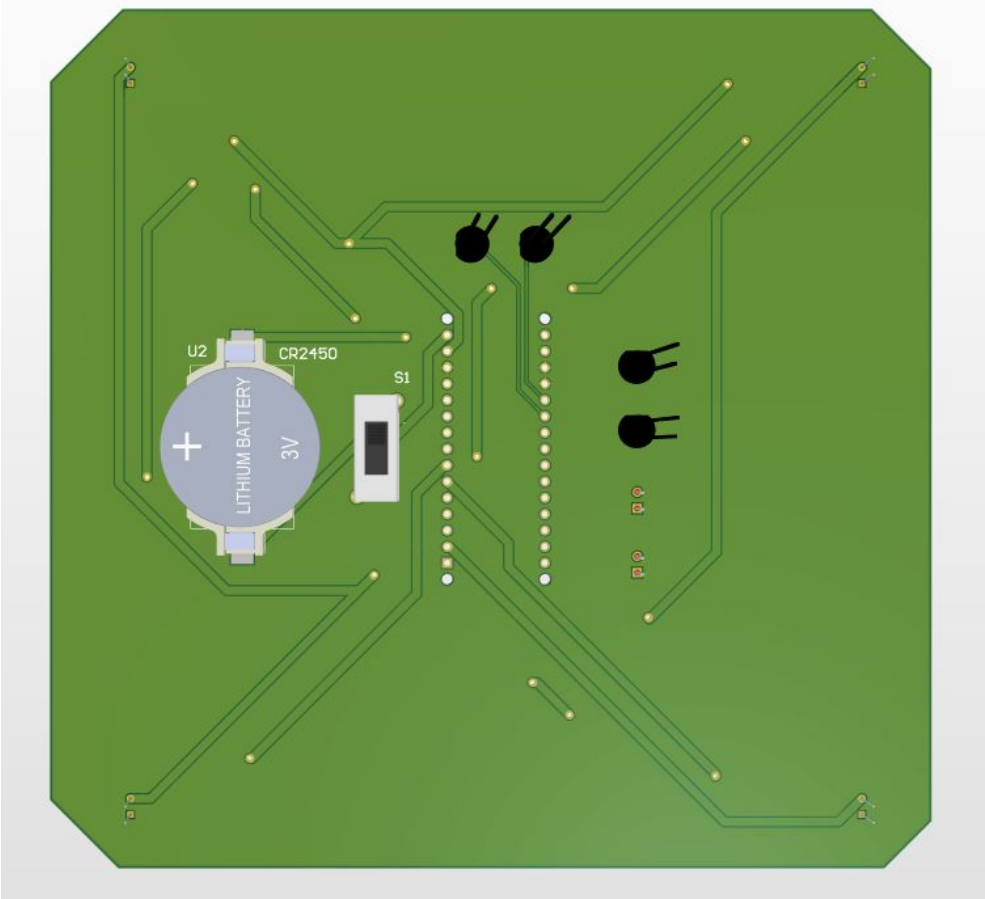
Warnings 0
Rule Violations 0

Warnings	
Total	0

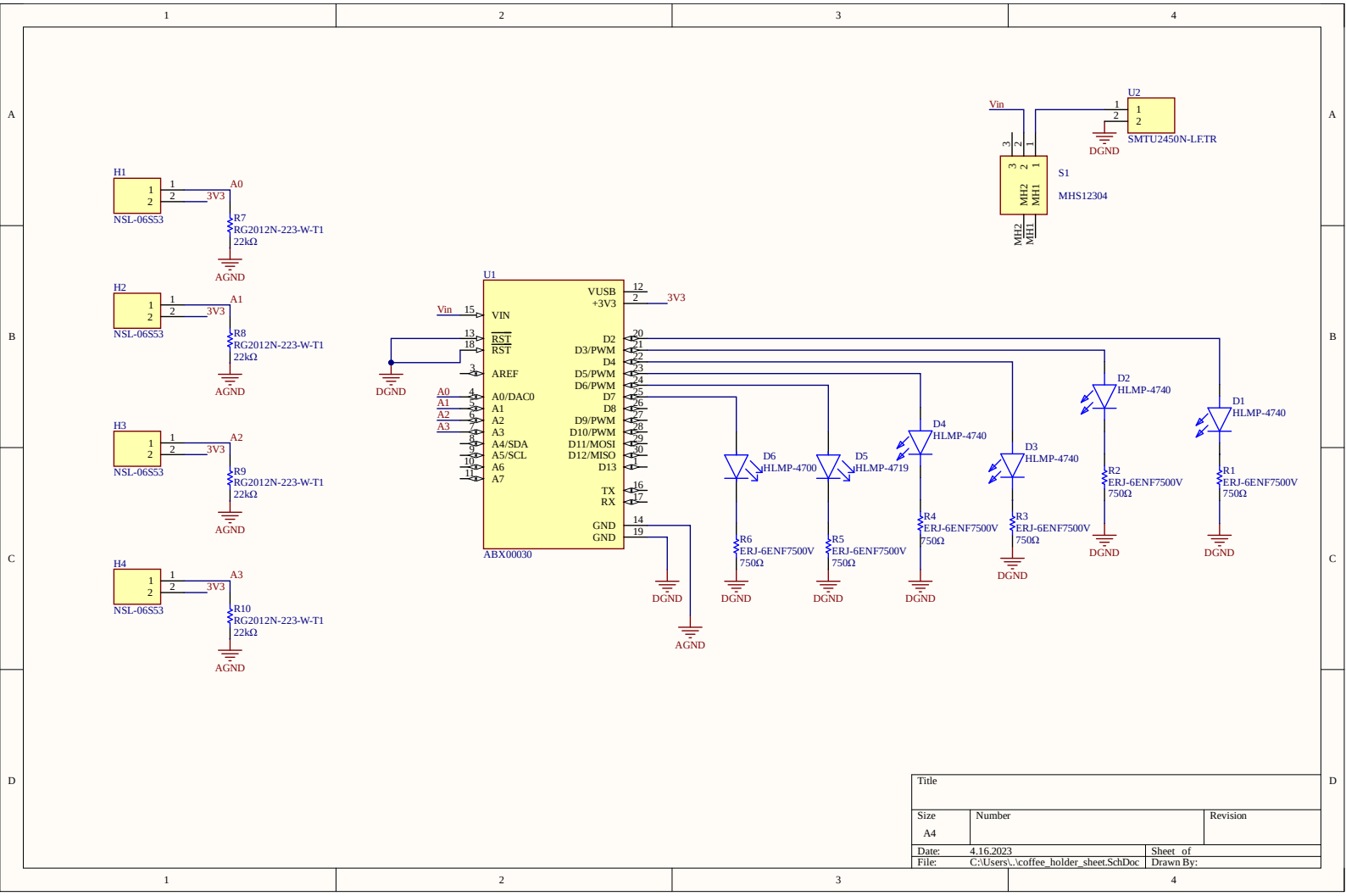
Rule Violations	
Clearance Constraint (Gap=10mil) (All),(All)	0
Short-Circuit Constraint (Allowed=No) (All),(All)	0
Un-Routed Net Constraint ((All))	0
Modified Polygon (Allow modified: No), (Allow shelved: No)	0
Width Constraint (Min=10mil) (Max=10mil) (Preferred=10mil) (Disabled)(All)	0
Power Plane Connect Rule(Relief Connect)(Expansion=20mil) (Conductor Width=10mil) (Air Gap=10mil) (Entries=4)	0
Hole Size Constraint (Min=1mil) (Max=100mil) (All)	0
Hole To Hole Clearance (Gap=10mil) (All),(All)	0
Minimum Solder Mask Sliver (Gap=10mil) (All),(All)	0
Silk To Solder Mask (Clearance=10mil) (IsPad),(All)	0
Silk to Silk (Clearance=10mil) (All),(All)	0
Net Antennae (Tolerance=0mil) (All)	0
Height Constraint (Min=0mil) (Max=1000mil) (Prefered=500mil) (All)	0
Total	0

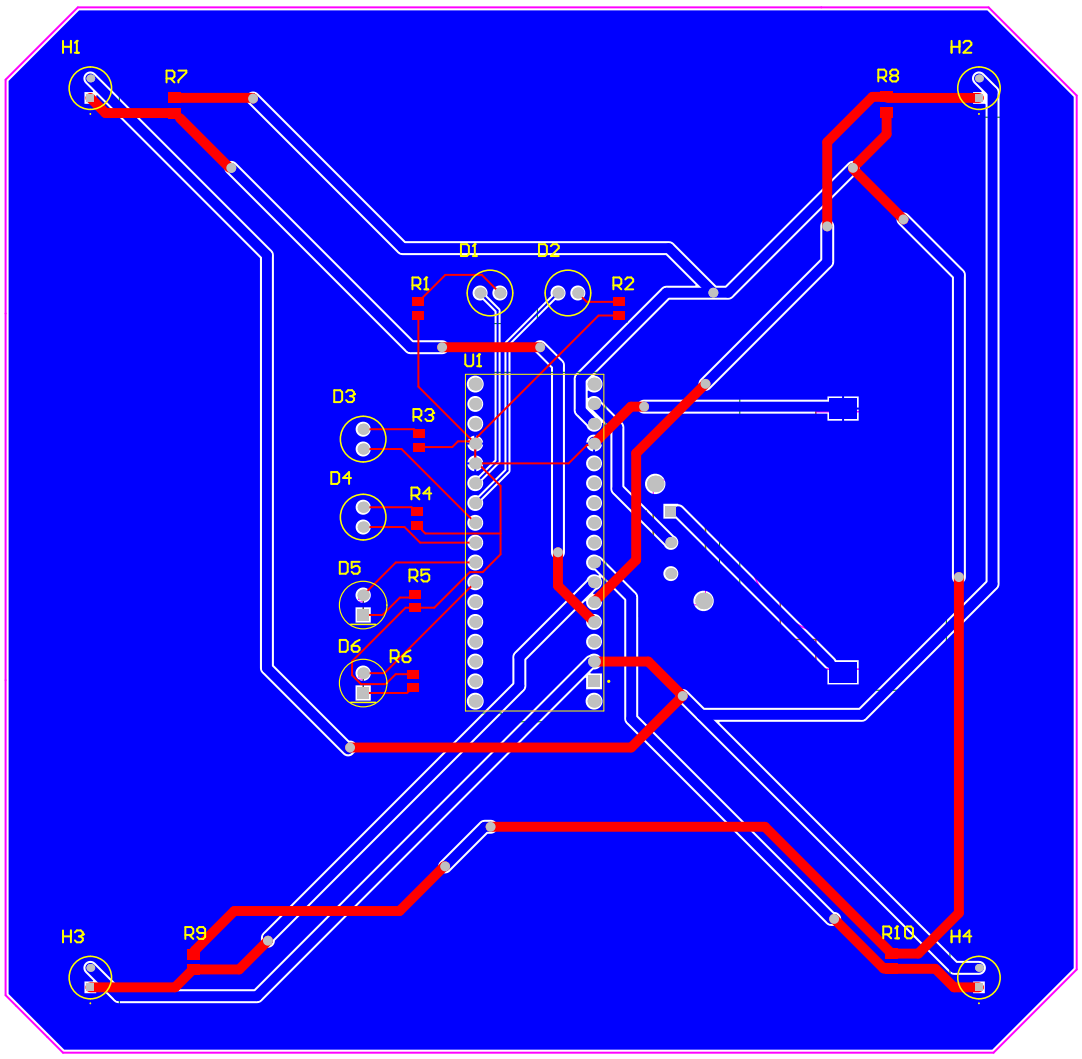


Figur A.21: Koppholder PCB-ens forside i 3D



Figur A.22: Koppholder PCB-ens bakside i 3D


H1
NSL-06S53
3V3
A0
R7
RG2012N-223-W-T1
22kΩ
AGND
H2
NSL-06S53
3V3
A1
R8
RG2012N-223-W-T1
22kΩ
AGND
H3
NSL-06S53
3V3
A2
R9
RG2012N-223-W-T1
22kΩ
AGND
H4
NSL-06S53
3V3
A3
R10
RG2012N-223-W-T1
22kΩ
AGND
U1
ABX00030
VIN
RST
AREF
A0/DAC0
A1
A2
A3
A4/SDA
A5/SCL
A6
A7
VUSB
+3V3
D2
D3/PWM
D4
D5/PWM
D6/PWM
D7
D8
D9/PWM
D10/PWM
D11/MOSI
D12/MISO
D13
TX
RX
GND
GND
Vin
DGND
AGND
U2
SMTU2450N-LF.TR
S1
MHS12304
MH2
MH1
D6
HLMP-4700
D5
HLMP-4719
D4
HLMP-4740
D3
HLMP-4740
D2
HLMP-4740
D1
HLMP-4740
R6
ERJ-6ENF7500V
750Ω
R5
ERJ-6ENF7500V
750Ω
R4
ERJ-6ENF7500V
750Ω
R3
ERJ-6ENF7500V
750Ω
R2
ERJ-6ENF7500V
750Ω
R1
ERJ-6ENF7500V
750Ω
Title
Size
A4
Number
Revision
Date: 4.16.2023
Sheet of
File: C:\Users\..\coffee_holder_sheet.SchDoc
Drawn By:


H1
R7
R8
H2
D1
D2
R1
R2
U1
D3
R3
D4
R4
D5
R5
D6
R6
H3
R9
R10
H4

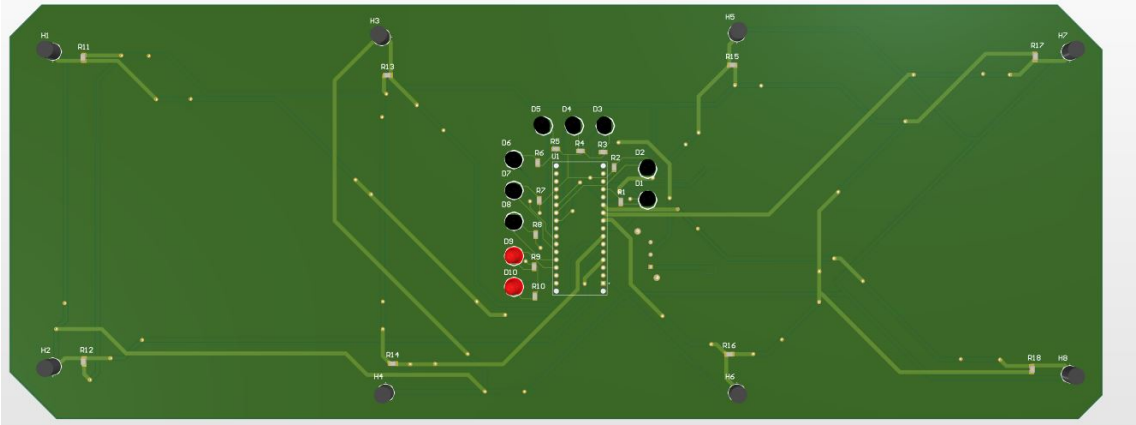
Design Rules Verification Report

Filename : C:\Users\Public\Documents\Altium\Projects\bachelor\coffee_holder_pcb\coffee

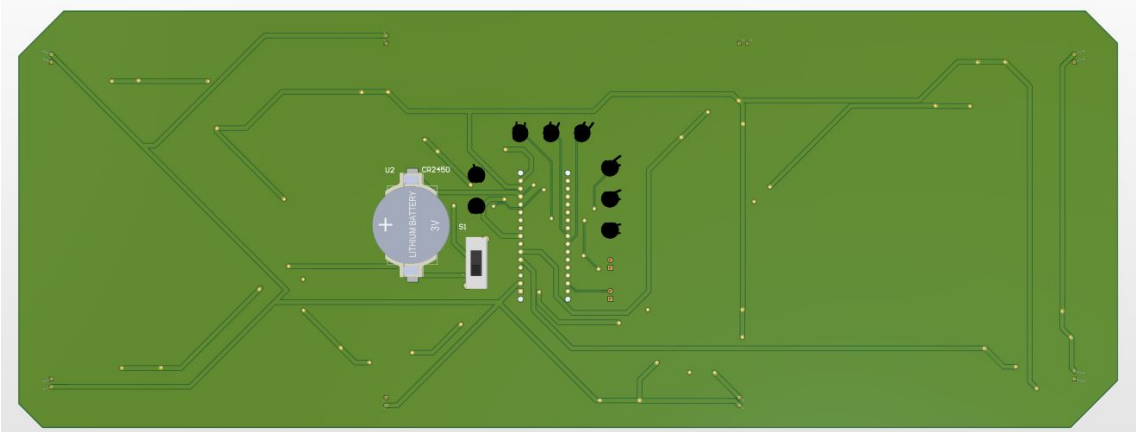
Warnings 0
Rule Violations 0

Warnings	
Total	0

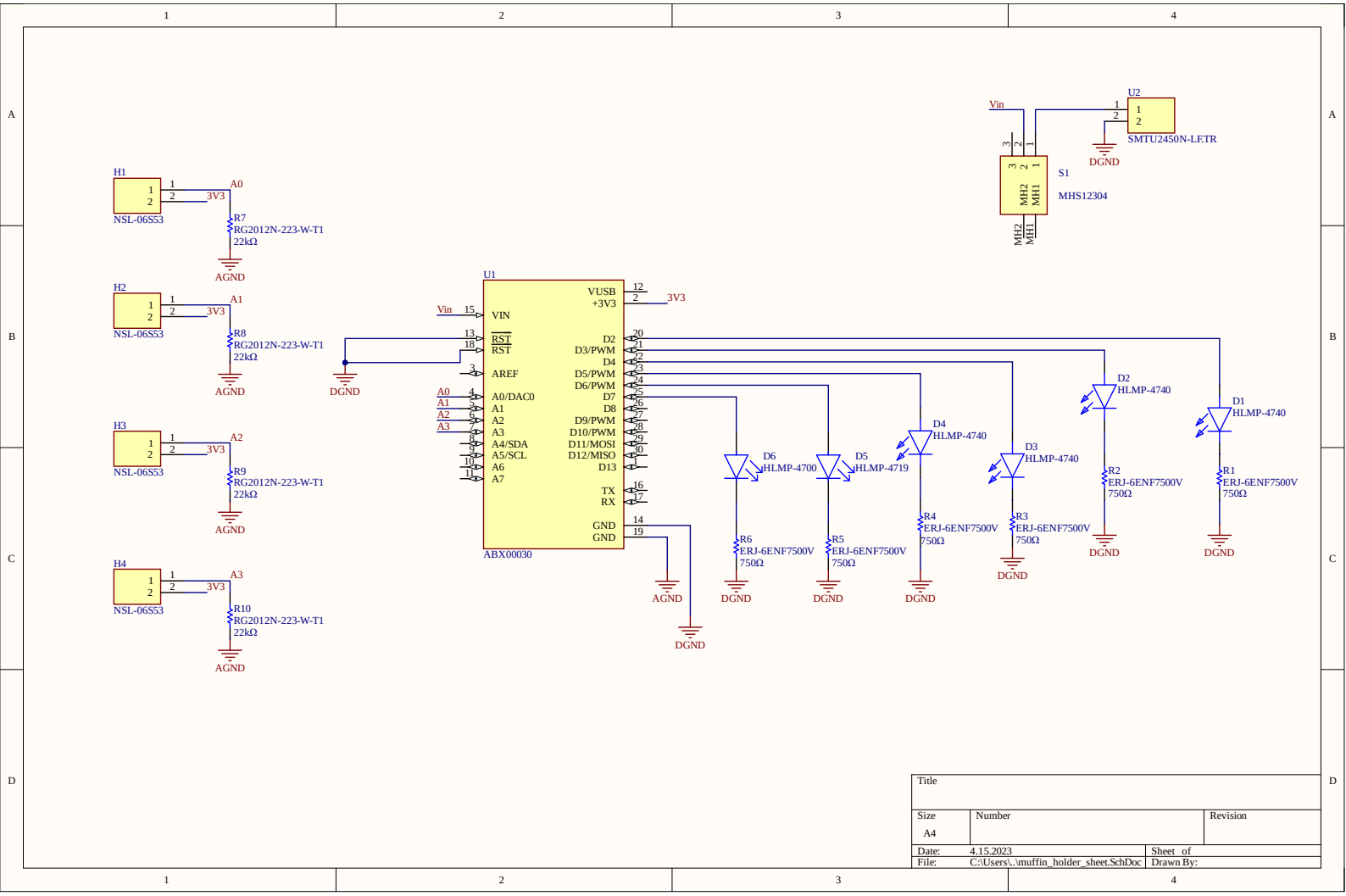
Rule Violations	
Clearance Constraint (Gap=0.254mm) (All),(All)	0
Short-Circuit Constraint (Allowed=No) (All),(All)	0
Un-Routed Net Constraint ((All))	0
Modified Polygon (Allow modified: No), (Allow shelved: No)	0
Width Constraint (Min=0.254mm) (Max=0.254mm) (Preferred=0.254mm) (Disabled)(All)	0
Power Plane Connect Rule(Relief Connect)(Expansion=0.508mm) (Conductor Width=0.254mm) (Air Gap=0.254mm)	0
Hole Size Constraint (Min=0.025mm) (Max=2.54mm) (All)	0
Hole To Hole Clearance (Gap=0.254mm) (All),(All)	0
Minimum Solder Mask Sliver (Gap=0.254mm) (All),(All)	0
Silk To Solder Mask (Clearance=0.254mm) (IsPad),(All)	0
Silk to Silk (Clearance=0.254mm) (All),(All)	0
Net Antennae (Tolerance=0mm) (All)	0
Height Constraint (Min=0mm) (Max=25.4mm) (Prefered=12.7mm) (All)	0
Total	0

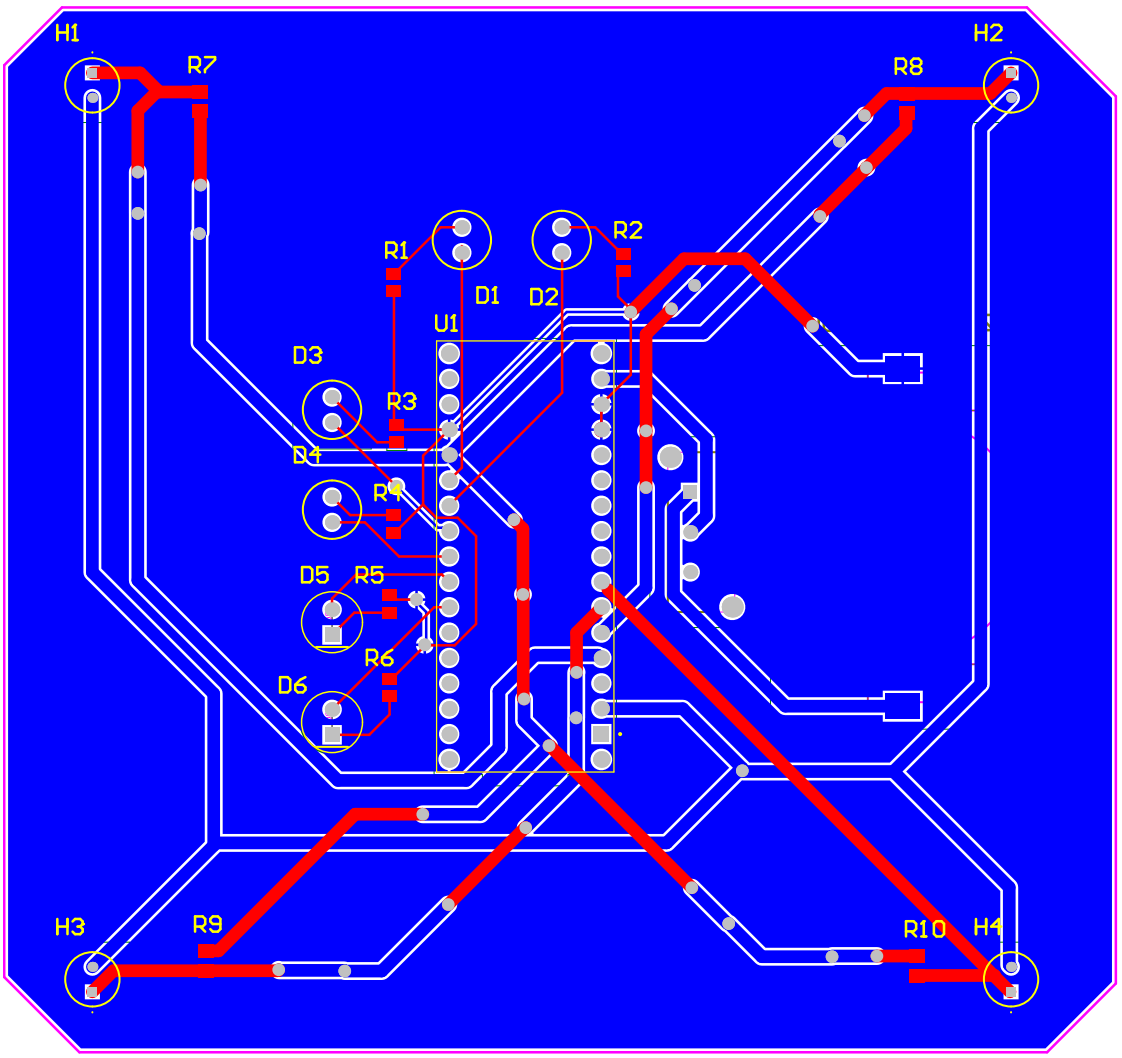


Figur A.23: Serveringsbrett PCB-ens forside i 3D



Figur A.24: Serveringsbrett PCB-ens bakside i 3D


H1
NSL-06S53
H2
NSL-06S53
H3
NSL-06S53
H4
NSL-06S53
3V3
A0
A1
A2
A3
R7
RG2012N-223-W-T1
22kΩ
R8
RG2012N-223-W-T1
22kΩ
R9
RG2012N-223-W-T1
22kΩ
R10
RG2012N-223-W-T1
22kΩ
AGND
U1
ABX00030
VIN
RST
AREF
A0/DAC0
A1
A2
A3
A4/SDA
A5/SCL
A6
A7
VUSB
+3V3
D2
D3/PWM
D4
D5/PWM
D6/PWM
D7
D8
D9/PWM
D10/PWM
D11/MOSI
D12/MISO
D13
TX
RX
GND
GND
Vin
DGND
U2
SMTU2450N-LF.TR
S1
MHS12304
MH2
MH1
D6
HLMP-4700
D5
HLMP-4719
D4
HLMP-4740
D3
HLMP-4740
D2
HLMP-4740
D1
HLMP-4740
R6
ERJ-6ENF7500V
750Ω
R5
ERJ-6ENF7500V
750Ω
R4
ERJ-6ENF7500V
750Ω
R3
ERJ-6ENF7500V
750Ω
R2
ERJ-6ENF7500V
750Ω
R1
ERJ-6ENF7500V
750Ω
Title
Size
A4
Number
Revision
Date: 4.15.2023
Sheet of
File: C:\Users\..\muffin_holder_sheet.SchDoc
Drawn By:


H1
R7
H2
R8
R1
R2
D1
D2
U1
D3
R3
D4
R4
D5
R5
R6
D6
H3
R9
R10
H4

Design Rules Verification Report

Filename : C:\Users\Public\Documents\Altium\Projects\bachelor\serving_tray_pcb\serving_ Warnings 0
Rule Violations 0

Warnings	
Total	0

Rule Violations	
Clearance Constraint (Gap=0.254mm) (All),(All)	0
Short-Circuit Constraint (Allowed=No) (All),(All)	0
Un-Routed Net Constraint ((All))	0
Modified Polygon (Allow modified: No), (Allow shelved: No)	0
Width Constraint (Min=0.254mm) (Max=0.254mm) (Preferred=0.254mm) (Disabled)(All)	0
Power Plane Connect Rule(Relief Connect)(Expansion=0.508mm) (Conductor Width=0.254mm) (Air Gap=0.254mm)	0
Hole Size Constraint (Min=0.025mm) (Max=2.54mm) (All)	0
Hole To Hole Clearance (Gap=0.254mm) (All),(All)	0
Minimum Solder Mask Sliver (Gap=0.254mm) (All),(All)	0
Silk To Solder Mask (Clearance=0.254mm) (IsPad),(All)	0
Silk to Silk (Clearance=0.254mm) (All),(All)	0
Net Antennae (Tolerance=0mm) (All)	0
Height Constraint (Min=0mm) (Max=25.4mm) (Prefered=12.7mm) (All)	0
Total	0

Name	Description	Muffin holder	Coffee holder	Serving tray	Two muffin holders, two coffee holders and one serving tray	Small backup	In total to order
ERJ-6ENF7500V	SMD resistor	6	6	10	34	22	56
RG2012N-223-W-T1	SMD resistor	4	4	8	24	16	40
ABX00034	Arduino Nano BLE	1	1	1	5	1	6
NSL-06S53	LDR	4	4	8	24	2	26
HLMP-4740	Green LED	4	4	8	24	2	26
HLMP-4700	Yellow LED	1	1	1	5	1	6
HLMP-4719	Red LED	1	1	1	5	1	6
MHS12304	Slide switch	1	1	1	5	2	7
SMTU2450N-LF.TR	Coin cell battery holder	1	1	1	5	2	7

Figur A.25: Bill of materials for PCB-ene

```
#include <ArduinoBLE.h>

BLEService coffeeManager("C0FE");
BLEStringCharacteristic nextCoffee("C0F1", BLERead |
    BLENotify, 5);

#define ledStatusPin1 2
#define ledStatusPin2 3
#define ledStatusPin3 4
#define ledStatusPin4 5
#define wirelessPin 6
#define errorPin 7

#define lightObject1 A0
#define lightObject2 A1
#define lightObject3 A2
#define lightObject4 A3

int ledStatus1;
int ledStatus2;
int ledStatus3;
int ledStatus4;

void setup() {
  Serial.begin(9600);

  pinMode(ledStatusPin1, OUTPUT);
  pinMode(ledStatusPin2, OUTPUT);
  pinMode(ledStatusPin3, OUTPUT);
  pinMode(ledStatusPin4, OUTPUT);
  pinMode(lightObject1, INPUT);
  pinMode(lightObject2, INPUT);
  pinMode(lightObject3, INPUT);
  pinMode(lightObject4, INPUT);

  if (!BLE.begin()) {
    Serial.println("Starting Bluetooth Low Energy failed");
    digitalWrite(errorPin, HIGH);
    while (1);
```

```
39    }
40
41    BLE.setLocalName("White coffee holder");
42    BLE.setAdvertisedService(coffeeManager);
43    coffeeManager.addCharacteristic(nextCoffee);
44    BLE.addService(coffeeManager);
45    BLE.advertise();
46    Serial.println("Coffee holder peripheral");
47
48    //power saving measures
49    //digitalWrite(LED_PWR, LOW);
50    digitalWrite(PIN_ENABLE_SENSORS_3V3, LOW);
51    digitalWrite(PIN_ENABLE_I2C_PULLUP, LOW);
52
53  }
54
55
56
57  void loop() {
58    int darkValue = 500;
59    BLEDevice central = BLE.central();
60
61    if (central) {
62      Serial.print("Connected to central: ");
63      Serial.println(central.address());
64      digitalWrite(wirelessPin, HIGH);
65
66      while (central.connected()) {
67        int lightObjectValue1 = analogRead(lightObject1);
68        int lightObjectValue2 = analogRead(lightObject2);
69        int lightObjectValue3 = analogRead(lightObject3);
70        int lightObjectValue4 = analogRead(lightObject4);
71
72
73        //object 1
74        if (lightObjectValue1 <= darkValue) {
75          digitalWrite(ledStatusPin1, HIGH);
76          ledStatus1 = 1;
77        }
78
79        else if (lightObjectValue1 > darkValue) {
80          digitalWrite(ledStatusPin1, LOW);
81          ledStatus1 = 0;
82        }
83
84
85        //object 2
86        if (lightObjectValue2 <= darkValue) {
87          digitalWrite(ledStatusPin2, HIGH);
88          ledStatus2 = 1;
```

```
89        }
90
91        else if (lightObjectValue2 > darkValue) {
92          digitalWrite(ledStatusPin2, LOW);
93          ledStatus2 = 0;
94        }
95
96
97        //object 3
98        if (lightObjectValue3 <= darkValue) {
99          digitalWrite(ledStatusPin3, HIGH);
100         ledStatus3 = 1;
101       }
102
103       else if (lightObjectValue3 > darkValue) {
104         digitalWrite(ledStatusPin3, LOW);
105         ledStatus3 = 0;
106       }
107
108
109       //object 4
110       if (lightObjectValue4 <= darkValue) {
111         digitalWrite(ledStatusPin4, HIGH);
112         ledStatus4 = 1;
113       }
114
115       else if (lightObjectValue4 > darkValue) {
116         digitalWrite(ledStatusPin4, LOW);
117         ledStatus4 = 0;
118       }
119
120       if (ledStatus1 == 1) {
121         Serial.println("Pick up object 1");
122         nextCoffee.writeValue("0000");
123       }
124
125       else if (ledStatus2 == 1) {
126         Serial.println("Pick up object 2");
127         nextCoffee.writeValue("0001");
128       }
129
130       else if (ledStatus3 == 1) {
131         Serial.println("Pick up object 3");
132         nextCoffee.writeValue("0010");
133
134       }
135
136       else if (ledStatus4 == 1) {
137         Serial.println("Pick up object 4");
138         nextCoffee.writeValue("0011");
```

```
139         }
140
141         else {
142           nextCoffee.writeValue("1111");
143         }
144
145
146
147       }
148     }
149     digitalWrite(wirelessPin, LOW);
150     Serial.print("Disconnected from central");
151     Serial.println(central.address());
152
153 }
```

Kodeblokk A.1: Programmet på coffeeHolderWhite (og coffeeHolderBlack)

```
1  #include <ArduinoBLE.h>
2
3  BLEService muffinManager("CA0E");
4  BLEStringCharacteristic nextMuffin("CA1E", BLERead |
       BLENotify, 5);
5
6  #define ledStatusPin1 2
7  #define ledStatusPin2 3
8  #define ledStatusPin3 4
9  #define ledStatusPin4 5
10 #define wirelessPin 6
11 #define errorPin 7
12
13 #define lightObject1 A0
14 #define lightObject2 A1
15 #define lightObject3 A2
16 #define lightObject4 A3
17
18 int ledStatus1;
19 int ledStatus2;
20 int ledStatus3;
21 int ledStatus4;
22
23 void setup() {
24   Serial.begin(9600);
25
26   pinMode(ledStatusPin1, OUTPUT);
27   pinMode(ledStatusPin2, OUTPUT);
28   pinMode(ledStatusPin3, OUTPUT);
29   pinMode(ledStatusPin4, OUTPUT);
30   pinMode(lightObject1, INPUT);
```

```
31    pinMode(lightObject2, INPUT);
32    pinMode(lightObject3, INPUT);
33    pinMode(lightObject4, INPUT);
34
35    if (!BLE.begin()) {
36      Serial.println("Starting Bluetooth Low Energy failed");
37      digitalWrite(errorPin, HIGH);
38      while (1);
39    }
40
41    BLE.setLocalName("White muffin holder");
42    BLE.setAdvertisedService(muffinManager);
43    muffinManager.addCharacteristic(nextMuffin);
44    BLE.addService(muffinManager);
45    BLE.advertise();
46    Serial.println("Muffin holder peripheral");
47
48    //power saving measures
49    //digitalWrite(LED_PWR, LOW);
50    digitalWrite(PIN_ENABLE_SENSORS_3V3, LOW);
51    digitalWrite(PIN_ENABLE_I2C_PULLUP, LOW);
52
53  }
54
55
56
57  void loop() {
58    int darkValue = 500;
59    BLEDevice central = BLE.central();
60
61    if (central) {
62      Serial.print("Connected to central: ");
63      Serial.println(central.address());
64      digitalWrite(wirelessPin, HIGH);
65
66      while (central.connected()) {
67        int lightObjectValue1 = analogRead(lightObject1);
68        int lightObjectValue2 = analogRead(lightObject2);
69        int lightObjectValue3 = analogRead(lightObject3);
70        int lightObjectValue4 = analogRead(lightObject4);
71
72
73        //object 1
74        if (lightObjectValue1 <= darkValue) {
75          digitalWrite(ledStatusPin1, HIGH);
76          ledStatus1 = 1;
77        }
78
79        else if (lightObjectValue1 > darkValue) {
80          digitalWrite(ledStatusPin1, LOW);
```

```
81          ledStatus1 = 0;
82        }
83
84
85        //object 2
86        if (lightObjectValue2 <= darkValue) {
87          digitalWrite(ledStatusPin2, HIGH);
88          ledStatus2 = 1;
89        }
90
91        else if (lightObjectValue2 > darkValue) {
92          digitalWrite(ledStatusPin2, LOW);
93          ledStatus2 = 0;
94        }
95
96
97        //object 3
98        if (lightObjectValue3 <= darkValue) {
99          digitalWrite(ledStatusPin3, HIGH);
100         ledStatus3 = 1;
101       }
102
103       else if (lightObjectValue3 > darkValue) {
104         digitalWrite(ledStatusPin3, LOW);
105         ledStatus3 = 0;
106       }
107
108
109       //object 4
110       if (lightObjectValue4 <= darkValue) {
111         digitalWrite(ledStatusPin4, HIGH);
112         ledStatus4 = 1;
113       }
114
115       else if (lightObjectValue4 > darkValue) {
116         digitalWrite(ledStatusPin4, LOW);
117         ledStatus4 = 0;
118       }
119
120       if (ledStatus1 == 1) {
121         Serial.println("Pick up object 1");
122         nextMuffin.writeValue("0000");
123       }
124
125       else if (ledStatus2 == 1) {
126         Serial.println("Pick up object 2");
127         nextMuffin.writeValue("0001");
128       }
129
130       else if (ledStatus3 == 1) {
```

```
131         Serial.println("Pick up object 3");
132         nextMuffin.writeValue("0010");
133       }
134
135       else if (ledStatus4 == 1) {
136         Serial.println("Pick up object 4");
137         nextMuffin.writeValue("0011");
138       }
139
140       else {
141         nextMuffin.writeValue("1111");
142       }
143
144     }
145   }
146   digitalWrite(wirelessPin, LOW);
147   Serial.print("Disconnected from central");
148   Serial.println(central.address());
149
150 }
```

Kodeblokk A.2: Programmet på muffinHolderWhite (og muffinHolderBlack)

```
1  #include <ArduinoBLE.h>
2
3  BLEService servingManager("0B0C");
4  BLEStringCharacteristic servingData("0B1C", BLERead |
       BLENotify, 9);
5
6  char servingArray[9] = "00000000";;
7
8  #define ledStatusPin1 2
9  #define ledStatusPin2 3
10 #define ledStatusPin3 4
11 #define ledStatusPin4 5
12 #define ledStatusPin5 6
13 #define ledStatusPin6 7
14 #define ledStatusPin7 8
15 #define ledStatusPin8 9
16 #define wirelessPin 10
17 #define errorPin 11
18
19 #define lightObject1 A0
20 #define lightObject2 A1
21 #define lightObject3 A2
22 #define lightObject4 A3
23 #define lightObject5 A4
24 #define lightObject6 A5
```

```
#define lightObject7 A6
#define lightObject8 A7

int ledStatus1;
int ledStatus2;
int ledStatus3;
int ledStatus4;
int ledStatus5;
int ledStatus6;
int ledStatus7;
int ledStatus8;

void setup() {
  Serial.begin(9600);

  pinMode(ledStatusPin1, OUTPUT);
  pinMode(ledStatusPin2, OUTPUT);
  pinMode(ledStatusPin3, OUTPUT);
  pinMode(ledStatusPin4, OUTPUT);
  pinMode(ledStatusPin5, OUTPUT);
  pinMode(ledStatusPin6, OUTPUT);
  pinMode(ledStatusPin7, OUTPUT);
  pinMode(ledStatusPin8, OUTPUT);
  pinMode(wirelessPin, OUTPUT);
  pinMode(errorPin, OUTPUT);

  pinMode(lightObject1, INPUT);
  pinMode(lightObject2, INPUT);
  pinMode(lightObject3, INPUT);
  pinMode(lightObject4, INPUT);
  pinMode(lightObject5, INPUT);
  pinMode(lightObject6, INPUT);
  pinMode(lightObject7, INPUT);
  pinMode(lightObject8, INPUT);

  if (!BLE.begin()) {
    Serial.println("Starting Bluetooth Low Energy failed");
    digitalWrite(errorPin, HIGH);
    while (1);
  }


  BLE.setLocalName("Serving tray");
  BLE.setAdvertisedService(servingManager);
  servingManager.addCharacteristic(servingData);
  BLE.addService(servingManager);
  BLE.advertise();
  Serial.println("Serving tray pheripheral");

  //power saving measures
```

```
75   //digitalWrite(LED_PWR, LOW);
76   digitalWrite(PIN_ENABLE_SENSORS_3V3, LOW);
77   digitalWrite(PIN_ENABLE_I2C_PULLUP, LOW);
78 }
79
80 void loop() {
81
82   int darkValue = 200;
83
84   BLEDevice central = BLE.central();
85
86   if (central) {
87     Serial.print("Connected to central: ");
88     Serial.println(central.address());
89     digitalWrite(wirelessPin, HIGH);
90
91     while (central.connected()) {
92       int lightObjectValue1 = analogRead(lightObject1);
93       int lightObjectValue2 = analogRead(lightObject2);
94       int lightObjectValue3 = analogRead(lightObject3);
95       int lightObjectValue4 = analogRead(lightObject4);
96       int lightObjectValue5 = analogRead(lightObject5);
97       int lightObjectValue6 = analogRead(lightObject6);
98       int lightObjectValue7 = analogRead(lightObject7);
99       int lightObjectValue8 = analogRead(lightObject8);
100
101      //object 1
102      if (lightObjectValue1 <= darkValue) {
103        digitalWrite(ledStatusPin1, HIGH);
104        ledStatus1 = 1;
105      }
106
107      else if (lightObjectValue1 > darkValue) {
108        digitalWrite(ledStatusPin1, LOW);
109        ledStatus1 = 0;
110      }
111
112
113      //object 2
114      if (lightObjectValue2 <= darkValue) {
115        digitalWrite(ledStatusPin2, HIGH);
116        ledStatus2 = 1;
117      }
118
119      else if (lightObjectValue2 > darkValue) {
120        digitalWrite(ledStatusPin2, LOW);
121        ledStatus2 = 0;
122      }
123
124
```

```
125       //object 3
126       if (lightObjectValue3 <= darkValue) {
127         digitalWrite(ledStatusPin3, HIGH);
128         ledStatus3 = 1;
129       }
130
131       else if (lightObjectValue3 > darkValue) {
132         digitalWrite(ledStatusPin3, LOW);
133         ledStatus3 = 0;
134       }
135
136
137       //object 4
138       if (lightObjectValue4 <= darkValue) {
139         digitalWrite(ledStatusPin4, HIGH);
140         ledStatus4 = 1;
141       }
142
143       else if (lightObjectValue4 > darkValue) {
144         digitalWrite(ledStatusPin4, LOW);
145         ledStatus4 = 0;
146       }
147
148       //object 5
149       if (lightObjectValue5 <= darkValue) {
150         digitalWrite(ledStatusPin5, HIGH);
151         ledStatus5 = 1;
152       }
153
154       else if (lightObjectValue5 > darkValue) {
155         digitalWrite(ledStatusPin5, LOW);
156         ledStatus5 = 0;
157       }
158
159
160       //object 6
161       if (lightObjectValue6 <= darkValue) {
162         digitalWrite(ledStatusPin6, HIGH);
163         ledStatus6 = 1;
164       }
165
166       else if (lightObjectValue6 > darkValue) {
167         digitalWrite(ledStatusPin6, LOW);
168         ledStatus6 = 0;
169       }
170
171
172       //object 7
173       if (lightObjectValue7 <= darkValue) {
174         digitalWrite(ledStatusPin7, HIGH);
```

```
175         ledStatus7 = 1;
176       }
177
178       else if (lightObjectValue7 > darkValue) {
179         digitalWrite(ledStatusPin7, LOW);
180         ledStatus7 = 0;
181       }
182
183
184       //object 8
185       if (lightObjectValue8 <= darkValue) {
186         digitalWrite(ledStatusPin8, HIGH);
187         ledStatus8 = 1;
188       }
189
190       else if (lightObjectValue8 > darkValue) {
191         digitalWrite(ledStatusPin8, LOW);
192         ledStatus8 = 0;
193       }
194
195       if (ledStatus1 == 1) {
196         servingArray[7] = '1';
197       }
198
199       else {
200         servingArray[7] = '0';
201       }
202
203       if (ledStatus2 == 1) {
204         servingArray[6] = '1';
205       }
206
207       else {
208         servingArray[6] = '0';
209       }
210
211       if (ledStatus3 == 1) {
212         servingArray[5] = '1';
213       }
214       else {
215         servingArray[5] = '0';
216       }
217
218
219       if (ledStatus4 == 1) {
220         servingArray[4] = '1';
221       }
222       else {
223         servingArray[4] = '0';
224       }
```

```
225
226       if (ledStatus5 == 1) {
227         servingArray[3] = '1';
228       }
229       else  {
230         servingArray[3] = '0';
231       }
232
233       if (ledStatus6 == 1) {
234         servingArray[2] = '1';
235       }
236       else {
237         servingArray[2] = '0';
238       }
239
240       if (ledStatus7 == 1) {
241         servingArray[1] = '1';
242       }
243       else {
244         servingArray[1] = '0';
245       }
246
247       if (ledStatus8 == 1) {
248         servingArray[0] = '1';
249       }
250       else {
251         servingArray[0] = '0';
252       }
253
254       servingData.writeValue(servingArray);
255       Serial.print("servingData transmitted: ");
256       Serial.println(servingArray);
257
258     }
259   }
260
261   digitalWrite(wirelessPin, LOW);
262   Serial.print("Disconnected from central: ");
263   Serial.println(central.address());
264 }
```

Kodeblokk A.3: Programmet på servingTray

```
1 #!/usr/bin/env python3
2
3 import asyncio
4 import rospy
5 from std_msgs.msg import String
6 from bleak import BleakClient
7 from bleak.exc import BleakDBusError, BleakDeviceNotFoundError,
```

```
    BleakError

async def main():

    #MAC addresses
    muffin_holder_white_mac = "A0:9A:B7:3A:DE:F3"
    muffin_holder_black_mac = "E4:FB:DF:C7:30:A4"
    coffee_holder_white_mac = "FB:DB:2B:41:E2:E3"
    coffee_holder_black_mac = "31:6D:13:CA:BA:1B"
    serving_tray_mac= "A0:61:91:54:42:97"

    #characteristics
    muffin_holder_white_charac = "0000CA1E-0000-1000-8000-00805
    F9B34FB"
    muffin_holder_black_charac = "0000CA2E-0000-1000-8000-00805
    F9B34FB"
    coffee_holder_white_charac = "0000C0F1-0000-1000-8000-00805
    F9B34FB"
    coffee_holder_black_charac = "0000C0F2-0000-1000-8000-00805
    F9B34FB"
    serving_tray_charac = "00000B1C-0000-1000-8000-00805F9B34FB"

    #define BleakClient objects
    white_muffin_holder = BleakClient(muffin_holder_white_mac)
    black_muffin_holder = BleakClient(muffin_holder_black_mac)
    white_coffee_holder = BleakClient(coffee_holder_white_mac)
    black_coffee_holder = BleakClient(coffee_holder_black_mac)
    serving_tray = BleakClient(serving_tray_mac)

    #init values
    next_muffin_white = "1110"
    next_muffin_black = "1110"
    next_coffee_white = "1110"
    next_coffee_black = "1110"
    #since serving_data doesn't have error codes and if it doesn't
    have a value yet, it'll most likely be in the beginning of the
    program
    serving_data = "00000000"

    rospy.init_node("raspi_gateway")
    muffin_manager = rospy.Publisher("nextMuffin", String,
    queue_size = 5)
    coffee_manager = rospy.Publisher("nextCoffee", String,
    queue_size = 5)
    serving_manager = rospy.Publisher("servingData", String,
    queue_size = 9)
    #rate = rospy.Rate(10)

    #clearing previous connections
    print("Disconnecting devices")
    while (white_muffin_holder.is_connected == True):
        await white_muffin_holder.disconnect()

    while (black_muffin_holder.is_connected == True):
```

```
52          await black_muffin_holder.disconnect()
53
54      while (white_coffee_holder.is_connected == True):
55          await white_coffee_holder.disconnect()
56
57      while (black_coffee_holder.is_connected == True):
58          await black_coffee_holder.disconnect()
59
60      while (serving_tray.is_connected == True):
61          await serving_tray.disconnect()
62
63      await asyncio.sleep(5)
64
65      print("Starting program")
66
67      while True:
68
69          ####################WHITE COFFEE HOLDER###################
70
71          try:
72              print("Initiating connection with white coffee holder")
73              await white_coffee_holder.connect()
74              next_coffee_white = await white_coffee_holder.
    read_gatt_char(coffee_holder_white_charac)
75              print("FETCHED next_coffee_white :", next_coffee_white)
76              await asyncio.sleep(1)
77          except BleakDBusError:
78              print("Retrying to connect")
79              try:
80                  await asyncio.sleep(2.5)
81                  await white_coffee_holder.connect()
82                  next_coffee_white = await white_coffee_holder.
    read_gatt_char(coffee_holder_white_charac)
83                  print("FETCHED next_coffee_white :",
    next_coffee_white)
84                  await asyncio.sleep(1)
85              except Exception:
86                  print("Reconnecting white coffee holder failed, we'
    ll try again later")
87          except (BleakDeviceNotFoundError, BleakError):
88              print("Device not found")
89              try:
90                  await white_coffee_holder.disconnect()
91                  await asyncio.sleep(2.5)
92                  await white_coffee_holder.connect()
93                  next_coffee_white = await white_coffee_holder.
    read_gatt_char(coffee_holder_white_charac)
94                  print("FETCHED next_coffee_white :",
    next_coffee_white)
95                  await asyncio.sleep(1)
96              except Exception:
97                  print("Reconnecting white coffee holder failed, we'
    ll try again later")
98
```

```
99
100         await white_coffee_holder.disconnect()
101         await asyncio.sleep(2.5)
102
103
104          ##########################BLACK COFFEE HOLDER
    ######################
105
106
107         try:
108             print("Initiating connection with black coffee holder")
109             await black_coffee_holder.connect()
110             next_coffee_black = await black_coffee_holder.
    read_gatt_char(coffee_holder_black_charac)
111             print("FETCHED next_coffee_black :", next_coffee_black)
112             await asyncio.sleep(1)
113         except BleakDBusError:
114             print("Retrying to connect")
115             try:
116                 await asyncio.sleep(2.5)
117                 await black_coffee_holder.connect()
118                 next_coffee_black = await black_coffee_holder.
    read_gatt_char(coffee_holder_black_charac)
119                 print("FETCHED next_coffee_black :",
    next_coffee_black)
120                 await asyncio.sleep(1)
121             except Exception:
122                 print("Reconnecting black coffee holder failed, we'
    ll try again later")
123         except (BleakDeviceNotFoundError, BleakError):
124             print("Device not found")
125             try:
126                 await black_coffee_holder.disconnect()
127                 await asyncio.sleep(2.5)
128                 await black_coffee_holder.connect()
129                 next_coffee_black = await black_coffee_holder.
    read_gatt_char(coffee_holder_black_charac)
130                 print("FETCHED next_coffee_black :",
    next_coffee_black)
131                 await asyncio.sleep(1)
132             except Exception:
133                 print("Reconnecting black coffee holder failed, we'
    ll try again later")
134
135
136         await black_coffee_holder.disconnect()
137         await asyncio.sleep(2.5)
138
139
140         ###########COMPARING NEXT COFFEE VALUES AND SENDING TO ROS
    ##############
141
142         next_coffee_white = str(next_coffee_white)
143         if ("bytearray" in next_coffee_white):
```

```
144             next_coffee_white = next_coffee_white[12:16]
145 
146         next_coffee_black = str(next_coffee_black)
147         if ("bytearray" in next_coffee_black):
148             next_coffee_black = next_coffee_black[12:16]
149 
150 
151         if (next_coffee_white != "1111"):
152             next_coffee = next_coffee_white
153 
154         elif (next_coffee_black != "1111") and (next_coffee_white ==
    "1111"):
155             next_coffee = next_coffee_black
156 
157 
158         else: #next_coffee_black and next_coffee_white = 1111
159             next_coffee = "1111"
160 
161 
162         coffee_manager.publish(next_coffee)
163         await asyncio.sleep(2.5)
164 
165 
166        ##########################WHITE MUFFIN HOLDER
    ######################
167 
168         try:
169             print("Initiating connection with white muffin holder")
170             await white_muffin_holder.connect()
171             next_muffin_white = await white_muffin_holder.
    read_gatt_char(muffin_holder_white_charac)
172             print("FETCHED next_muffin_white: ", next_muffin_white)
173             await asyncio.sleep(1)
174         except BleakDBusError:
175             print("Retrying to connect")
176             try:
177                 await asyncio.sleep(2.5)
178                 await white_muffin_holder.connect()
179                 next_muffin_white = await white_muffin_holder.
    read_gatt_char(muffin_holder_white_charac)
180                 print("FETCHED next_muffin_white: ",
    next_muffin_white)
181                 await asyncio.sleep(1)
182             except Exception:
183                 print("Reconnecting white muffin holder failed, we'
    ll try again later")
184         except (BleakDeviceNotFoundError, BleakError):
185             print("Device not found")
186             try:
187                 await white_muffin_holder.disconnect()
188                 await asyncio.sleep(2.5)
189                 await white_muffin_holder.connect()
190                 next_muffin_white = await white_muffin_holder.
    read_gatt_char(muffin_holder_white_charac)
```

```
191                 print("FETCHED next_muffin_white: ",
    next_muffin_white)
192                 await asyncio.sleep(1)
193             except Exception:
194                 print("Reconnecting white muffin holder failed, we'
    ll try again later")
195
196
197         await white_muffin_holder.disconnect()
198         await asyncio.sleep(2.5)
199
200
201         #############BLACK MUFFIN HOLDER############
202
203
204         try:
205             print("Initiating connection with black muffin holder")
206             await black_muffin_holder.connect()
207             next_muffin_black = await black_muffin_holder.
    read_gatt_char(muffin_holder_black_charac)
208             print("FETCHED next_muffin_black: ", next_muffin_black)
209             await asyncio.sleep(1)
210         except BleakDBusError:
211             print("Retrying to connect")
212             try:
213                 await asyncio.sleep(2.5)
214                 await black_muffin_holder.connect()
215                 next_muffin_black = await black_muffin_holder.
    read_gatt_char(muffin_holder_black_charac)
216                 print("FETCHED next_muffin_black: ",
    next_muffin_black)
217                 await asyncio.sleep(1)
218             except Exception:
219                 print("Reconnecting black muffin holder failed, we'
    ll try again later")
220         except (BleakDeviceNotFoundError, BleakError):
221             print("Device not found")
222             try:
223                 await black_muffin_holder.disconnect()
224                 await asyncio.sleep(2.5)
225                 await black_muffin_holder.connect()
226                 next_muffin_black = await black_muffin_holder.
    read_gatt_char(muffin_holder_black_charac)
227                 print("FETCHED next_muffin_black: ",
    next_muffin_black)
228             except Exception:
229                 print("Reconnecting black muffin holder failed, we'
    ll try again later")
230
231
232         await black_muffin_holder.disconnect()
233         await asyncio.sleep(2.5)
234
235
```

```
236         #############COMPARING NEXT MUFFIN VALUES AND SENDING TO
    ROS############
237 
238         next_muffin_white = str(next_muffin_white)
239         if ("bytearray" in next_muffin_white):
240             next_muffin_white = next_muffin_white[12:16]
241 
242         next_muffin_black = str(next_muffin_black)
243         if ("bytearray" in next_muffin_black):
244             next_muffin_black = next_muffin_black[12:16]
245 
246 
247         if (next_muffin_white != "1111"):
248             next_muffin = next_muffin_white
249 
250         elif (next_muffin_black != "1111") and (next_muffin_white ==
     "1111"):
251             next_muffin = next_muffin_black
252 
253         else: #next_muffin_black and next_muffin_white = 1111
254             next_muffin = "1111"
255 
256         muffin_manager.publish(next_muffin)
257         await asyncio.sleep(2.5)
258 
259 
260         ########################SERVING TRAY#######################
261 
262         try:
263             print("Initiating connection with serving tray")
264             await serving_tray.connect()
265             serving_data = await serving_tray.read_gatt_char(
    serving_tray_charac)
266             print("FETCHED serving_data: ", serving_data)
267             await asyncio.sleep(1)
268         except BleakDBusError:
269             print("Retrying to connect")
270             try:
271                 await asyncio.sleep(2.5)
272                 await serving_tray.connect()
273                 serving_data = await serving_tray.read_gatt_char(
    serving_tray_charac)
274                 print("FETCHED serving_data: ", serving_data)
275                 await asyncio.sleep(1)
276             except Exception:
277                 print("Reconnecting serving tray failed, we'll try
    again later")
278         except (BleakDeviceNotFoundError, BleakError):
279             print("Device not found")
280             try:
281                 await serving_tray.disconnect()
282                 await asyncio.sleep(2.5)
283                 await serving_tray.connect()
284                 serving_data = await serving_tray.read_gatt_char(
```

```
    serving_tray_charac)
285                 print("FETCHED serving_data: ", serving_data)
286                 await asyncio.sleep(1)
287             except Exception:
288                 print("Reconnecting serving tray failed, we'll try
    again later")
289
290
291         print("Disconnecting serving tray")
292         await serving_tray.disconnect()
293         await asyncio.sleep(2.5)
294
295
296         #############SENDING SERVING DATA TO ROS############
297
298         serving_data = str(serving_data)
299         if ("bytearray" in serving_data):
300             serving_data = serving_data[12:20]
301
302         serving_manager.publish(serving_data)
303         await asyncio.sleep(2.5)
304
305
306 if __name__ == "__main__":
307
308     try:
309         loop = asyncio.get_event_loop()
310         loop.run_until_complete(main())
311     except KeyboardInterrupt:
312         #the C comes from when you interrupt with ^C
313         print("losing event loop")
314         print("Program stopped by user")
315         loop.close()
```

Kodeblokk A.4: RPi-koden med BLE- og ROS-funksjonalitet

Koden til noden som styrer Kompaï

kompai_controller > scripts > movement_node.py > ...

```python
#!/usr/bin/env python3
import rospy


from kompai_drivers.srv import Navigate, NavigateRequest
from kompai_drivers.msg import Docking
from std_srvs.srv import Empty, EmptyResponse

import time


#when dock_statuse = 3 then Kompai is in the charger
dock_status = 3


#this definition i checinkg Komapis dock statuse
def on_dock_statuc_changed(obj):
    global dock_status
    dock_status = obj.status


#If Kompai is docking it will disconnect from the charger
def on_move_request(obj):
    global dock_status
    if dock_status == 3:
        disconnect = rospy.ServiceProxy('/dock/disconnect', Empty)
        disconnect.call()

    #wait 5 seconds
    time.sleep(5)


    # using the builtin service in Kompai /naviagat/goto to make Kompai go to the wanted possition
    navigate = rospy.ServiceProxy("/navigation/goto", Navigate)
    response = navigate.call(NavigateRequest (name = 'nachi2'))

    print(response)
    return EmptyResponse()
```

```
39
40
41  def on_move_bed(obj):
42
43
44      navigate = rospy.ServiceProxy("/navigation/goto", Navigate)
45      response = navigate.call(NavigateRequest (name = 'bed2'))
46
47      print(response)
48      return EmptyResponse()
49
50
51  def on_move_chair(obj):
52
53
54      navigate = rospy.ServiceProxy("/navigation/goto", Navigate)
55      response = navigate.call(NavigateRequest (name = 'chair1'))
56
57      print(response)
58
59
60      return EmptyResponse()
61
62
63
64  def on_move_armchair(obj):
65
66      navigate = rospy.ServiceProxy("/navigation/goto", Navigate)
67      response = navigate.call(NavigateRequest (name = 'armchair2'))
68
69      print(response)
70
71
72      return EmptyResponse()
73
```

```
74
75  def on_move_doorchair(obj):
76
77      navigate = rospy.ServiceProxy("/navigation/goto", Navigate)
78      response = navigate.call(NavigateRequest (name = 'doorchair1'))
79
80      print(response)
81
82
83      return EmptyResponse()
84
85
86  if __name__ == '__main__' :
87      rospy.init_node("movement_node")
88
89      rospy.loginfo("Node has ben started")
90
91      rospy.Subscriber("/dock/status", Docking, on_dock_statuc_changed)
92
93
94      #defining the functions as rosservices.
95      rospy.Service("/nachi", Empty, on_move_request)
96      rospy.Service("/bed1",Empty, on_move_bed)
97      rospy.Service("/armchair1",Empty, on_move_armchair)
98      rospy.Service("/chair1", Empty, on_move_chair)
99      rospy.Service("/doorchair1", Empty, on_move_doorchair)
100
101     #make the node running untill it is killed
102     rospy.spin()
103
```