

Prosjektoppgave i Datateknikk

"Smart-Kollektiv"

**IELET1002
Gruppe 8
Trondheim, vår 2021**



KANDIDATER (etternavn, fornavn):

Mo, Jørgen Andreas
Oseid, Magnus Dag
Rafuna, Albertin
Speleman, Sebastian Gjerde
Tønnesen, Jonathan Haugen
Vik, Andreas

DATO:	FAGKODE:	GRUPPE (navn/nr):	SIDER/BILAG:
17.05.2021	IELET1002	8	53 / 0

FAGLÆRER:

Arne Midjo

TITTEL:

Prosjektoppgave i Datateknikk

SAMMENDRAG:

Innhold

1 Innledning	5
2 Terminologier	7
3 Teori	8
3.1 Hardware	8
3.2 Software	13
3.2.1 Programmeringsspråk	13
3.2.2 Kommunikasjon	14
3.2.3 Kryptering	16
4 Metode	17
4.1 Romkontroller - Varme, lys, lufting, takvifte	18
4.1.1 Prototype	18
4.1.2 Programvareutvikling	19
4.1.3 CoT	25
4.2 Håndkontroller - Booking, dørklokke	26
4.2.1 Booking	26
4.3 Adgangskontroll	29
4.3.1 Logging av personer i kollektivet	29
4.3.2 Innlåsing og kryptering	30
4.3.3 Dørklokke	31
4.4 Raspberry Pi - Server	32
5 Resultat	33
5.1 Romkontroller	33
5.2 Håndkontroller	36
5.2.1 Booking	36
5.3 Adgangskontroll	38
5.4 Raspberry Pi - Server	39
5.5 MQTT-komunikasjon	41
6 Diskusjon	42
6.1 Romkontroller	42
6.1.1 Ytelsesvurdering	42
6.1.2 Produksjonsmodell	43
6.2 Håndkontroller	44
6.2.1 Booking	44
6.3 Adgangskontroll	47
6.3.1 Logging av personer	47
6.3.2 Innlåsing og Kryptering	48
6.3.3 Dørklokke	48
6.4 Raspberry Pi - Server	49
7 Drøfting & konklusjon	50

7.1 Konklusjon	50
7.2 Tilbakemeldinger	50
8 Vedlegg	53

1 Innledning

Denne rapporten omhandler prosjektoppgaven gitt i emnet IELET1002 ved NTNU i Trondheim.

Prosjektet dreier seg rundt å lage en smarthus-styring for et gitt student-kollektiv i en tidsperiode som preges av en global pandemi. Huset er utstyrt med solcellepanel og produserer strøm når forholdene ligger til rette for dette. Det skal være mulig for hver enkelt beboer å booke ulike ressurser i bygget, som f.eks stue, bad og kjøkken, slik at gjeldende smittevernregler til enhver tid kan overholdes.

Prosjektet innebærer programmering av mikrokontrollere og oppkobling mot en server i en IoT-konfigurasjon.

Oppgaven har blitt løst ved å utvikle tre typer mikrokontroller-enheter; en veggmontert romkontroller, en håndholdt kontroller, og et adgangskontrollpanel. I tillegg sørger en felles server for kommunikasjon mellom enhetene og datalagring, i tillegg til innhenting av eksterne data, strømberegning osv.

Målet med dette prosjektet var primært å heve kompetansen på mikrokontroller-programmering innad i gruppa, samtidig som det skulle utvikles et godt produkt som har en høy nok kvalitet til å evt. kunne lanseres til et marked i etterkant. Denne rapporten tar for seg løsningene som har blitt valgt, samt teori, tankegang og vurderinger som ligger bak. Til slutt oppsummeres resultatene som ble oppnådd og potensielle forbedringer.

Som rapporten vil gå nærmere inn på senere, ble prosjektet delt i tre hoved-deler som ble fordelt på gruppemedlemmene. Følgende har vert de forskjellige medlemmenes ansvarsområde under prosjektet:

Romkontroller

- **Jørgen Andreas Mo** - Totalutvikling (sw & hw) ESP32 med unntak av MQTT-kommunikasjon. CoT-kommunikasjon. Bibliotek for LCD-display og menysystem.
- **Albertin Rafuna** MQTT-kommunikasjon mellom ESP32 og RPi.

Håndkontroller

- **Magnus Dag Oseid** - Utvikling bookingsystem (ESP32 og RPi).
- **Jonathan Haugen Tønnesen** - MQTT-kommunikasjon for booking, og funksjon for fremtidsbooking.
- **Andreas Vik** - Funksjon for å åpne ytterdør dersom det ringer på.
- **Jørgen Andreas Mo** - Bibliotek for LCD-display og menysystem

Adgangskontroll

- **Andreas Vik** - Utvikling av adgangskontroll i samarbeid med Sebastian, ringeklokke, og logging av antall personer i leiligheten. Samt MQTT-bibliotek for datakommunikasjon på nettverket.
- **Sebastian Gjerde Speleman** - Utvikling av adgangskontroll i samarbeid med Andreas, ringeklokke, og logging av antall personer i leiligheten.

Server

- **Andreas Vik** - Totalutvikling av server, med unntak av bookingsystem. Utvikling av MQTT-kommunikasjon, oppsett av CoT-kommunikasjon, logging av energiforbruk og lagring av adgangs-nøkler.
- **Albertin Rafuna** - Innhenting av værdata, utvikling av system for beregning av energiproduksjon fra solcellepaneler (RPi).
- **Magnus Dag Oseid** - Bookingsystem

2 Terminologier

ADC:	Analog to Digital Converter
API:	Application Programming Interface
CoT:	Circus of Things
DDoS:	Distributed Denial-of-Service (overbelastningsangrep)
IoT:	Internet of Things
ISR:	Interrupt Service Routine (Avbruddsrutine)
JSON:	Java Script Object Notation
LDR:	Light Dependent Resistor (Fotoresistor)
MQTT:	Message Queuing Telemetry Transport
NFC:	Near Field Communication
NTP:	Network Time Protocol
OOP:	Objektorientert Programmering
PWM:	Pulse Width Modulation (Pulsbreddemodulering)
SPI:	Serial Peripheral Interface
I2C:	Inter-integrated circuit
RFID:	Radio Frequency Identification
RPi:	Raspberry Pi
SHA-2:	Secure Hash Algoritm 2
UID:	Unique Identification

3 Teori

3.1 Hardware

Raspberry Pi

Raspberry Pi er et britisk selskap som spesialiserer seg på "single board computers". Selskapet produserer flere modeller med ulik ytelse og i dette prosjektet er det brukt RPi 3 modell B+. Den har innebygde antenner for WiFi og Bluetooth, og er et verktøy som er godt egnet for å lære programmering, da den har mange av de samme egenskapene som en tradisjonell datamaskin. En RPi er ved hjelp av dedikerte utvidelsesmoduler også i stand til å kommunisere med verden ved hjelp av sensorer. I dette prosjektet har RPi-en spilt en sentral rolle med tanke på kommunikasjon, databehandling og innhenting av eksterne datasett. Eksempelvis blir den i dette prosjektet brukt til innhenting av meteorologisk værdata, og fungerer som server for MQTT-kommunikasjon og administrator for bookingsystemet.[\[11\]](#)

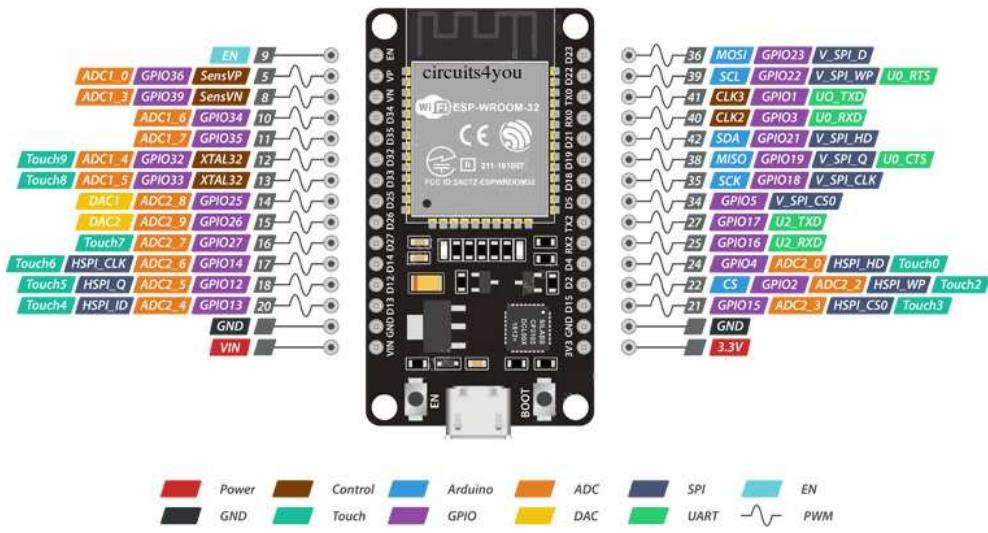


Figure 1: Raspberry pi model 3B+

Espressif Systems - ESP32

ESP32 er en mikrokontroller utviklet av Espressif Systems. ESP32en har integrerte antenner for WiFi og Bluetooth og benytter 2,4 GHz båndet til kommunikasjon. Chipen i denne mikrokontrolleren er en ESP32 WROOM 32D som inneholder en Tensilica Xtensa 32-bit LX6 mikroprosessor og en co-prosessor som vil være på i deep-sleep modus, og som kan brukes til mindre krevende oppgaver. Begge prosessorene opererer med en klokkefrekvens på 160/240 MHz. Mulighet for forskjellige strømsparingsmodus sørger for at energiforbruket kan reduseres ved behov. Prosessorene driftes med en spennin fra 2.2V til 3.3V. Typisk trekkstrøm er 80mA. Programkode lastes opp via en micro-USB-terminal på kortet. Denne terminalen leverer videre 5V til kretskortet, og der er dermed ikke behov for ekstern spenningsforsyning dersom lasten ikke overskridt hva en USB-utgang kan levere. Spenningen blir internt i ESP32-en redusert til 3.3v. [\[14\]](#)

ESP32en inneholder 34 digitale I/O-porter som kan konfigureres som inn- eller utganger avhengig av behov. Kortet inneholder to 12bit ADC-er (Analog to Digital converter) som konverterer et analogt signal til en integer-verdi og en 8-bits DAC (Digital to Analog converter) som gjør det motsatte. Kortet har videre 16 PWM-kanaler som kan brukes til effektregulering, styring av servomotorer og dimming av LEDs, for å nevne noen. For komplett pin-out, se neste figur



ESF 52 Dev. Board / Timeout

Figure 2: Funksjonaliteten til de forskjellige pinnene

Lav kostpris, høy ytelse og mulighet for trådløs kommunikasjon over WiFi og Bluetooth gjør denne mikrokontrolleren til et godt alternativ ved utvikling av IoT-prosjekter. [5]

Servo-motor

MG90S er en liten servomotor som leverer høy utgangseffekt og har 180 graders utslag. I motsetning til en DC-motor har servomotoren en integrert kontroller som gjør at den til enhver tid leser ut orienteringen og dermed lett kan kontrollere posisjonen med et PMW-signal. Servomotoren har driftspenning på 4V-6V. Det finnes C++ bibliotek, åpent på nettet, som gjør programmering og bruk av disse ukomplisert. Dreiemomentet til denne modellen er 2,2 kg ved 4V. I dette prosjektet er servomotoren brukt til å åpne og lukke vinduet i hvert enkelt rom, i tillegg til å operere låsesylinderen i inngangsdøra.[\[15\]](#)

DC-motor

En motor som drives av likespenning. Dreieretningen til motoren bestemmes av strømretningen gjennom den. Den valgte motoren opererer med en driftsspenning på 9V og trekker 68mA ved 100% last. Til effektregulering og dreieretnings-endring benyttes en L293D motordriver. Denne komponenten kan endre retningen på strømmen som går gjennom motoren, ved behov. I dette prosjektet benyttes DC-motoren til visualisering av takvifter på hvert enkelt rom. [16]



Figure 3: Servomotor - MG90S

Temperatursensor.

TMP36 er en sensor som måler temperatur ved at spenningen over en innebygd diode øker/synker lineært med temperaturen. Ved å forsterke spenningsforskjellen genereres et analogt signal som er proporsjonalt med temperaturen i rommet. Ettersom utgangsspenningen er proporsjonal med temperaturen i rommet kan temperaturen finnes v.h.a følgende formel:

$$\text{Temp in } \text{C} = [(V_{\text{out}} \text{ in mV}) - 500]/10.$$



Figure 4: TMP36

Driftsspenninga til sensoren er 2.7V-5.5V. Sensoren har blitt brukt i forbindelse med måling av innetemperaturen i hvert rom. [1]

Lyssensor

En LDR er en fotoresistor som endrer resistans etter hvor mye lys som treffer den. I dette prosjektet benyttes den til automatisk lysregulering på de forskjellige rommene.

NFC kortleser

RFID-RC522 er en NFC-leser som bruker radiofrekvenser til å lese en UID (Unique ID) i et NFC kort. Eksempel på slike kort er student- og ansattkortene til NTNU. Disse kortene har en innebygd chip som blant annet kan lagre en 4-array heksadesimal kode. Studentkortene har alt lagt inn en UID kode, men RFID-RC522 har mulighet til å overskrive UID og legg inn en ny UID. RFID-RC522 bruker SPI og I2C for kommunikasjon, men i dette prosjektet benyttes hovedsakelig I2C. Leseren opererer med en driftspenning på mellom 2.5 og 3.3 V. [6][8]



Figure 5: RFID-RC522 Pin-Out

3x4 Matrix Keypad

Keypad-matrisen er en trykknapp-enhet som er mer plassbesparende på pinouten til en mikrokontroller enn å bruke en inngang per trykknapp. Hver enkelt trykknapp har to brytere, der den ene leser den vertikale plasseringen og den andre leser den horisontale plasseringen. For å registrere hvilken knapp som er trykket inn, undersøker mikrokontrolleren hvilken rad og kolonne som er aktivert. Hvis den, for eksempel, registrerer rad 2 og kolonne 2 som høy, vil det tilsvare knapp nr.5. Det finnes ferdige biblioteker for denne typen keypads og den som utvikler programmet trenger derfor bare å definere rader og kolonner, i tillegg til hvilke pinner som skal definere de ulike radene og kolonnene. [9]

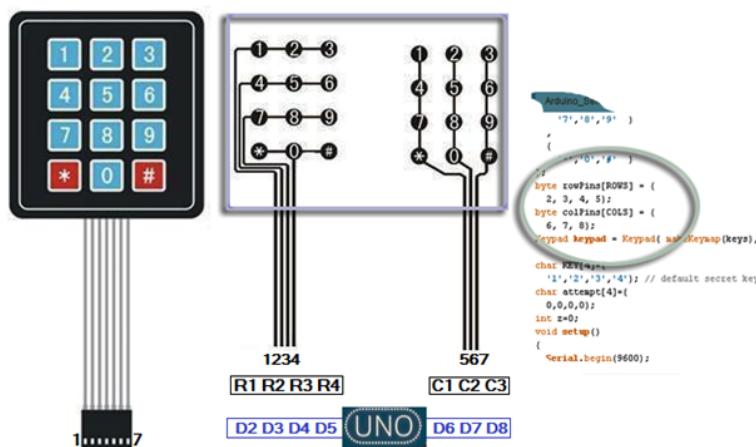


Figure 6: Matrix keypad

IR beam break

En "beam break" sensor består av to hovedkomponenter; en emitter og en receiver. For å detektere et brudd vil emitteren sende ut et infrarødt lys ved en bestemt frekvens som receiveren mottar, dersom det ikke er noen objekter i veien. Hvis f.eks en person bryter denne forbindelsen vil receiveren miste kontakten og indikerer dette på utgangspinnen sin i hendhold til databladet. Sensoren er tatt i bruk for å detektere bevegelse gjennom en dørkarm. Det er valgt å bruke en kombinasjon av TSSP4038 og VSLB3940 med bakgrunn i fornuftig driftspenning på 2.5v - 5.5v og en rekkevidde på 20cm til 2m reflekterende, i tillegg til hurtig deteksjon. Se vedlegg 3



Figure 7: IR resiver



Figure 8: IR emitter

OLED display SSD1306 er et OLED-display som kan vise skrift og bilder og som kommuniserer med SPI kommunikasjon. Den har 4 pins for kommunikasjon og 2 pins til strømtilførsel (3.3V - 5.0V). Den har en lade-pumpe-krets som tar seg av at spenningen er rett. Skjermen er et 128 x 64 bits grafisk display som er delt opp i 8 "Pages" der hver side har 128 segment som har 8 bit per segment. Ref. [10]

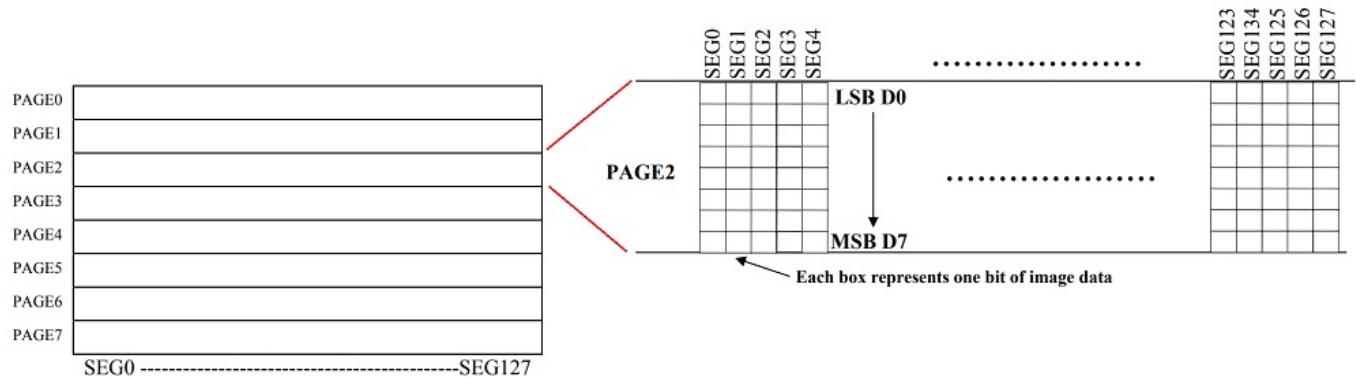


Figure 9: Binert Oppsett Oled Display

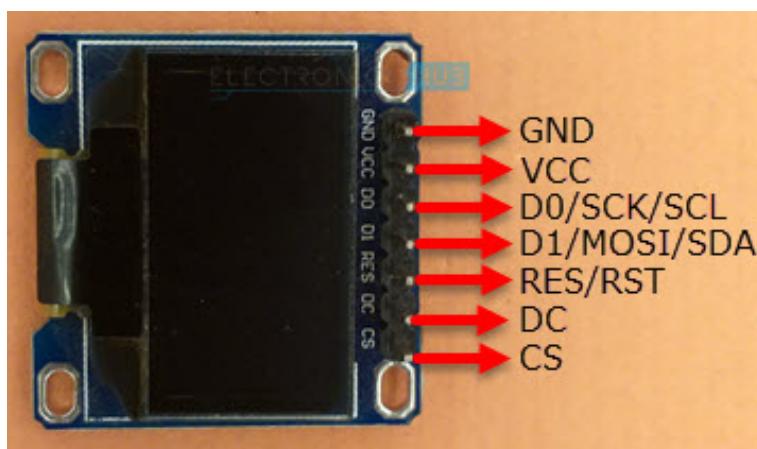


Figure 10: SSD1306 OLED Driver IC

3.2 Software

3.2.1 Programmeringsspråk

I dette prosjektet har det blitt brukt forskjellige programmeringsspråk for forskjellig hardware, for å best mulig kunne utnytte egenskapene til de forskjellige plattformene. Et språk som er effektivt med tanke på eksekusjonstid i forhold til prosessorkapasitet vil ha klare fordeler på en plattform som er begrenset av sin hardware, selv om det vil bety en mer omfattende utviklingsprosess. På enheter som derimot har stor nok kapasitet til å nyttegjøre seg av et mer komplekst programmeringsspråk, vil utviklingsarbeidet kunne reduseres, og på den måten være det beste valget.

Python

Python er et høynivå programmeringsspråk som er godt utbredt idag. Det brukes blant annet til utvikling av applikasjoner, dataanalyse, og automatisering av prosesser, for å nevne noen. Python blir sett på som et enkelt og intuitivt programmeringsspråk, i og med at syntax i språket ligger veldig nært det menneskelige språket. Python er et tolket språk, som vil si at man ikke kompilerer et helt program i forkant av gjennomkjøring, men heller tolker programmet underveis. Dette, i kombinasjon med at Python er et mindre effektivt programmeringsspråk enn f.eks C++, gjør at det krever mer prosesserkraft og dermed er lite egnet for mikrokontrollere. Likevel er Python et språk man med fordel kan bruke dersom man har prosesserkraften som kreves. I dette prosjektet vil en Raspberry Pi benyttes til dataanalyse osv., og Python vil bli brukt utstrakt til disse applikasjonene.

All utvikling av Python-program har blitt gjennomført i Spyder IDE. [3]

C++ / Arduino C

C++ og Arduino C er høynivåspråk som blir benyttet i stor skala internasjonalt. Mens C++ blir brukt i profesjonelle sammenhenger, er Arduino C et språk som er basert på C++, men på mange måter forenklet for å treffe hobby- og utdanningsmarkedet. Felles for begge er at de i motsetning til Python er kompliserte programmeringsspråk. Det vil si at hele programmet blir gjort om til maskinkode før programmet kjører, i stedet for å tolke linje for linje. Ved utvikling i C++, importeres Arduino-biblioteket for enkel styring av periferienhetene til mikrokontrolleren. Det finnes i tillegg en rekke open-source biblioteker tilgjengelig som dekker spesifikke funksjoner. Eksempelvis finnes det biblioteker tiltenkt enkel styring av servomotorer, TFT-display, tidssyknronisering osv. Disse bibliotekene er stadig under utvikling, men mange anerkjente og vel-establerte biblioteker har oppnådd en kvalitet som gjør de til en viktig ressurs under utvikling av program.

Med tanke på valg av IDE har prosjektgruppa vert delt i sin oppfatning av hva som er det beste alternativet. Enkelte utviklere har valgt å bruke Arduino IDE til utvikling, mens andre følte seg begrenset av dette IDEet og valgte å bruke Microsoft Visual Studio Code med utvidelsen Platform IO for kommunikasjon med mikrokontrollerene.

3.2.2 Kommunikasjon

MQTT

MQTT er en av mange protokoller som benyttes til kommunikasjon i IoT-applikasjoner og systemet gjør det enkelt og effektivt å lese og skrive meldinger mellom ulike enheter. Ved å programmere enhetene til å lete etter meldinger samme sted, oppnår man at hver enkelt enhet ikke må lete gjennom store mengder data for å finne hvilke meldinger som er ment for den. Valget falt på MQTT fordi protokollen for dataoverføring er "lightweight" og effektiv og dermed godt egnet i IoT-systemer. MQTT bruker en asynkron kommunikasjonsprotokoll som vil si at data sendes periodisk og ikke i en jevn strøm. Dette gjør MQTT ideell til bruk på steder hvor nettverkssignalet er svakt eller ustabilt.

Istedentfor at dataen sendes i meldingskøer, bruker MQTT en publish- og subscribe-modell. Klienten, som i dette tilfellet er de ulike konsollene hver hybel og beboer har, sender data under et "topic" til en "broker" ved å publisere. Serveren (Rpi) kan koble seg til samme broker og subscribe til samme topic. Den har dermed mulighet til å få instruksjoner fra mikrokontroller, som avgjør hvilke oppgaver som skal kjøres. Ref. [18]

Figure 1. The MQTT publish and subscribe model for IoT sensors

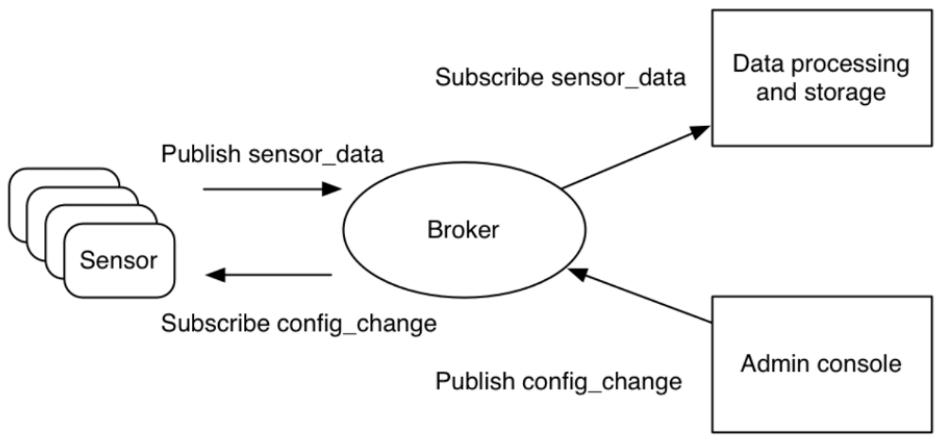


Figure 11: MQTT kommunikasjonsmodell

På tross av MQTT sine avanserte funksjonaliteter, er det et lett og lite program. MQTT kan overføre en rekke forskjellige data-format, eksempelvis JSON, XML eller Base 64, der det eneste kriteriet er at MQTT-klientene kan analysere "payloaden". Ulempen ved å benytte MQTT er at man får en del overhead, men den er likevel rask og effektiv i en konfigurasjon med mikrokontrollere og en Raspberry Pi server. Ref. [19]

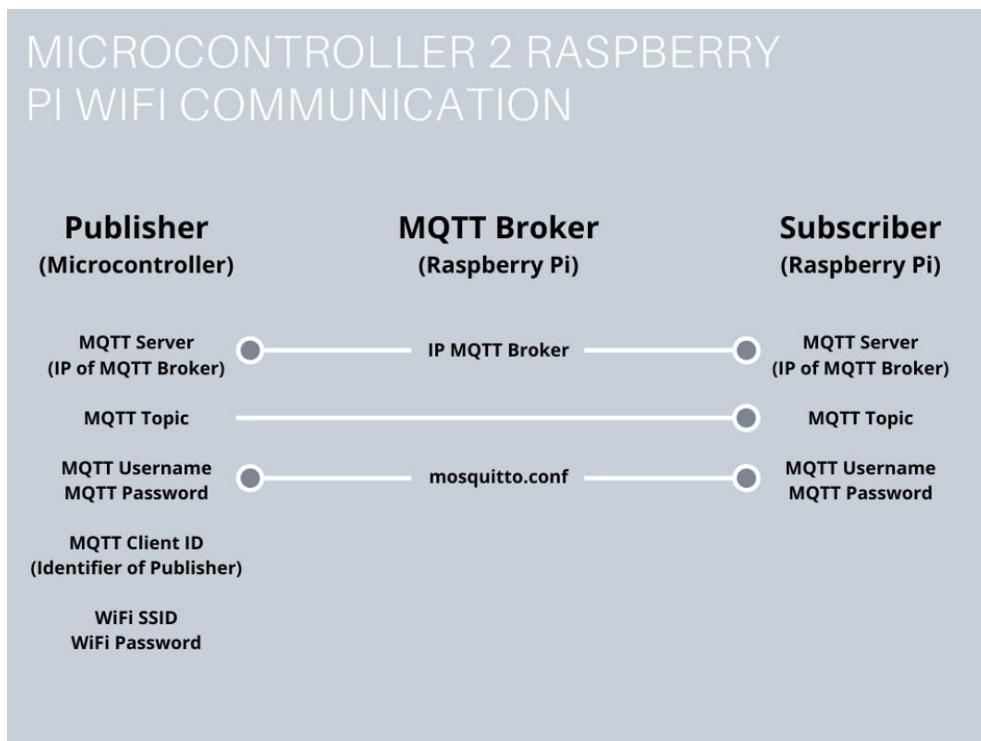


Figure 12: MQTT - Kommunikasjonsmodell

CoT

Circus of Things er en IoT-platform for ekstern datautveksling og datalogging. Det er en «open-source» tjeneste, som betyr at den er tilgjengelig for alle. Nettsiden kan brukes til dataovervåking, systemstyring, utdanning, for å nevne noen. CoT er basert på "signal" som involverer et register som lagrer en kjøretidsverdi som blir sendt fra et produkt og evt. lest av et annet. Under "workshop" i CoT kan en bruker eller utvikler lage, endre og slette signaler. Dette kan for ekspemel være en av og på knapp, temperaturvisning osv. Etter at signaler er opprettet vil det være mulig å betjene disse under "dashboard". [?]

3.2.3 Kryptering

Hash

HASH er en stringbasert kode. Det blir generert av en "salt" eller en nøkkelverdi, som f. eks "f1nd1ngn3m0" og en valgt string-verdi som brukes til kode eller passord. Eksempel: 1234 + salt (f1nd1ngn3m0) -> 1234.f1nd1ngn3m0). Ut i fra hvilken salt du har vil den samme koden/passordet bli helt likt hver gang

Ekesmpel:

1234.f1nd1ngn3mo -> Hash: 07dbb6e6832da0841dd79701200e4b179f1a94a7b3dd26f61281

123b7.f1nd1ngn3mo -> Hash: 11c150eb6c1b776f390be60a0a5933a2a2f8c0a0ce766ed92fea

1234.et52ed4556jgg -> Hash: 72ae25495a7981c40622d49f9a52e4f1565c90f048f59027bd9c

En "hash" har alltid et fast antall bit (f. eks: 32 bit). Derfor har alle hashene helt lik form, unnsett hvor lang salten eller passordene er. Denne formen oppnåes ved bruk av en matematisk algoritme på koden + salten. Sikkerhetsgraden kan økes ved å kjøre denne matematiske algoritmen på produktet av første hash-generering, noe som vil gjør den vesentlig vanskeligere å dekryptere (krever mer datakraft). På en vanlig nettside idag, kjøres denne algoritmen typisk 15 ganger, for å være trygg. Denen teknologien blir også brukt i f.eks blockchain og kryptovaluta.[\[4\]](#)

SHA-2 (SHA-256)

SHA-2 er krypteringsmetode som ble opprettet i USA av NASA i 2001. Se "Hash" for konstruksjon, da de er relativt like i sin oppbygning. Forskjellen ligger i at SHA-2 benytter enveis kompresjon, som betyr at den originale koden aldri vil kunne rekonstrueres. Det vil si at dersom noen avbryter og fanger opp HASH-koden, vil de ikke kunne finne ut hvilken kode som opprinnelig ble sendt før "salten" ble lagt til. Selv om man har "salten" vil man ikke kunne gjenskape den originale koden. I dette prosjektet ble SHA-256 benyttet, der 256 betyr at det benyttes en 8 segments kode med 32 bit

$$n(32 \cdot 8 = 256)$$

SHA-256 er samme kryptering som er brukt av Bitcoin og flere lignende kryptovaluta.[\[13\]](#)[\[4\]](#)

Openweathermap

Openweathermap er et onlinebasert selskap som distribuerer værdata via API. Selskapet tilbyr mange forskjellige tjenester innen værdata for alle geografiske plasseringer. Tjenesten kan brukes av hvem som helst og er kostnadsfri. [\[17\]](#)

4 Metode

I oppstarten av prosjektet ble prosjektgruppens fokus rettet mot idemyldring, overordnede diskusjoner rundt hvordan funksjonene som var listet i funksjonsbeskrivelsen kunne implementeres på en fornuftig måte, hvordan de forskjellige egenskapene til systemet skulle fordeles på forskjellige enheter, og i grove trekk hvordan den overordnede kommunikasjonssstrukturen for totalløsningen skulle være utformet. Løsningen som var foreslått i funksjonsbeskrivelsen som fulgte prosjektet ble tidlig ansett som upraktisk. Funksjonsbeskrivelsen gav utsyn for et ønske om å implementere alle funksjoner i én håndholdt kontroller, men etter interne diskusjoner kom det frem en løsning som delvis avviker fra funksjonsbeskrivelsen.

Å styre alle funksjoner på hvert enkelt soverom fra en personlig, bærbar håndkontroller ble ansett som å være et potensielt større irritasjonsmoment for en sluttbruker enn en fordel. Flere av utviklerene i prosjektgruppa har tidligere vert utøvende montører i elektrobransjen og opplevd at enkle løsninger for faste installasjoner ofte er det mest brukervennlige. Tilbakemeldinger fra kunder som har valgt smarthus-løsninger som både er mulige å styre via app og via enkle betjeningspanel, er at appen normalt ikke blir brukt fordi det ofte blir for tungvint. Med et fast betjeningspanel for å styre varme, lys, vifte osv. på et rom, i stedet for å ha en bærbar smart-enhet, vil en kunde ha et fast referansepunkt for hvor innstillingene for rommet kan justeres, og man unngår i tillegg problematikken med at dersom en bruker f.eks mister eller ødelegger den bærbare enheten, kan han dermed heller ikke kan skru på og av lyset på et rom.

Det ble istedenfor vurdert at den beste totalløsingen ville oppnås ved å utvikle tre forskjellige enhetstyper:

- En bærbar håndkontroller for booking av rom og ressurser, i tillegg til å svare og styre inngangsdøra dersom det skulle ringe på.
- En fastmontert romkontroller som styrer lys, varme og lufting på hvert enkelt rom via et integrert betjeningspanel
- En fastmontert adgangskontrollenhet der beboere kan komme seg inn og ut av leiligheten og gjester kan ringe på til de respektive beboerne.

For å fordele arbeidsmengden, ble prosjektgruppen delt inn i tre, der hver undergruppe fikk hovedansvaret for utvikling av en av de tre overnevnte kontrollerene. I og med at det ikke var helt tydelig hvor mye arbeid de enkelte kontrollerene ville innbefatte, ble det bestemt at den gruppen som kom i mål med sin kontroller først skulle ta ansvar for utvikling av program for strømbudsjett, beregning av produksjon fra solceller osv.

Prosjektgruppen har gjennom hele utviklingsperioden hatt faste møter hver onsdag og fredag, der det har blitt samarbeidet om problemer, diskutert rundt forskjellige løsninger,

og rapportert fremdrift på de forskjellige underprosjektene. For å oppnå en mest mulig presis rapport, ble det også bestemt at hver enkelt undergruppe skriver om sitt system i den endelige rapporten.

For å ha muligheten til å gå inn og se på de andre undergruppene sin software, i tillegg til å ha en sentralisert backup av all software som har blitt utviklet, ble det opprettet et prosjekt i GitHub der hvert enkelt gruppemedlem lastet opp sitt program jevnlig. Videre ble det opprettet et loggdokument, delt mellom alle gruppemedlemmene, som ble brukt til huskelister, bestemmelser, møtereferat, felles beskjeder osv.

Før oppstart av softwareutvikling ble det fastsatt en standard for hvordan programmene skulle skrives. For å gjøre software mer lettleselig, og for at programmene skulle fremstå som mer ryddige, ble det bestemt at all utvikling skulle foregå på engelsk. Videre ble det også oppfordret til å bruke snake case ved navngiving av variabler. Alle variabelnavn skulle være enkle, men forklarende, og programmene skulle være godt dokumenterte i form av kommentarer innad i programmene.

4.1 Romkontroller - Varme, lys, lufting, takvifte

Ettersom det ble vurdert som mest hensiktsmessig å ikke implementere alle funksjoner i en håndholdt enhet, ble det utviklet en romkontroller som styrer alle funksjoner som er knyttet til regulering av lys, varme og lufting på hvert enkelt rom. Romkontrolleren baserer seg på en Espressif Systems ESP32 mikrokontroller og er utviklet med mål om å kunne være en del av den faste elektriske installasjonen i et bygg. Produksjonsmodellen vil ha fysiske mål i størrelsesordenen til en standard veggtermostat slik at den kan monteres i en standard Ø = 71mm veggboks. Slik kan den også ettermonteres i eksisterende elektriske anlegg, inkluderes i eksisterende bryterpanel osv. uten behov for videre behov for rehabilitering av rommet.



Figure 13: Romkontroller - Produksjonsmodell

4.1.1 Prototype

Det ble tidlig i planleggingsfasen bestemt hvilke funksjoner romkontrolleren skulle ha, og den første prototypen på hardwareoppsettet var klar allerede første utviklingsuke. Etterhvert som mer og mer software ble utviklet, dukket det stadig opp hardware-relaterte problem og hardware-konfigurasjonen ble derfor revidert jevnlig. Eksempelvis ble det oppdaget at ved bruk av WiFi-funksjonen til mikrokontrolleren, ble ADC2 opptatt med å serve dette, og hardwarepinnene på utviklingskortet tom var tilknyttet denne ADC'en kunne derfor ikke benyttes som analoginnganger.

4.1.2 Programvareutvikling

Allerede før programvareutviklingen startet var det tydelig at dette ville bli et komplekst prosjekt, og at Arduino IDE, som er rettet mot hobbymarkedet, ville bli for begrensende til å kunne utvikle programvaren effektivt. All programvareutvikling for romkontrolleren ble derfor gjort i Microsoft Visual Studio Code med utvidelsesmodulen Platform IO for kommunikasjon med mikrokontrolleren.

For å unngå å fylle hovedprogrammet med utallige funksjonsdefinisjoner, ble det opprettet separate .cpp og tilhørende .h-filer for de fleste funksjonene som ble brukt. Funksjonene ble erklært i .h-filene og definerte i .cpp-filene. Funksjonene ble videre kategorisert etter hovedområde og fordelt i respektive filpar. Eksempelvis ble det opprettet et filpar "Controller config" der hardwarekonstanter ble definert, hardwaresetup-funksjon ble definert osv., og et filpar "OLED" der alle funksjoner knyttet til displayet ble definerte. Headerfilene ble så inkluderte i hovedprogrammet, og hovedprogrammet kunne dermed kalle alle funksjonene som hadde blitt definerte i de eksterne filene.

Display

Displayet som ble valgt var et TFT LCD display som kommuniserte med kontrolleren via SPI. Den britiske utvikleren som går under kallenavnet Bodmer, har utviklet et bibliotek (TFT eSPI) [21] som gjør bruk av disse displayene relativt brukervennlig, men det kreves likevel omfattende etterarbeid for å skape de skjermbildene som er ønsket.

Det ble i dette prosjektet definert fleksible funksjoner for generering av forskjellige verdijusterings-vinduer, informasjonsvinduer, menynavigasjonevinduer osv. Disse funksjonene ble definerte i OLED-filen og kalt opp i hovedprogrammet etter behov.



Figure 14: LCD-display

Menysystem

For å enkelt kunne betjene alle funksjoner som skulle inkluderes i romkontrolleren, ble det tidlig bestemt at et meny-navigasjonssystem måtte etableres. Et adresseringssystem ble derfor definert og implementert i kontrollerprogrammet. Navigasjon i menysystemet gjøres ved hjelp av de integrerte trykknappene på romkontrolleren. Som visualisert i flytskjemaet for menysystemet, er det delt inn i forskjellige nivå. Det første tallet i den gjeldende meny-adressen representerer hvilket nivå i menyen brukeren til enhver tid har aktivert. De følgende verdiene i adressen representerer hvilken verdi som er valgt for de forskjellige nivåene. Formatet for adressering er følgende:

[Aktivt nivånummer , Nivå 1-verdi , Nivå 2-verdi , Nivå 3-verdi ... osv.]

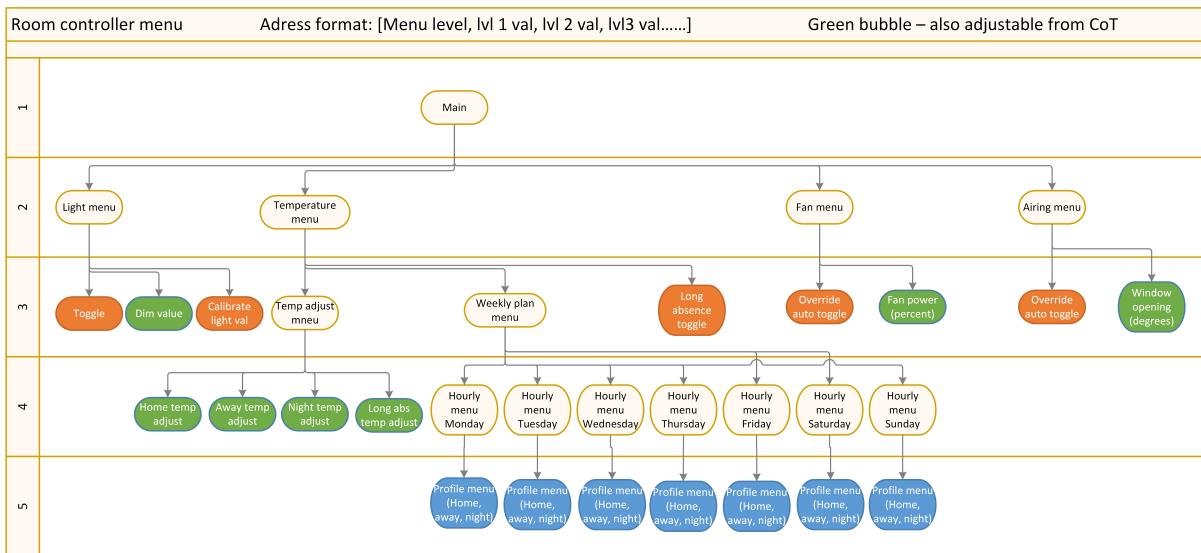


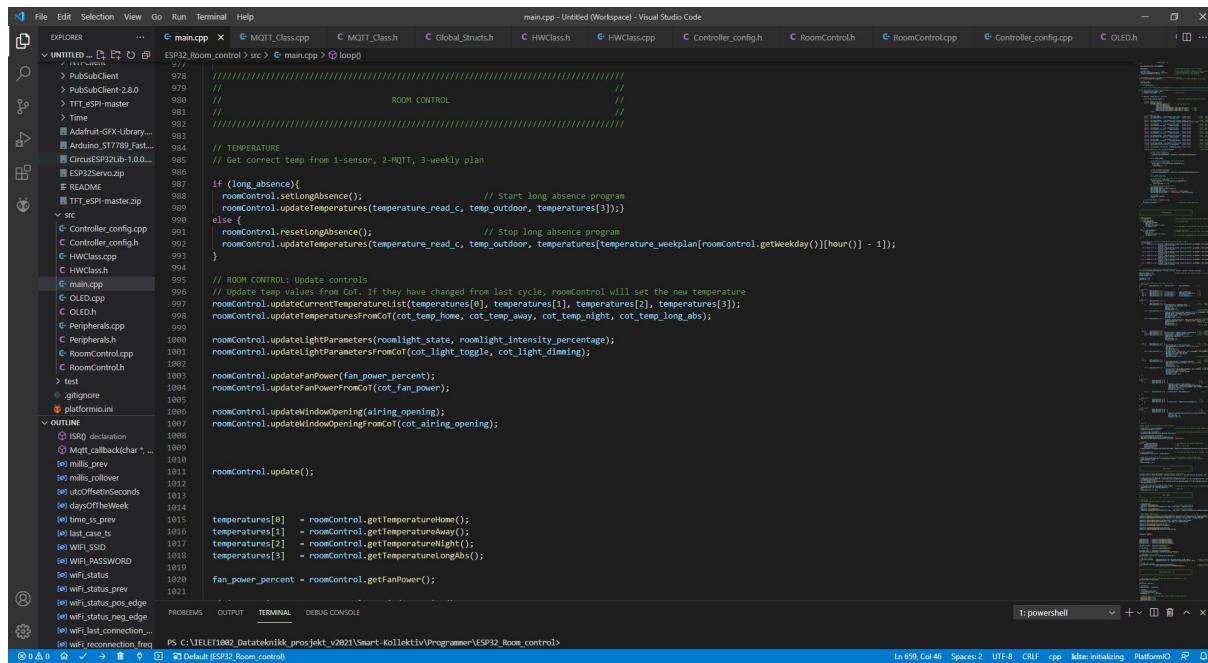
Figure 15: Romkontroller - Menysystem

Dersom brukeren befinner seg på nivå 1, blir nivå 1-verdi endret opp eller ned. Dersom brukeren går til neste nivå, vil nivå 2-verdien bli endret opp eller ned. Denne kjeden av enkeltnivåverdier gir til enhver tid hvilken "path" en bruker har navigert seg gjennom menyen, og kan dermed bestemme hva som til enhver tid skal utføres eller justeres basert på brukeren sin input. I de øverste nivåene av menyen navigerer brukeren mellom forskjellige lister av kategorier. I de nederste nivåene ligger det enkeltfunksjoner for justering av parametere, toggling av forskjellig funksjonalitet osv.

Romkontroll-regulering

Før programmering av selve romkontroller-reguleringen startet, ble det investert tid i studie av objektorientert programmering (OOP), for å ha flere verktøy tilgjengelig under utvikling av romkontroll-reguleringen. Det ble opprettet en "RoomControl"-klasse der en rekke funksjoner muliggjorde interaksjon mellom hovedprogrammet og det instansierte objektet av klassen. RoomControl-klassen ble opprettet for å utføre en rekke oppgaver. Først og fremst ble klassen laget for å regulere innetemperatur ved hjelp av varmepådrag fra en panelovn, lufting gjennom vinduet og pådrag på takviften. Reguleringen ble utformet for å jobbe trinnvis etter en forhåndsbestemt hysterese. I følgende eksempel er ønsket temperatur satt til $T = 20$ grader C. Styringen er programmert til å oppføre seg som følgende:

- $T > 27 \rightarrow$ Panelovn Av - Vindusåpning 60 gra - Takvifte 100%
- $T > 25 \rightarrow$ Panelovn Av - Vindusåpning 40 gra - Takvifte 60%
- $T > 23 \rightarrow$ Panelovn Av - Vindusåpning 20 gra - Takvifte 40%
- $T > 21 \rightarrow$ Panelovn Av - Vindusåpning 0 gra - Takvifte 30%
- $T < 19 \rightarrow$ Panelovn På - Vindusåpning 0 gra - Takvifte 30%



```

1 // ROOM CONTROL
2
3 // Get correct temp from 1-sensor, 2-MQTT, 3-weekly plan
4
5 if (long_absence){
6     roomControl.setLongAbsence(); // Start long absence program
7     roomControl.updateTemperatures(temperature_read_c, temp_outdoor, temperatures[3]);
8 } else {
9     roomControl.resetLongAbsence(); // Stop long absence program
10    roomControl.updateTemperatures(temperature_read_c, temp_outdoor, temperatures[temperature_weekplan[roomControl.getWeekday()][hour()] - 1]);
11 }
12
13 // ROOM CONTROL: Update controls
14 //Update temp values from CoT if they have changed from last cycle. roomControl will set the new temperature
15 roomControl.updateCurrentTemperatureList(temperatures[0], temperatures[1], temperatures[2], temperatures[3]);
16 roomControl.updateTemperaturesFromCoT(cot_temp_home, cot_temp_away, cot_temp_night, cot_temp_long_abs);
17
18 roomControl.updateLightParameters(roomlight_state, roomlight_intensity_percentage);
19 roomControl.updateLightParametersFromCoT(cot_light_toggle, cot_light_dimming);
20
21 roomControl.updateFanPower(fan_power_percent);
22 roomControl.updateFanPowerFromCoT(cot_fan_power);
23
24 roomControl.updateWindowOpening(alarm_opening);
25 roomControl.updateWindowOpeningFromCoT(cot_airing_opening);
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
917
918
919
919
920
921
922
923
924
925
926
927
927
928
929
929
930
931
932
933
934
935
936
937
937
938
939
939
940
941
942
943
944
945
945
946
947
947
948
949
949
950
951
952
953
953
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1630
1631
1631
1632
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1640
1641
1641
1642
1642
1643
1643
1644
1644
1645
1645
1646
1646
1647
1647
1648
```

Introduksjon til Button-klassen

```

1 // Dokumentasjon for bruk av Button-klassen
2
3 Button up; // Instansiering av objekt "opp"
4 Button down; // Instansiering av objekt "ned"
5
6 // Oppdatering av trykknappstatus (fra hardware)
7 up.setState(bool state);
8
9 // Returnerer naverende trykknappstatus
10 up.readState();
11 // Returnerer hoy verdi ved positiv flanketrigging
12 up.posEdge();
13 // Returnerer timestamp for siste positive flanketrigging
14 up.posEdgeTS();
15 // Returnerer hoy verdi ved negativ flanketrigging
16 up.negEdge();
17 // Returnerer timestamp for siste negative flanketrigging
18 up.negEdgeTS();
19 // Returnerer hoy verdi x millisekund etter trykknappstatus har
   fattet hoy verdi.
20 up.onDelayMS(int ms)
21 // Returnerer hoy verdi x sekund etter trykknappstatus har fattet
   hoy verdi.
22 up.onDelayS(int s)
23 // Returnerer hoy verdi i x millisekund etter trykknappstatus har fattet
   lav verdi.
24 up.offDelayMS(int ms)
25 // Returnerer hoy verdi i x sekund etter trykknappstatus har fattet
   lav verdi.
26 up.offDelayS(int s)

```

```

139 // BUTTON: Initialize button objects
140 Button up;
141 Button down;
142 Button left;
143 Button right;

```

```

182 // Update current button statuses
183 up.setState(digitalRead(pin_up));
184 down.setState(digitalRead(pin_dwn));
185 left.setState(digitalRead(pin_lft));
186 right.setState(digitalRead(pin_rgt));

```

Figure 18: Instansiering av objektet

Figure 19: Oppdatering av trykknappstatus

```

888 // Menu level 4
889 else if (menu_adress[0] == 4) {
890     if (menu_adress[1] == 2) {
891         if (menu_adress[2] == 1) {
892             temperature_profile = menu_adress[3] - 1;
893             temperature_set = temperatures[temperature_profile];
894             temperature_set = mod_temp(up.posEdge(), down.posEdge(), temperature_set);
895             temp_adjust = HIGH;
896             deep_end_reached = HIGH;

```

Figure 20: Bruk av posEdge-funksjon

Lysregulering

Et annet ENØK-tiltak som ble implementert var selv-regulerende lys i det stykte rommet. Styringen ble laget slik at en gitt lysintensitet skal opprettholdes i et gitt rom uansett dagslysinnslipp. I praksis betyr det at midt på dag, når dagslysinnslippet er høyt, vil lyskilden i rommet være svak eller avslått, men etterhvert som mindre dagslys slipper inn, regulerer styringen på med mer kunstig lys for å opprettholde lysintensiteten. Referansepunkt for ønsket lysmengde kan til enhver tid kalibreres av brukeren ved å velge lyskalibrering i menyen. Romkontrolleren setter da nåværende lysmengde som referanse.

I tillegg til automatisk lys-regulering, skulle lyset kunne skrues av og på og dimmes ned av brukeren etter ønske, enten direkte på romkontrolleren eller gjennom CoT.

Tidssynkronisering

I og med at romkontrolleren opererer time for time etter en definert ukeplan, er det kritisk at den interne klokka til mikrokontrolleren er korrekt. Ved oppstart av romkontrolleren, må kontrolleren derfor synkronisere sin interne klokke og kalender mot en Europeisk NTP-server, gitt at den er tilkoblet trådløst nettverk. Dersom trådløst nettverk ikke er tilgjengelig ved oppstart, må den interne klokken oppdateres med en gang enheten kobles til trådløsnett. Mikrokontrolleren har en relativt korrekt intern klokke, så når klokke og kalender først er synkronisert, vil det i utgangspunktet ikke være behov for svært hyppig synkronisering. I dette programmet ble det likevel lagt opp til at klokke og kalender synkroniseres ca to ganger i minuttet som en del av bakgrunnsprosessene som skal kjøres i skjermsparermodus.

Oppstartsprosedyre

Ved oppstart av enheten må den gå gjennom en komplett setup-prosedyre. Først konfigureres hardware og seriell kommunikasjon. For videre må programmet vite hvilket rom kontrolleren skal settes opp for, så de ble derfor lagt inn en meny der gjeldende rom kan velges. Dette blir blant annet gjort for å kunne sende personlig forbruksdata til server og for at MQTT-kommunikasjon skal komme frem til korrekt mottaker. Når korrekt rom er valgt, vil romkontrolleren forsøke å koble til trådløst nettverk. Dersom den klarer å koble til trådløsnett, vil den så sette opp kommunikasjon med CoT, sette opp kommunikasjon til NTP-servere og synkronisere den interne klokka og sette opp kommunikasjon med MQTT-server. Dersom trådløsnett ikke er tilgjengelig, vil dette utsettes til enheten oppnår kontakt med trådløst nettverk. For hver operasjon romkontrolleren gjør under setup, blir displayet oppdatert med informasjon til brukeren, slik at brukeren har kontroll over hvilke problemer som evt. dukker opp ved oppstart og kan gjøre tiltak for å utbedre dette.



Figure 21: Setup

Skjermsparer og bakgrunnsprosedyrer

Mens en bruker navigerer i meny-systemet til romkontrolleren, er han avhengig av umiddelbar respons for at enheten ikke skal oppleves som "treg". Som et resultat av at blant annet kommunikasjon med CoT sine servere er veldig tidskrevende prosesser, der lesing av et enkelt signal kan resultere i en forsinkelse på 1-2 sekunder, ble det bestemt at disse prosessene bare skal kunne kjøre dersom skjermsparer er aktivert og brukeren ikke er avhengig av hurtig respons. Der er derfor allokkert en egen seksjon i programmet for prosesser som bare skal kjøre dersom skjermsparer er aktivert og trådløsnett er tilkoblet. Eksempel på prosesser som kjører her er kommunikasjon med CoT, MQTT og synkronisering av klokke mot NTP.

Når skjermspareren er aktivert, skal et egent vindu komme opp på displayet til romkontrolleren, der dag og dato viser konstant, og det siste feltet skal veksle mellom innendørstemperatur og klokkeslett med 10 sekunds intervall.

I og med at romkontrolleren er opptatt med bakgrunnsprosesser mens skjermsparer er aktivert, ville den normalt ikke reagert på knappetrykk, med mindre knappetrykket skjedde akkurat i begynnelsen av en program-scan. For å være sikker på at alle knappetrykk som skal "vekke" enheten fra skjermsparermodus blir detekterte, blir alle trykknapper linket til en ISR som gir hovedprogrammet beskjed om at skjermsparermodus skal deaktivieres, når enheten går inn i skjermsparermodus. Når enheten ikke lenger er i skjermsparermodus, fjernes avbrudds-triggingen av trykknappene og normal funksjon gjenopptas.

Langtidsfravær

Som supplement til den normale temperaturreguleringen som funksjon av ønsket temperatur gitt av ukeplan og utendørstemperatur, har romkontrolleren et modus for langtidsfravær. Langtidsfraværstemperaturen er som standard satt lavt og vil i praksis fungere som frostsikring ved langtidsfravær. Denne temperaturen kan selvsagt justeres av bruker.

Ved aktivering av langtidsfravær vil automatisk styring av takvifte og luftevindu skrues av og det er bare panelovnen som vil sørge for frostsikring av rommet. Grunnen til dette er at takvifte kan være en kilde til brannfare, og når rommet ikke skal brukes er dette en unødvendig risiko. Vinduet skal i dette modiet holdes lukket for innbruddssikring.

Strømsparingsmodus

I og med at romkontrolleren til enhver tid skal vere tilkoblet nettspenning, ble det ansett som en større ulempe enn fordel å implementere strømsparingsmodus. For at enheten skal fungere optimalt er den avhengig av å til enhver tid vere aktiv for å kunne regulere lys, styre temperatur og lufting, og være tilkoblet nettverk for å kunne oppdatere verdier fra blant annet CoT



Figure 22: Skjermsparer

4.1.3 CoT

Etter at prosjektet ble satt igang etablerte samtlige prosjektgrupper kommunikasjon med CoT. Når utviklerene begynte å teste kommunikasjonen og oppdaterte sine verdier til CoT med samme frekvens som arbeidssyklusen til en ESP32, opplevde CoT dette som et DDoS-angrep og serverene ble overbelastede. CoT sin løsning var å legge inn en tidsforsinkelse på ca et sekund for all signaloppdatering. Dette resulterte i at CoT som tjjeneste ble vesentlig mindre responsiv og anvendelig enn ønskelig og tjenesten ble til slutt bare brukt til visualisering av parametere og justeringer av daglige innstillingar da hastigheten gjorde det uaktuelt å bruke det til all kommunikasjon. All internkommunikasjon mellom de forskjellige enhetene ble derfor lagt over til MQTT.

Et dashboard ble utformet i CoT for å overvåke og justere følgende verdier:

- Justering av de fire temperatur-statusene: Home, Away, Night og Long Absence
- Utlesing av innendørstemperatur
- Betjening av lys, samt dimming
- Justering av takvifteeffekt for manuell modus
- Justering av vindusåpning i grader for manuell modus

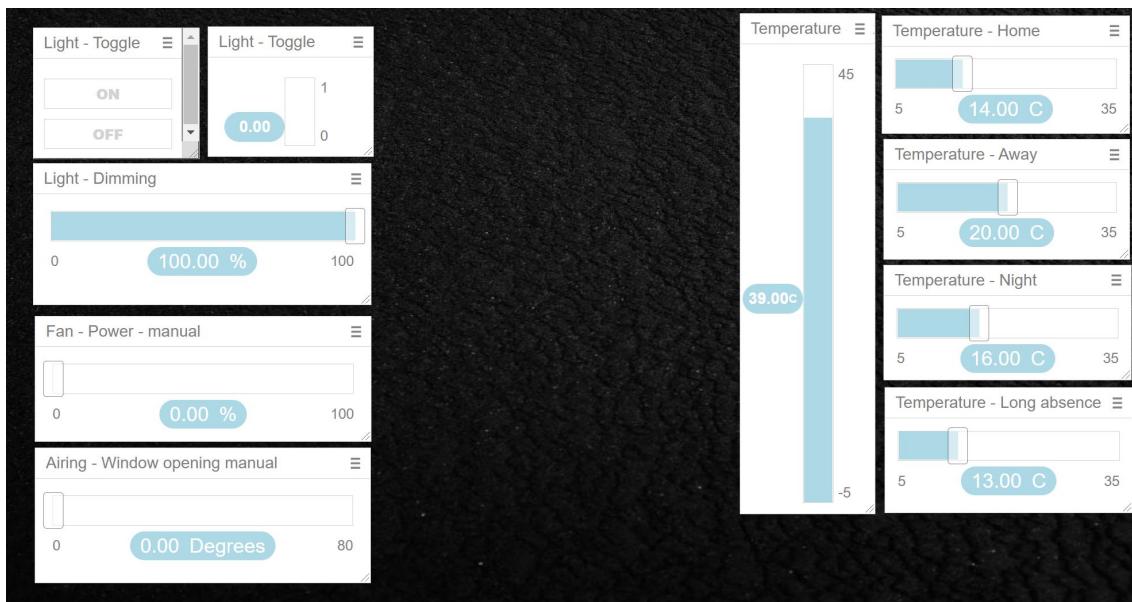


Figure 23: Dashboard SCADA - Romkontroller

4.2 Håndkontroller - Booking, dørklokke

4.2.1 Booking

Ved oppstart av utvikling av bookingsystemet ble flere forskjellige løsninger vurderte. Det ble bestemt at statusen til de ulike rommene skulle lagres i serveren (RPi), og dermed måtte systemet for bookingen programmeres i Python. Den første tilnærmingen til systemet var å bruke dictionaries. Dette er en måte å lagre data på i Python, hvor verdier lagres i par av "key:value". Ved bruk av dictionaries er det lett å endre en "key" til en ønsket verdi. Tankegangen var å sette opp en dictionary for hver av de tre rommene som er mulig å booke, kjøkken, bad og stue. Dermed skal hovedprogrammet sjekke om det er ledig eller ikke. Utgangspunktet er at alle rommene er ledige. Dictionariene ser ut som på figur til høyre:

Dette ble ansett som et godt utgangspunkt for systemet og det ble fort utarbeidet et eksempel på hvordan programmet kunne se ut og fungere. Det var flere deler som måtte utvikles og testes hver før vi fikk satt sammen alt til et felles script som fungerte slik vi ønsket. Første del var å lage selve bookingen. Dette gikk ut på at hovedfunksjonen for booking skulle ta inn en verdi, som avgjorde hvilket rom som skulle bookes. Deretter skulle programmet sjekke hvorvidt rommet som ble etterspurt var ledig eller ikke. For å undersøke at systemet hadde ønsket funksjonalitet, ble det definert en test-variabel som bookingfunksjonen skulle ta inn og deretter sjekke statusen på rommet.

Neste steg var å lage en tidsfunksjon som hadde kontroll på hvor lenge man kunne booke de ulike rommene, og samtidig hvor lenge en bruker hadde igjen av den bookede tiden sin. Vi lagde en funksjon som telte ned fra en gitt verdi og fungerte på lik linje med et nedtellingsur, da den telte nedover og viste hvor lang tid det var igjen. Funksjon ga derimot flere problemer enn løsninger. Problemet var at mens denne kjørte, okkuperte den hele scriptet, slik at man ikke fikk gjort annet mens den kjørte. Dersom kjøkkenet ble booket for 30 minutter, fikk ikke noen andre booket før klokken hadde telt ned til 0. Vi undersøkte mulige løsninger, hvor "threading" ble sett på som et mulig alternativ. Threading er en modul som tillater at flere oppgaver i samme program kjører samtidig. Tanken var at dette kunne tillate å kjøre tidsfunksjonen i bakgrunnen, samtidig som det fremdeles var mulighet til å legge inn nye bookinger. Derimot så var ikke dette pensum relatert og fremsto som unødvendig vanskelig. Av den grunn ble tidsfunksjonen skrotet, og erstattet med bruk av millis. Millis sørget for at vi oppnådde ønsket funksjonalitet av programmet, uten store problemer.

```

kitchen = {
    'slot_1': 'free',
    'slot_2': 'free'
}

bathroom = {
    'slot': 'free'
}

livingroom = {
    'slot_1': 'free',
    'slot_2': 'free',
    'slot_3': 'free'
}

```

Figure 24: Booking dictionaries - Python

Samtidig som programmet for bookingsystemet i Python ble utarbeidet, var det også nødvendig å sette opp et script for hvordan håndkontrolleren skulle fungere. Hvert medlem i husstanden skulle ha sin egen håndkontroller som de skulle kunne bruke til å booke de ulike rommene med. Håndkontrolleren var basert på en ESP32 mikrokontroller, og dermed måtte dette programmet utvikles i C++ / Arduino C. Arduino IDE ble brukt til utviklingen. Ideen var at det skulle være mulig å bla gjennom en meny med de ulike rommene, og deretter velge om det skal bookes en plass eller kansellere en eksisterende booking. Det første vi gjorde var å sette opp en "switch case"-statement som skulle velge hva som skulle bookes. Dette tillater å sette opp flere og forskjellige tilstander i programmet, som hver har sin egen oppgave. Planen var at et knappetrykk skulle aktivere for eksempel case for kjøkkenet, og da skulle ESP32en sende en verdi til bookingsystemet i Python som sa at brukeren ønsket å booke kjøkkenet.

Derimot så skulle håndkontrolleren komme med en LCD-skjerm, hvor man kunne navigere gjennom menyer for booking og kansellering av de ulike rommene. Dette er en helt annen kodeprosess enn det vanlig switch case tillater, derfor ble switch-casen utelatt. Displaybiblioteket ble først utarbeidet av Jørgen Mo for bruk på romkontrolleren, og deretter ble det implementert på håndkontrolleren til å fungere med bookingen. Funksjonaliteten til displayet er at det blir satt opp ulike nivåer, hvor et nivå ned vil gi nye valgmuligheter. På denne måten er det mulig å navigere frem til den handlingen som er ønsket av brukeren. Når man har kommet til siste nivå og velger for eksempel "book kitchen", vil ESP32en sende en verdi til Server, som ber bookingsystemet sjekke om det er ledig på kjøkkenet.

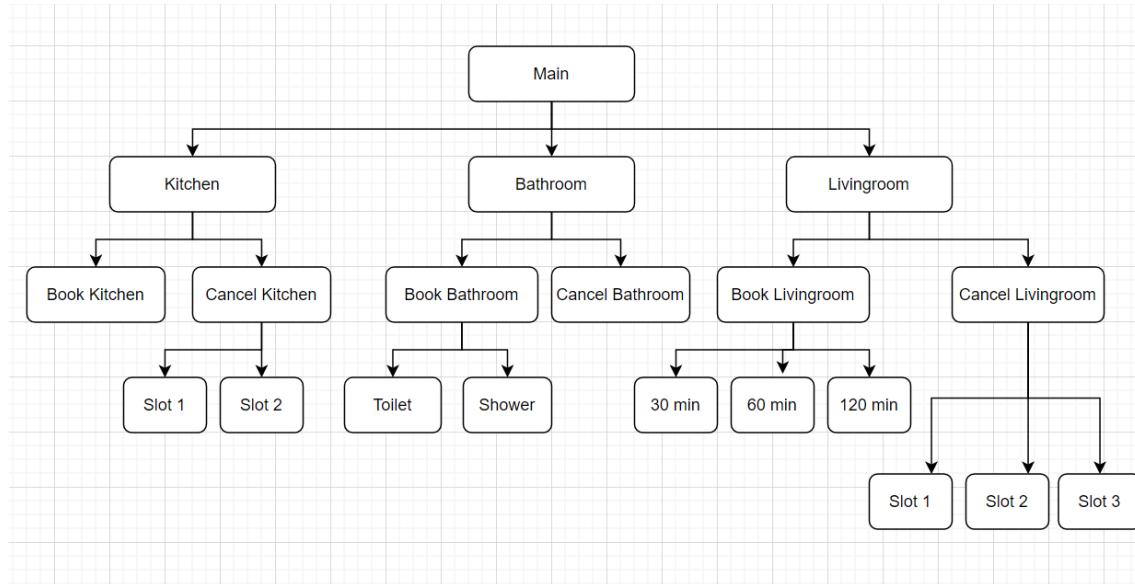


Figure 25: Flytskjema for menynavigasjon på håndkontroller.

For at det skulle være mulig å sende en verdi fra håndkontrolleren til serveren, måtte vi opprette kommunikasjon mellom disse to enhetene. Dette ble gjort via MQTT. Bruk av MQTT kommunikasjon krever nettverkstilgang, men ikke nødvendigvis internetttilgang, og dermed var det først nødvendig å koble håndkontrolleren til internett. Deretter ble

programmene for Serveren og håndkontrolleren utviklet slik at de kobler seg opp til samme MQTT broker. På den måten kunne håndkontrolleren publisere en verdi til et "MQTT-topic!", for så at Serveren kunne subscribe til samme topic og dermed motta riktig verdi. Serveren mottok da en verdi som den kunne bruke til å undersøke om det respektive rommet var ledig eller ikke.

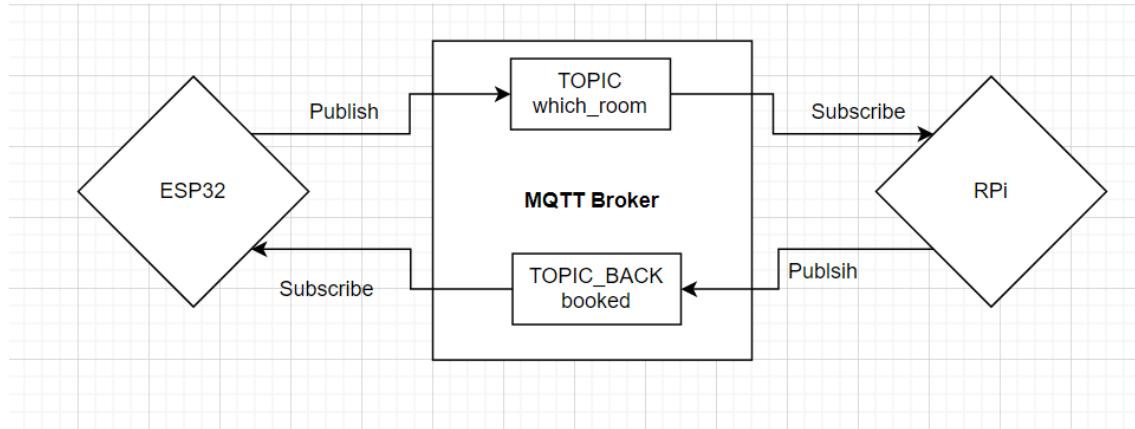


Figure 26: Flytskjema for MQTT kommunikasjon mellom håndkontroller og server for bookingsystemet

I tillegg til at håndkontrolleren sender en verdi til Serveren, skulle også Serveren sende en verdi tilbake. Bookingsystemet går gjennom og sjekker om det er ledig i rommet som er etterspurt, deretter publiserer den en verdi til et topic, om hvorvidt personen har fått tildelt plass eller om det er fullt. Dette gjorde vi å opprette en annen topic, som tok for seg verdien sendt tilbake til håndkontrolleren. Bruker vil da få beskjed om bookingen har gått gjennom eller om det er nødvendig å vente til det er ledig.

4.3 Adgangskontroll

I adgangskontroll-gruppen ble det tidlig satt opp ett møte for å diskuterte hva funksjonen til inngangpartiet skulle være. Det ble raskt avgjort at de tre hovedfunksjonene som skulle implementeres var adgangskontroll, ringeklokke og logging av personer. Derifra ble det diskutert hvordan funksjonalitetene best kunne utvikles med et mest mulig virkelighetsnært perspektiv. Fordelingen innad i adgangskontrollgruppa, med tanke på utvikling og arbeidsfordeling, ble at den med mest erfaring fikk de mindre oppgavene, slik at den med minst erfaring fikk prøve seg alene først på de mer utfordrene seksjonene, for etterhvert å samarbeide med den med mest erfaring og utvikle sammen. Denne løsningen ble valgt for den ble sett på som den mest lærerike metoden.

Systemet som ble besluttet å utvikle har følgende flytdiagram:

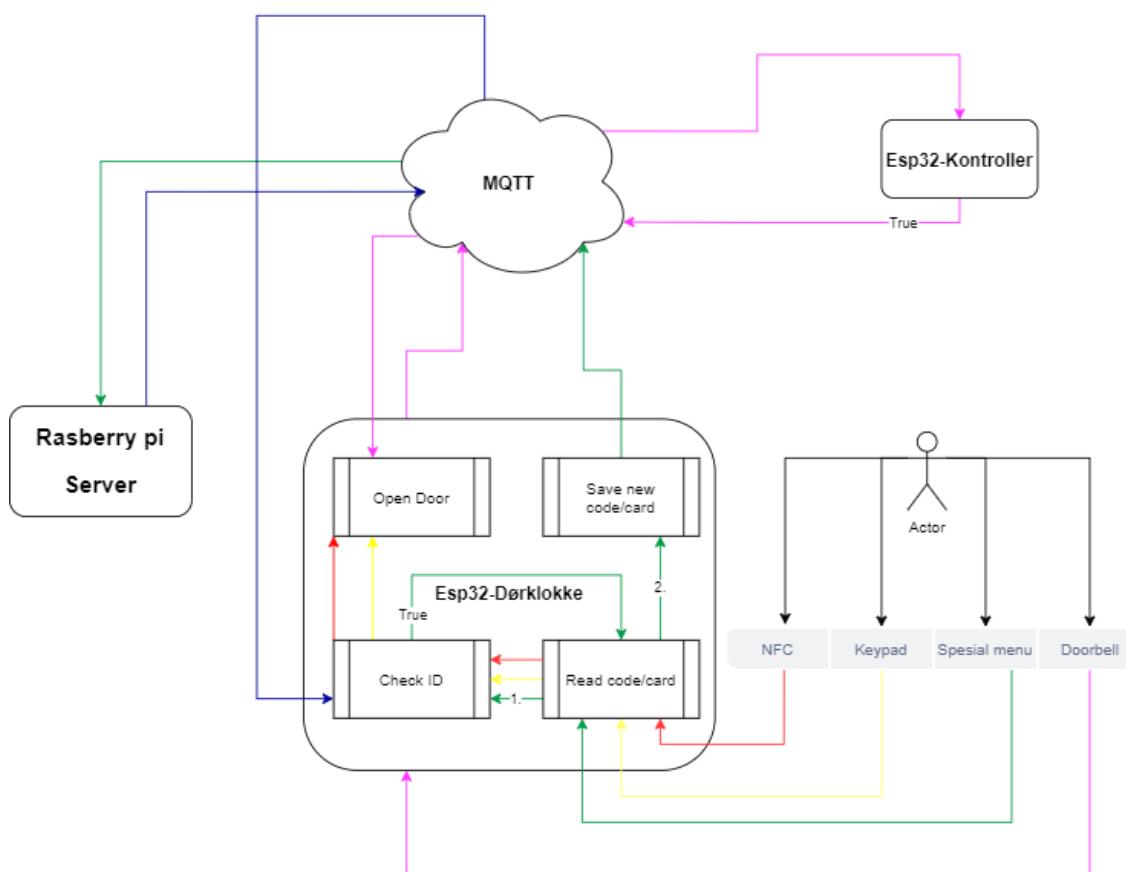


Figure 27: Flytskjema - Adgangskontroll

4.3.1 Logging av personer i kollektivet

Under idemyldringen av metoder for logging av personer i kollektivet ble det til slutt besluttet å bruke 2 beam-break sensorer som plasseres i dørkarmen. Denne løsningen ble

besluttet på grunnlag av at folk fysisk blir tvunget til å gå en og en forbi sensoren. For å sikrere at de ikke ville telt ekstra folk som kunne lene seg mot døren ble det bestemt å bruke en dør magnet sensor for å være sikker på at døren er åpen for at den skal telle eller trekke fra personer. Det ble vurdert flere alternative ideer til adgangskontroll, som f.eks trykksensorer for å plukke opp antall som passerer og kamera med ansikts-gjenkjenning. Disse løsningene ble forkastet basert på vurdering av kostnader mot effektivitet, personvern, begrensninger og kompleksitet. En løsning som ble vurdert som en potensiell fremtidig utvidelse var besøkskort, som er mer pålitelighet, og kan brukes til å identifisere hvem sin gjest som også forlater bygget.



Figure 28: Betjeningspanel inngang - produksjonsmodell

4.3.2 Innslåsing og kryptering

Når brukere og gjester kunne logges på en effektiv måte, var neste del av adgangskontrollanlegget innslåsing av brukere. For å unngå behov for fysiske nøkler, ble det bestemt at en RFID-leser (Radio Frequency Identification) skulle kunne brukes til å lese f.eks studentkort eller andre kort med RFID-chip og låse opp døren. I tillegg ble det implementert mulighet for å åpne døren ved inntasting av en 4- til 6-sifret kode, i tilfelle tap av adgangskontrollkortet. Først ble det vurdert å bare koble 12 trykknapper opp mot 12 porter på ESP32en, som var kjernen i adgangskontrollpanelet, men denne ideen ble skrinlagt når det førte til problem med tanke på antall ledige pins på mikrokontrollerkortet. Det ble derfor gått over til en matrise-keypad som bare bruker 7 pins.

Ved tap av kort, er en bruker avhengig av å kunne legge inn et nytt kort og endre pinkode.

Denne muligheten ble derfor implementert. Med tanke på sikkerhet, kan ikke hvem som helst endre kort eller kode, og en bruker må derfor identifisere seg før dette skal kunne utføres.

I og med at data lagres volatilt i minnet til ESP32en, var det behov for sikkerhetskoping av UDI, passord osv, og disse data lagres derfor på serveren og synkroniseres imellom enhetene jevnlig. Slik mister ikke adgangskontrollsystemet data ved f.eks strømbrudd. Sikkerhetskopiering av sensitiv data drar likevel med seg utfordringer med tanke på sikkerhet og for å minimere sjansen for datainnbrudd fra f.eks rivaliserende linjeforeninger, ble datakryptering implementert.

Krypteringen baserer seg på SHA-256 som gjør at dersom noen intersepter koden og fanger den opp, har de ingen måte å finne ut av hvilken kode som ble brukt for å komme inn, siden det er en-veis kompresjons funksjon (forklart i 3.2.3). Til slutt, når du har kommet inn og innser på at du glemte å handle, la vi inne en knapp som åpner døren innenfra.

4.3.3 Dørklokke

For å unngå trengsel i inngangssarealet dersom det ringer på til kollektivet og alle beboerne går for å åpne døra, ble det ansett som hensiktsmessig å ha et adresserbart ringeklokkeanlegg som bare ringer opp beboeren som får besøk. Oppringingen går til håndkontrolleren. Med den vil en beboer ha mulighet til å avise eller åpne døren inn til kollektivet. Dermed er det mindre fare for kræsje med det som bruker stua. På adgangskontrollpanelet må det være mulighet for å velge hvem man vil ringe på til. I og med at en ESP32 har et bregrenset antall hardwarepins, vil man ikke kunne ha en trykknapp per beboer. En slik løsning vil i tillegg bety at navn må endres fysisk ved utskifting av beboere i kollektivet, og at utviding av antall brukere kompliseres. Derfor valgte vi og gå for en interface der en gjest får en meny med navigering der han kan velge mellom navnene.

Tilsatt ble det lagt inn et energisparingsmodus, som skrur av displayet for strømsparing og for utvidelse av levetiden. Ved videre utvikling vil det også kunne være ønskelig å sette ESP32en i light sleep mode når displayet er avslått og kun være koblet til nettverk når det skal sendes data eller venter på respons. Dette ble ikke implementert i denne omgang ettersom det ikke ble ansett som kritisk da enheten til enhver tid er tilkoblet fast strøm.

4.4 Raspberry Pi - Server

Det ble vurdert som mest hensiktsmessig å benytte en Raspberry Pi som server i systemet, da den er kompakt, billig og kraftig nok til oppgaven. I motsetning til mikrokontrollerene, har en RPi mulighet for langtidslagring av data, noe som er fordelaktig med tanke på lagring av brukerinstillinger, dataanalyse osv.

På serveren ble det bestemt at følgene oppgaver skal utføres:

- Administrasjon av MQTT
- Administrasjon av booking
- Administrasjon av brukernøkler
- Innhenting og bearbeiding av værdata
- Beregning / simulering av kraftproduksjon fra solcelleanlegg
- Logging av strømforbruk individuelt for hver rom.

5 Resultat

5.1 Romkontroller

En omfattende utviklingsprosess har resultert i et produkt som opererer som ønsket. Den endelige prototypen bestod av følgende komponenter:

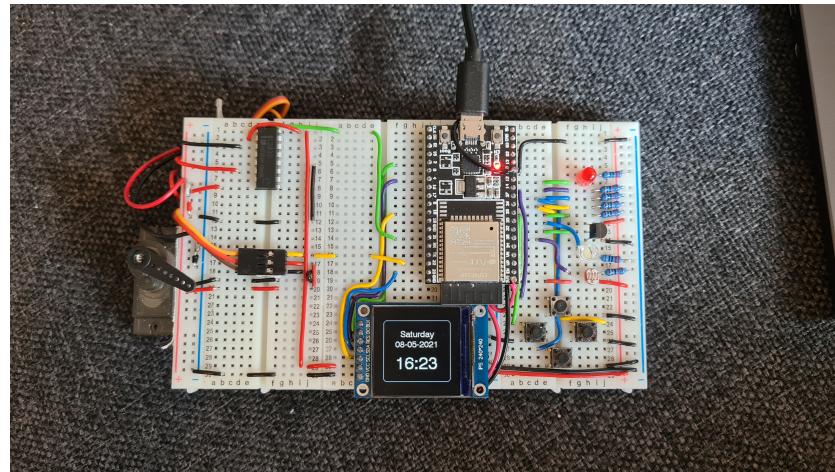


Figure 29: Romkontroller - Endelig prototype

- Fire trykknapper, med $10\text{k}\Omega$ pull-up resistor, for interaksjon med enheten
- Et 1,3" LCD-display av type IPS240x240 koblet til ESP32en og satt opp med SPI-kommunikasjon
- En fotoresistor for måling av lysintensitet i rommet, koblet opp som en spenningsdeler sammen med en $10\text{k}\Omega$ resistor der spenningen blir avlest av en ADC-kanal på ESP32en.
- En hvit LED koblet opp med 220Ω serieresistor og regulert av en 3.3V PWM-utgang på ESP32en, for visualisering av lyspådrag.
- En TMP36 temperatursensor, der utgangsspenningen blir avlest av en ADC-kanal på ESP32en, for måling av romtemperatur
- En rød LED med 220Ω serieresistor, koblet til en styrt 3.3v utgang på ESP32en, for visualisering av varmestyring
- En H-bro til styring av viftemotor, der effekt-kretsen var tilkoblet 9.0V tilførselsspenning, den ene dreieretnings-inngangen var fast tilkoblet 3.3V og enable-terminalen var regulert av en 3.3V PWM-utgang på ESP32en, for viftepådragsjustering.
- En servomotor tilkoblet 9.0V tilførselsspenning og regulert av en 3.3V PWM-utgang på ESP32en

Menynavigasjon

Menynavigasjonen fungerer som planlagt. En bruker kan intuitivt navigere inn i undermenyer og gjøre nødvendige innstillinger. Pil opp og pil ned gjør at en bruker kan bla opp og ned i en menyliste, pil høyre brukes for å åpne valgt undermeny eller utføre operasjoner, og pil venstre tar brukeren tilbake til forrige nivå i menyen. LCD-displayet visualiserer til enhver tid hvor brukeren er i menyen, med mindre enheten utfører setup-prosedyre, viser at skjermssparer er aktivert eller en verdi endres. I disse tilfellene vises dedikerte vinduer

Lysstyring

Den implementerte lysstyringen regulerer lysnivået i det gitte rommet automatisk basert på hvor mye dagslys som til enhver tid slippes inn. Dersom dagslysinnslippet er lavt, vil styringen øke mengden kunstig lys i rommet, slik at den totale lysintensiteten tilsvarer ønsket lysintensitet. Brukeren kan gjennom valgmenyen sette et nytt referansepunkt for ønsket lysintensitet, og styringen vil regulere lyspådraget til å treffe dette nye nivået. Innstilling av nytt referansepunkt gjøres typisk midt på dag når dagslysinnslippet er stort.

Dersom det er ønskelig kan en bruker også manuelt dimme ned lysstyrken, enten via valgmenyen på romkontrolleren, eller gjennom brukerpanelet i CoT. Lyset kan også skrues av og på fra valgmenyen på romkontrolleren og fra CoT.

Automatisk temperatur- og luftekontroll

Den automatiske temperatur- og luftekontrollen opererer etter en fastsatt ukeplan. Ukeplanen er delt inn i 7 dager der hver dag er delt inn i 24 timer. For hver time kan brukeren definere en av tre statuser: Home, Away og Night. En standard ukeplan blir automatisk lagt inn ved oppstart, men brukeren kan til enhver tid endre ukeplanen via valgmenyen på romkontrolleren. Dersom en bruker eksempelvis nавигerer seg gjennom menyen på undernevnte måte, vil han til slutt komme til skjermbildet vedlagt til høyre.

Temperature → Weekly plan → Monday → 00.00 - 01.00

Den blå ringen rundt "Night" vil i dette tilfellet bety at nattmodus er definert for mandag fra 00.00-01.00. Dersom brukeren vil endre på det, trykker han oppover eller nedover i menyen. Den hvite pilen viser til enhver tid hvilket modus han kan velge. Når pekeren viser ønsket status, oppdaterer brukeren statusen ved å trykke pil høyre. Den valgte statusen vil nå få blå ring rundt i stedet for den gamle statusen.

Videre kan en bruker bestemme hvilken temperatur som skal vere gjeldende for de forskjellige statusene. Ved å navigere seg gjennom valgmenyen på følgende måte, vil en bruker eksempelvis komme frem til justering av Home-temperatur:

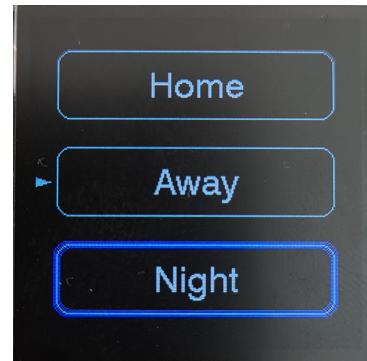


Figure 30: Romkontroller
Valg av status for valgt time

Temperature → Temp. adjust → Home

Justering av temperatur for de forskjellige statusene er også mulig i CoT.

Et eget modus for langtidsfravær aktiverer en frost-sikringsstyring for rommet. Takvifte vil ikke kunne starte på grunn av økt fare for brann, luftevindu vil ikke kunne åpnes på grunn av økt fare for innbrudd, og varmereguleringen vil skje utelukkende ved hjelp av varmeovn. Dette moduset kan bare aktiveres manuelt via menyen til romkontrolleren, men ønsket temperatur for moduset kan justeres gjennom CoT.



Figure 31: Romkontroller
Temperaturjustering

Manuell luftekontroll

Dersom det er ønskelig, kan en bruker velge å overstyre den automatiske styringen av vindusåpning og takvifte.

Brukeren kan da i stedet selv bestemme effekt på takvifte og hvor mange grader åpning vinduet skal ha. Manuell modus kan bare aktiveres via valgmenyen på romkontrolleren, men dersom manuell modus er aktivert, kan vindusåpning og vifteeffekt justeres både via valgmenyen på romkontrolleren og fra CoT.

Forbruksmålig

Forbruket på det enkelte rommet blir til enhver tid regnet ut basert på styringen til panelovnen. Forbruksverdien lagres i en liste med 25 elementer, der det første elementet indikerer dagen i måneden og de resterende 24 elementene gir forbruket time for time. Denne verdilisten sendes så til serveren (Raspberry Pi) for forbruksanalyse og logging. Forbruksdata lagres lokalt på romkontrolleren i inntil 48 timer før det blir overskrevet, mens det på server lagres over lengre tid for logging.

Nettverkskommunikasjon

Romkontrolleren er ikke avhengig av nettverkstilkobling for å fungere på kort sikt, men ved lengre tids nettverksutfall vil noen funksjonaliteter ikke være tilgjengelige.

Ved oppstart er det dog viktig at romkontrolleren har nettverkstilgang, da den er avhengig av å synkronisere den interne klokken mot NTP-servere. Dersom nettverk ikke er tilgjengelig, vil kontrolleren operere ut ifra 1. januar 1970. 00.00.

Den interne klokken til kontrolleren er korrekt nok til at mangel på jevnlig synkronisering ikke skal være et vesentlig problem, men det anbefales likevel at enheten til enhver tid har tilgang til nettverk. Dersom romkontrolleren ikke har tilgang på nettverk, vil det heller ikke være mulig å fjernstyre den gjennom CoT. Kommunikasjon av forbruksdata mellom serveren og romkontrolleren vil fungere selv om internetttilkobling er fraværende, så lenge det lokale nettverket er operasjonelt. Dette er fordi MQTT-tjenesten som benyttes,

opererer lokalt. Dersom det ikke er mulig å koble til et lokalt nettverk, vil heller ikke forbrukslogging fungere.

Meteorologiske data for temperaturregulering vil heller ikke være tilgjengelig dersom romkontrolleren ikke har nettverksforbindelse, hvilket betyr at romkontrolleren ikke han dra nytte av høy utetemperatur for innendørs oppvarming.

5.2 Håndkontroller

5.2.1 Booking

Programmet vi lagde for å booke de ulike rommene, ga ønsket resultat, ved at den leste om det er var ledig eller ikke, og oppdaterte statusen med nye verdier.

Kode for rom status

```

1 #Koden sjekker stausen på kjøkkenet. Hvis det er en plass som er
  ledig, vil verdien bli erstattet med "busy", og det vil bli
  sendt en melding tilbake til ESPen om at man har booket kjø
  kkenet for 30 minutter. I tillegg vil det lagres en verdi som
  er nåverende tid i millisekunder. Denne verdien blir brukt for
  å finne ut på hvilket tidspunkt i millisekunder, tiden for
  rommet har løpt ut

2
3 if kitchen['slot_1'] == 'free':
4     kitchen['slot_1'] = 'busy'
5     kit_time1 = current_milli_time()
6     write(user, 'Kitchen')
7     mqtt_publish("booked", "You have booked kitchen slot 1 for 30
      minutes")

8
9 elif kitchen['slot_2'] == 'free':
10    kitchen['slot_2'] = 'busy'
11    kit_time2 = current_milli_time()
12    write(user, 'Kitchen')
13    mqtt_publish("booked", "You have booked kitchen slot 2 for 30
      minutes")

```

For at bookingprogrammet skulle være klar over hvilket rom som skulle bookes, trengte ESP32en å sende en unik verdi for de ulike rommene, og serveren måtte så lese den verdien. Verdiene som ble sendt var henholdsvis "Kitchen", "Toilet", "Shower", "Livingroom", som representerer de ulike bookingalternativene.

Kode for publisering av verdier til MQTT, skrevet i Arduino IDE

```

1 //Koden sender en verdi for det spesifikke rommet til en MQTT
   topic, etter at du har valgt å booke kjøkkenet
2
3 if (menu_adress[2] == 1) {
4     mqtt_message.resiver = "Hub";
5     mqtt_message.header = Booking;
6     mqtt_message.room = Kitchen;
7     mqtt_message.data_int[0] = { "user", user };
8     mqtt.pub(mqtt_message, TOPIC, false);
9     menu_lvl = 1;
10    menu_adress[0] = 1;
11    current_lvl_val = 1;}
```

```

Connected with result code 0
{"id": "User4", "header": 1, "room": 7, "data_int": {"key": 10},
 "data_String": {}}
{"slot_1": "busy", "slot_2": "free"}
data published

{"id": "User4", "header": 1, "room": 7, "data_int": {"key": 10},
 "data_String": {}}
{"slot_1": "busy", "slot_2": "busy"}
data published
```

Figure 32: Viser melding sendt fra ESP32en, samt statusen på kjøkkenet etter at den har mottatt verdien for kjøkken

Figuren over viser meldingen som server mottar. Deretter i programmet fjernes 'room' fra meldingen, som i dette tilfellet tilsvarer kjøkkenet. Deretter blir funksjonen for booking av kjøkkenet kjørt.

Kode for når bookingfunksjonen blir kalt, skrevet i Python

```

1 # Koden sjekker hvilken verdi 'room' har fra meldingen mottat fra
   ESP32, deretter starter funksjonen for booking av kjøkkenet.
2 if (payload['room'] == Room.Kitchen):
3     time = 30*60000                      # Setter bookingtid for kjøkkenet
   til 30 minutter
4     kitchen = kitchen_status('Kitchen', time, user)
```

Dette fungerer på samme måte for de andre rommene man kan booke. Hvis det er ledig, vil statusen bli omgjort til at det er opptatt. Til slutt i funksjonen sender den en melding tilbake til ESP32 om at man enten har fått tildelt plass, eller så er det fullt. Dette vises også i figur 8 ved meldingen "data published".

Oppnådd resultatet samsvarer derfor med oppgaven og ideene vi hadde ved starten.

5.3 Adgangskontroll

Løsningen vi endte opp med var en ringeklokke med 3 knapper for navigering på et display, med en keypad og en RFID-kortleser koblet til en ESP32. ESP32en kommuniserte gjennom MQTT for å sende en datapakke til den valgte håndholdte kontrolleren hvis noen ringer på, og ellers bruker den MQTT til å sende brukernøkler som lagres på serveren. ESP32en åpner døren ved hjelp av en servo som dytter låsesylinderen inn og ut. Du kan styre servoen ved positiv respons fra håndkontrolleren ved dørklokke funksjonen eller med en av adgangsmulighetene implementert i dørpanelet (keypad eller RFID-leseren). På skjermen kan du navigere deg fram til hvem du vil ringe på til og det vil generere et anrop til brukeren, der han kan velge å åpne døren eller avvise anropet. Dersom ingen respons kommer, oppfattes dette som at brukeren ikke er hjemme.

Om du, som bruker, skal komme deg inn trenger du bare skrive inn din personlige kode eller skanne studentkortet ditt, så vil døren åpne seg og lukke seg 10 sekund etter døren sist ble registrert lukket. Dersom adgangskortet ikke er tilgjengelig eller det er tapt, kan du legge til et nytt og med din personlige kode. Dette kan du gjøre gjennom en spesiell meny du kommer til ved å holde inne 0-knappen i 2 sekund. Du må og autentiseres som en beboer med en kode eller et kort og du kan nå velge om du vil legge til et kort eller kode med henholdsvis alternativ 1 eller 2. Når du kommer inn brukes det 3 sensorer for å passe på om du kommer inn eller er på vei ut. Dette viser bare hvor mange som er i kollektivet, men ingen indeksering av hvem individene er. Om ingenting skjer på adgangskontrollpanelet vil den skru av skjermen for å spare strøm. Vi la også tilslutt inn en åpne knapp fra innsiden som vil åpne døren fra innsiden når du skal ut igjen.

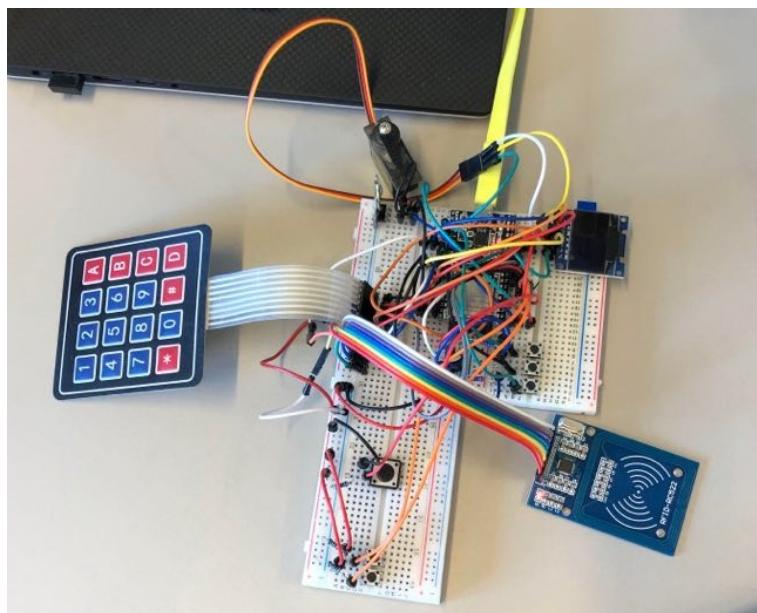


Figure 33: Endelig prototype - Adgangskontroll

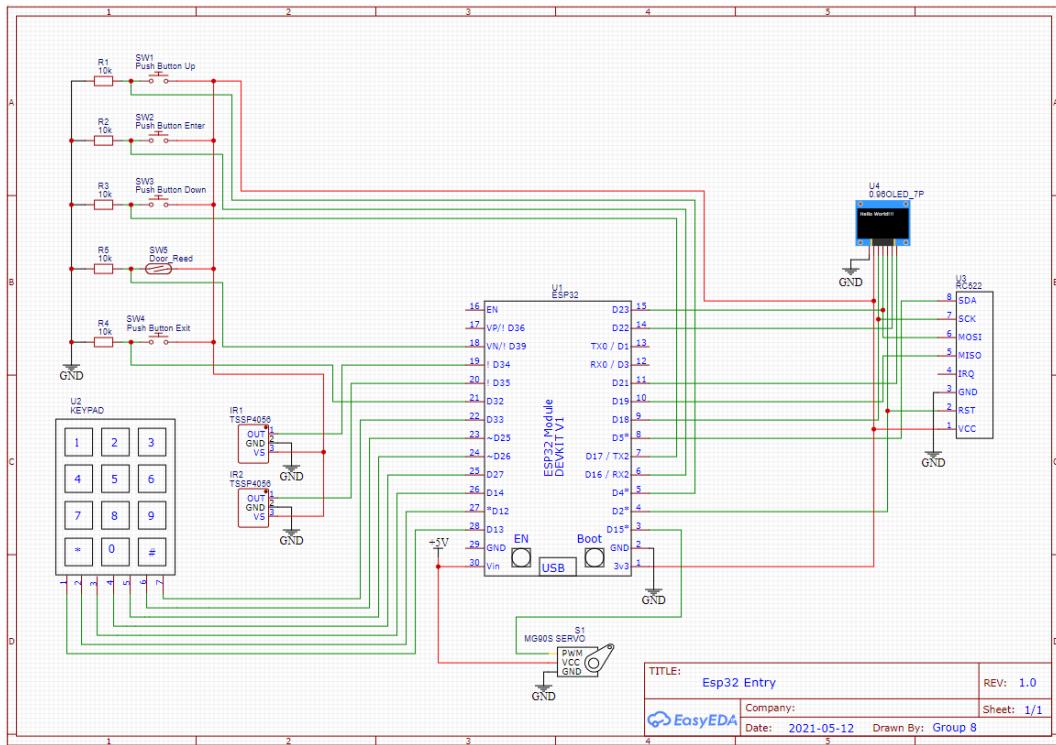


Figure 34: Koblingsskjema - Adgangskontroll

5.4 Raspberry Pi - Server

Serveren (Raspberry pi) er tatt i bruk som en hovedsentral. Mesteparten av signalene sendes gjennom denne, for å ha et sentralisert punkt som vil ha oversikt over hva som skjer på systemet. Systemer som booking, energilogging, innhenting av værdata og lagring av data havnet her. Fra serveren blir dataen distribuert videre til riktig node (ESP32) eller innhentet fra noder som ønsker å lagre data.

Bookingsystemet havnet først på serveren, og om noen ønsker å booke et rom, vil det i serveren bli satt til opptatt. Forespørselen kommer som en datapakke over MQTT, spør om det er ledig, og responderer om det er ledig. Hvis det er ledig vil brukeren få beskjed om at bookingen har gått gjennom og man kan bruke rommet et visst antall minutter. Ellers vil det bli gitt beskjed om at det er opptatt.

Videre ble det utviklet et energiloggingssystem for å få oversikt over strømmen produsert av solcellepanelene og energien forbrukt av beboerne. Systemet mottar data fra nodene i bygget. I første utgave hentes det kun data fra romkontrollerene, men systemet er laget for å motta data fra alle nodene ved full utvikling av prosjektet. For hvert år lagres det en separat fil med året og "Power usage" kombinert for å enkelt kunne finne frem til ønsket data. Det vil logges time for time, og er lagret i en csv-fil. Det blir også brukt en h5-fil for rask og enkel behandling av dataframe.

På neste side følger et bilde av formatet data'en lagres i:

			10/05/2021	10/05/2021	10/05/2021	10/05/2021
			0	1	2	3
ID	Room	Booked				
Sum			0	0	0	0
Sun panel			0	0	0	0
user1	Bathroom	Shower	0	0	0	0
user1	Bathroom	Toilet	0	0	0	0
user1	Dorm		0	0	0	0
user1	Kitchen		0	0	0	0
user1	Livingroom		0	0	0	0
user2	Bathroom	Shower	0	0	0	0
user2	Bathroom	Toilet	0	0	0	0
user2	Dorm		0	0	0	0
user2	Kitchen		0	0	0	0
user2	Livingroom		0	0	0	0

Figure 35: Power usage 2021.csv

Data'en fra energi forbruket blir oppdatert hvert 15min til CoT for en visuell graf av samlet forbruk hos brukerne.

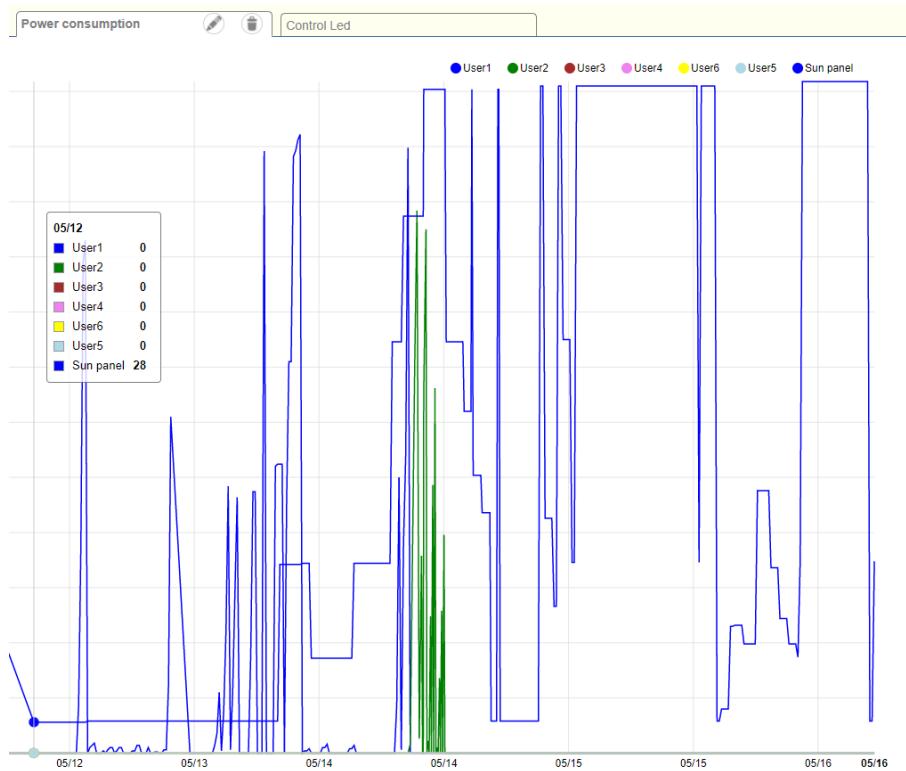


Figure 36: Energi forbruk i CoT

Beklagelig logges både user1 og sun panel i blått men den gjenvneste er energien produsert av solcellepanelene.

I en senere fase av prosjektet ble det diskutert rundt en mulig løsning på innhenting av værdata og beregning av energiproduksjon fra solcelleanlegget. På grunn av resterende tid til deadline og ble det bestemt at vi skulle gå for en enkel løsning som dekket det nødvendige. Openweathermap tilbyr en enkel distribuering av værdata, og ble ansett som et godt alternativ. Værdata ble hentet inn til serveren med koordinatene til Trondheim Sentrum. En enkel funksjon estimerer energiproduksjonen fra solcellene. Den består av soltimer, graden av skydekke og estimert solstyrke.

Serveren blir også brukt til å lagre brukernes adgangskontroll-nøkler som SHA-256-hasher i en csv-fil. Det er herfra adgangspanelet får tilsendt nøklene den bruker for å verifisere brukerene, og lagrer de nye som blir lagt til.

Alle disse funksjonene blir kjørt fra et hovedprogram som i tillegg tar seg av MQTT-kommunikasjonen og bestemmer frekvensen på de forskjellige oppgavene som skal kjøres.

For å enklere kunne feilsøke problemer som oppstår under programets levetid, blir beskjeder og feil som oppstår logget til en egen fil med tilhøring tidsstempel. Om en feil oppstår mer enn 10 ganger i løpet av det neste minuttet vil programmet termineres. Frem til dette vil programmet forsøke å resette koblingen til MQTT og forsøke å starte opp igjen.

5.5 MQTT-kommunikasjon

Tidlig i utviklingen ble det bestemt at det var behov for en standard på hvordan vi skulle sende data'en over intranettet for et mest mulig system. Det ble derfor utviklet et mqtt bibliotek basert på PubSubClient biblioteket[20], som enkelt kan integreres i programkodene. Det ble tatt i bruk json over mqtt for å pakke data'en til et ønsket format. Før data'en pakkes lagres den i en strukt som gjør det enkelt å opprettholde en standard for alle datapakkene som skal sendes.

Kode for publisering til MQTT

```

1 // Her settes navnet på den som skal motta data`en
2 mqtt_message.id          = "Hub"
3 // Her settes hva meldingen omhandler
4 mqtt_message.header      = Room_Controller; // Enum
5 // Her settes hvilket rom som data`en omhandler
6 mqtt_message.room         = Kitchen; // Enum
7 // Her settes data`en som ønskes å sende i type int}
8 mqtt_message.data_int[0]   = { "TempOut", 10 };
9 // Her settes data type int nr2
10 mqtt_message.data_int[1]  = { "TempInside", 19 };
11 // Her settes data`en som ønskes å sende type string
12 mqtt_message.data_String[0] = { "Symbol", "C" };
13 // Her publiseres datapakken
14 mqtt.pub(mqtt_message);
15

```

6 Diskusjon

6.1 Romkontroller

6.1.1 Ytelsesvurdering

Selv om romkontrolleren sin funksjon er som ønsket, er der en rekke funksjonaliteter som ville blitt løst annerledes dersom de skulle gjenskapes.

Display

Systemet som ble implementert for tegning av ulike vinduer på LCD-displayet er en av de delene av programmet som ville blitt gjort på en helt annen måte dersom systemet skulle gjenskapes. Det nåværende systemet fungerer, men det er lite programmeringsvennlig. Det bærer preg av at det har blitt utviklet steg for steg etterhvert som nye behov har meldt seg, og dermed fremstår som uoversiktlig. "Biblioteket" ble utviklet før utvikler hadde fått kunnskap om objektorientert programmering, og i etterkant av at denne programmeringsmetoden ble studert ble det vurdert å gjenskape display-biblioteket med en objektorientert tilnærming. Dette ble derimot vurdert til å være for arbeidskrevende til å kunne forsvarer med den tiden som var disponibel.

Menynavigasjon

Teorien bak systemet som ble implementert for menynavigasjon ansees som god, og måten å adressere en meny på ville blitt kopiert dersom det skulle gjøres igjen. Likevel kunne nok dette systemet også med fordel blitt skrevet om med en objektorientert tilnærming for å gjøre hovedprogrammet mer lettleselig og oversiktlig.

Lysregulering

Funksjonen til systemet som ble implementert for lysregulering ansees som godt, men dersom systemet skulle gjenskapes, hadde denne funksjonaliteten også blitt bakt inn i RoomControl-klassen som allerede regulerer varme, lufting og takvifte. Systemet ville da fremstått som mer komplett og gjennomtenkt.

Romkontroll-regulering

RoomControl-klassen som ble opprettet for regulering av temperatur, lufting, takvifte, forbruksberegnung osv. fremstår som gjennomtenkt, og veldig få ting ville blitt endret dersom systemet skulle gjenskapes. Det hadde, som nevnt, vært fordelaktig å inkludere lysreguleringen i denne klassen, slik at klassen kunne fungere for en komplett regulering av alle aspekt ved den faste installasjonen på et gitt rom. Dersom styringen skulle utvides med flere funksjonaliteter, ville det blitt inkludert en separat funksjon som blir trigget ved deteksjon av brann. Luftevindu ville da blitt lukket for å hindre spredning, strømtilførsel til alle stikkontakter, takvifte og panelovn ville blitt kuttet, i tilfelle brannen skyldes

elektrisk feil eller overoppfeting, og lysintensiteten ville blitt justert opp på maksimalt nivå for å gjøre evakuering enklere.

Backup av innstillinger

En av bakdelene ved måten romkontrolleren nå fungerer er at mange innstillinger bare lagres volatilt i minnet til mikrokontrolleren. Dette vil si at ved en reboot av systemet, vil de fleste innstillingene gå tilbake til fabrikkstandard. Temperaturinnstiller, dimmestatus, manuell vindusåpning og manuell takvifteffekt vil lagres i CoT og synkroniseres til romkontrolleren når den er tilkoblet trådløst nettverk, men ukeplanen der en bruker har definert hvilken bruksstatus de forskjellige timene i løpet av et døgn, og i løpet av en uke, skal ha, vil bli tilbakestilt til fabrikkstandard.

Dette vil enkelt kunne løses ved å oppbevare en "sikkerhetskopi" av ukeplanen på serveren (RPi). Denne ukeplanen kan enkelt synkroniseres mellom enhetene ved hjelp av MQTT, og en bruker vil på den måten ikke tape egendefinerte innstillinger ved f.eks strømbrudd eller overbelastning av en forbrukerkurs.

6.1.2 Produksjonsmodell

Hensikten med prototypen som ble utviklet til dette prosjektet var å kunne ha en utviklingsplattform med alle ønskede sensorer oppkoblet, i tillegg til å ha muligheter for å visualisere funksjonaliteten til styringssystemet. Følgelig vil det kreve videre hardwareutvikling før en komplett prototype er ferdigstilt og kan testes som en integrert del av en elektrisk installasjon. Vi skal her ta for oss de endringene som er forespeilet for en endelig prototype / produksjonsmodell:

Strømtilførsel

I og med at denne enheten skal kunne implementeres i en fast elektrisk installasjon, er den avhengig av å være installasjonsvennlig nok til at produktet blir kjøpt. Dette vil si at den må kunne kobles til et normalt 230V nett og ikke være avhengig av egen 5V tilførsel. Det må derfor implementeres en 230V AC → 5V DC strømforsyning internt i romkontrolleren. Videre må det også integreres en strømforsyning som kan forsyne en større servomotor enn den som er implementert i prototypen.



Figure 37: Romkontroller - Fremside

Sterkstrømskretser

En produksjonsmodell må ha mulighetet til å styre 230V laster uten behov for eksesterne modularer, da dette er standarden i elektriske installasjoner. Det må derfor implementeres utstyr som kan styre og drive laster som opererer på denne driftsspenningen. En normal forbrukerkurs kan bli sikret med intil 16A 230V, og de interne sterkstrømskretsene må derfor dimensjoneres til dette. I og med at varmestyring gjøres ved å slå på eller av en varmekilde, blir det ansett som mest hensiktsmessig å implementere et rele for å styre denne. Dette blir styrt fra en digitalutgang på ESP32en. Slik blir belastningen på tilførselsterminalene for driftspenning til romkontrolleren redusert, og man reduserer slik også brannfare. Videre må det implementeres to PWM-regulerte 230V dimmere til lys- og vifte-regulering. Disse blir styrt fra en PWM-utgang på ESP32en. Til slutt må det, som nevnt i forrige avsnitt, implementeres en strømforsyning som kan forsyne en servomotor stor nok til å åpne et vindu. Hovedspenningen til servomotoren kommer fra denne, mens styresignalet blir regulert av en PWM-utgang på ESP32en.

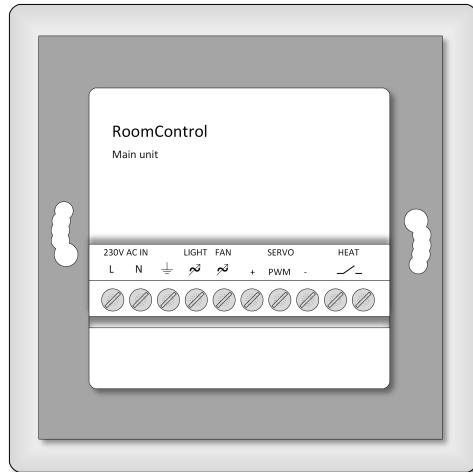


Figure 38: Romkontroller - Bakside

6.2 Håndkontroller

6.2.1 Booking

Ut fra resultatene så har vi oppnådd ønsket funksjon på booking. Hensikten var at det skulle være mulig å etterspørre booking av et room, for så at et program skulle sjekke hvorvidt det var ledig i rommet eller ikke. Systemet gjør det mulig for studentene i kollektivet å bruke utstyret som følger med leiligheten. Ved bookingen oppnås det kontroll over hvor mange som har tilgang på rommene, slik at smitterestriksjonen blir overholdt. I tillegg gir det også mulighet for at alle studentene kan dra nytte av inventaret til leiligheten. Siden det bookes på nåværende tidspunkt, lukes ut muligheten for å reservere for eksempel stuen en hel ettermiddag. Alle vil derfor ha mulighet til å slappe av litt på sofaen, samtidig som det er en effektiv rulling på bruk av kjøkkenet og badet.

Under programmering oppsto det til tider en del problemer, som ikke alle var like lett å finne en løsning på. Dette gjaldt særlig bruk av MQTT. Et av de største problemene var å bruke den leste verdien sendt fra håndkontrolleren til å booke riktig rom. For å lese verdien måtte vi opprette en funksjon som tok seg av dette. Dermed ville denne meldingen bli lagret lokalt i funksjonen, men vi hadde behov for å bruke den utenfor. Dette ble til slutt løst ved å kalle booking funksjonen inne i funksjonen som mottok meldingene. Det

sikret at vi fikk brukt meldingen til å avgjøre hvilket rom som skulle bookes.

Et annet problem som også var relatert til MQTT, var når Python-scriptet skulle publisere en melding tilbake til håndkontrolleren. Komplikasjonene oppsto når dette skulle implementeres med funksjonen som gjør at scriptet hele tiden leter etter nye verdier å lese. Funksjonaliteten var at den skulle lese meldinger, så det ga utfordringer da den i tillegg skulle kunne skrive verdier. Igjen ble dette løst med å etablere en funksjon som også tok seg av publiseringen, som i tillegg ikke skapte problemer for loopen. På den måte kunne man enkelt skrive "mqtt_publish" og så melding man ønsker å sende, for å skrive en verdi til håndkontrolleren.

Selv om programmet fungere slik det var ment til, så eksisterer det svakheter og forbedringspotensiale ved funksjonaliteten. Dette er ved nedtelling av bookinger på rom hvor det eksisterer forskjellige tider. I kjøkkenet er det samme tid, 30 minutter, uansett når du booker. Derimot ved badet og stuen, er det ulike tider utfra hva du skal gjøre. Ved bruk av toalett blir bruker tildelt 5 minutter, men 15 minutter hvis dusjen skal benyttes. Svakheten oppstår ved nedtellingen og undersøkelse av hvor lenge det er igjen. Hvis du har booket dusjen, som tilsvarer 15min, og en annen bruker legger inn en forespørsel om å bruke doen, vil han få beskjed om at det er 5 minutter til badet er ledig, selv om det ikke stemmer. Derimot så vil først plassen bli ledig etter 15 minutter, og ikke 5 minutter, eller hvis bruker velger å kansellere bookingen sin. Det samme problemet oppstår hvis det legges inn en booking på stuen, mens den er full, da vil det bli gitt beskjed om at stuen er ledig om et visst antall minutter, selv om det mest sannsynlig ikke er riktig tid. Dette har med at variablene for tiden som gjenstår, lagrer seg lokalt, så hvis du etterspør en annen tid enn den som er booket, vil den ikke klare å sammenligne de to, og dermed ikke konkludere med hvor lenge det egentlig er til rommet er ledig.

Et annet element ved booking som inneholder svakheter, er at det ikke er mulig å booke frem i tid. Dette er en funksjonalitet som ikke er implementert fordi tanken var at det ikke skisserer realistisk oppførsel for medlemmer i en husstand. Ideen er at i løpet av en dag så er det impulsive tanker som avgjør når det er ønskelig å ta en matbit, gå på do eller slappe av. Det er mulig å argumentere for at det kan være en nyttig funksjon, ettersom det gir mulighet til å planlegge dagen sin bedre. Derimot så er det mulighet for at de som er først ute med å booke, skaper et situasjon hvor kjøkkenet kan være opptatt i flere timer på ettermiddagen, uten at de andre har mulighet til å bruke kjøkkenet og lage seg mat. Hvis dette oppstår som et problem, vil det være naturlig å legge en funksjon som gjør at det ikke er mulig å booke i to omganger, at det kun vil være tilgjengelig å booke 30 minutter om gangen.

Først i programmet har vi en funksjon valid-date() som sjekker om nye bookinger befinner seg innenfor parameterne satt for år, måned og dag, også gitt at diverse måneder har ulike lengder. Dersom funksjonen mottar en feil input satt de gitte parameterne, returnerer funksjonen false. Videre har vi laget en funksjon Booking-Valid som har i oppgave å avgjøre om nye bookinger kolliderer med de foreliggende. For å unngå et hav av if-looper som sjekker om det er kollisjon mellom to bookinger for et rom, slo vi heller sammen alle verdiene for booking, slik at vi kunne sjekke at new-booking-start og new-booking-end

ligger utenfor variablene booking-start og booking-end hvor alle eldre bookinger er lagret. New-booking inneholder et dictionary som mottar booking verdiene. Under ligger bilde vedlagt Booking-Valid funksjonen.

```
def Booking_Valid(Room, new_booking):
    if Room == [0]:
        return True
    else:
        #new_booking = {"start": ci, "slutt": co, "name": n, "phone": p1}
        #20210513
        new_booking_start = new_booking["start"] [2]*1000+ new_booking["start"] [1]*100+new_booking["start"] [0]
        new_booking_end = new_booking["slutt"] [2]*1000+ new_booking["slutt"] [1]*100+new_booking["slutt"] [0]
        for booking in Room:
            #Itererer seg gjennom bookinger.
            #if ci <= booking_stue["tidspunktSlutt"] [n] and co >= booking_stue["tidpunktStart"] [n]:
            booking_start = booking["start"] [2]*1000+ booking["start"] [1]*100+booking["start"] [0]
            booking_end = booking["slutt"] [2]*1000+ booking["slutt"] [1]*100+booking["slutt"] [0]
            #Sjekker for når nye bookinger ikke krasjer
            if new_booking_start < booking_start and new_booking_end < booking_start:
                print("123")

            elif new_booking_start > booking_end and new_booking_end > booking_end:
                print("321")

            else:
                print("collision mothafacka")
                return False

        return True
```

Figure 39: Booking Valid Funksjon

Funksjonen sjekker når bookingen ikke kolliderer fremfor å sjekke når det oppstår kollisjon da vi fant ut at dette krever færrest sjekker vist grafisk i figur til høyre (grønne linjer for nye bookinger og blå for eksisterende). Alle bookinger forekommer i booking funksjonen hvor hver new-booking blir lagt til en global liste for hvert av rommene. Dessverre rakk vi ikke å lage et system for dagen så dette ville vært neste steg for å lage et fullstendig system for fremtidsbookingen.

Som nevnt tidligere skjer bookingen i en egen booking-funksjon som mottar brukerens ønske om hvilket rom som skal bookes. For å få booke ønsket rom sjekker den om ønsket booking for nytt rom ikke kolliderer med eventuelle tidligere bookinger. Dersom funksjonen Booking-Valid godkjenner at det ikke er kollisjon legges den nye bookingen til en global liste for det respektive rommet. For å sjekke at programmet virket lagde vi en record funksjon som har i oppgave å vise de gitte bookingene som er blitt godkjente.

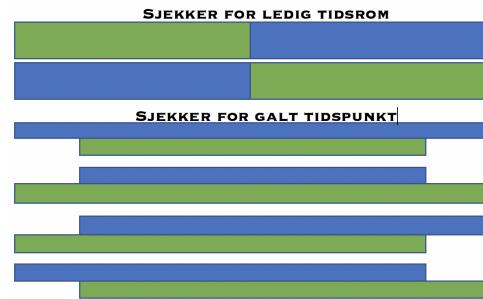


Figure 40

```
if ch==1:  
    if Booking_Valid(booking_kjøkken, new_booking):  
        booking_kjøkken.append(new_booking)  
        print("Room Type- Kitchen")  
    else:  
        print("Opptatt")
```

Figure 41: Booking valg!

6.3 Adgangskontroll

6.3.1 Logging av personer

Personloggings-systemet holder oversikt over hvor mange personer som til enhver tid er i leiligheten. Noen ting som kunne ha vært bedre, er at vi kun har brukt et system der kontrolleren hadde sendt et signal til serveren for å vise at den er kobla til nettverket. Om den sender et slikt signal, kan vi identifisere hvem som er i kollektivet til enhver tid. Da vet vi hvem som kan som kan være rammet dersom smitte oppstår i kollektivet og hvem som har vært i kollektive i den perioden smitten har brutt ut.

Videre kunne vi lagt inn et kontaktssystem der du kan legge til kontaktinformasjonen til de som besøker deg. Dermed kan du si hvem som har vært i kollektiv til enhver tid og sende de en melding dersom en smittesituasjon har oppstått. Det kan også være nyttig og legge til en effektiv måte lagre nye personen i kollektivet. En metode som kunne ha fungert (hadde det ikke vært for juridiske problemer) kunne vært å scanne studentkortet etter godkjent inngang fra beboerne. Adgangskontrollanlegget kunne da sjekket kortet opp mot en NTNU-database eller en ekstern database som kommunen kunne opprettet for dette formålet. Dermed kan man effektivt finne ut hvem gjesten er og lagre kontaktinformasjonen.

Vi kunne også montert en scanner for å låse opp døren på vei ut istedenfor trykknappen. En annen mulighet er å sette opp et system for varsling av alle som har besøk når sensoren for utgangen blir utløst, der alle som har besøk får en forespørsmål om deres besøk har dratt (og hvem av de om det er flere).

En kanskje mer interessant løsning ville være å utvikle en stempelmaskin som gir gjestene en QR-kode som refererte til når gjesten kom og hvem den på besøk til. En QR-kode som ville fungert som et besøkskort eller stempel som på konserter eller lignende og måtte bli skannet på vei ut. Dette ville gitt en bedre og sikrere kontroll på hvem som kommer og går, hvor mange gjester hver beboer har og når de eventuelt må dra. Med et slikt oppsett vil man kunne dekke alle nødvendige behov for logging og hadde vært et interessant konsept. Grunnet mangel på tid fikk vi ikke implementert dette i denne omgang men legger opp for dette i V2.0

6.3.2 Innlåsing og Kryptering

Innlåsing

På innlåsinga var arbeidsmengden mindre. Slik koden er lagt opp nå er det kun mulig å nullstille sin engen kode. Man burde kanskje ha en totrinns godkjenning dersom dette skal utføres, der vi tar bruk adgangskontrollpanelet. Et eksempel på dette er at på adgangskontrollpanelet vil det komme et varsel når du går til den spesielle menyen for å lage nytt passord, der du må verifisere din ID for å få endret passord, og ved hjelp av en av de andre sitt passord får du tilgang til å endre passordet. Noe som kan implementeres i videre utvikling for 2.0.

Du kan og legge inn en annen komponent enn en servo. Et stempel som kjører fram og tilbake kan være en bedre metode for å kjøre bevegelsen vi ønsker fram for en servo sine rotasjoner. Begge metodene vil fungere, men stempelet er mer driftsikkert og energibesparende.

Kryptering

Krypteringen er så sikker som vi kan få den. Det er vanskelig å fange opp nøklene digitalt over nettet for Ola Norman, men man har alltid en fare for fysisk intersept. Ondsinnede personer og grupper kan legges inn en egen RFID-leser, som leser verdiene i studentkortet ditt og stjeler de, noe som må forhindres på maskinvare siden med en anti-tamper sensor. Ellers er det et problem som ikke er lett å løse.

Til neste versjon kan f.eks face-id teknologi eller fingeravtrykk bli tatt i bruk, da dette er mye vanskeligere å kopiere [7]. Det beholder enkelheten ved å skanne kort, samtidig som det er mer sikkert. Det vil også gjør det å ha en kode eller en reset-mulighet unødvendig. Denne teknologien er likevel en mye dyrere løsning og vil kreve mer. Dette vil gjøre at anlegget vil øke i pris. Det vil bli mer sikkert, men man er avhengig av at kundegruppen er villig til å betale ekstra for dette.

6.3.3 Dørklokke

Dørklokkeløsingen ble veldig bra, enkel å bruke og driftsikker under den korte testingen. Fremover er det et behov for å kunne bytte ut navnene på dørklokkelisten for å slippe å laste opp et nytt program for hver person som flytter. Nå er navnene hardkodet inn som konsept, men det er ønskelig å implementere et felles navnesystem for smartkollektivet slik at navnene enkelt kan byttes ut i alle ledd av systemet. Det vil også være et ønske om å integrere en form for indikasjon for hvem som ringer på, enten ved hjelp av mikrofon eller kamera som vil fortelle brukeren hvem er ved døren. Denne integreringen er ikke en enkel og krever mye ekspertise og maskinvare som ikke var tilgjengelig under dette utviklingsprosjektet, og derfor ble ikke denne løsningen implementert i denne omgang, selv om det kan ansees som en viktig del av dørklokkesystemet.

6.4 Raspberry Pi - Server

Utviklingen av softwaren på serveren ga ønsket resultat, den utfører de oppgavene den ble tiltenkt i prosjektet og har hatt en stabil oppetid den uken den har vært i drift. Det er selfølgelig behov for å teste driftstiden over en lengre periode, men det der ikke blitt mulig å utføre innenfor tidsfristen på dette prosjektet.

Gjennom prosjektet har det kommet frem videre behov serveren burde håndtere. Fjernvarsling på mail til driftsansvarlig, et brukergrensesnitt for enkel håndtering av innstillinger på serveren og oppføring av nye beboere (eksempelvis en app).

På programsiden er det også allerede ideer til videreutvikling av bookingsystemet ved fremtidsbookingen, videre sikring av MQTT med kryptering av datatrafikken, inkludering av energilogging basert på brukernes bookinger, og komme med anbefalinger om pris dersom en student f.eks skulle skaffe seg en el-bil. Det er fremdeles mye å jobbe med, men resultatet legger grunnlaget for et spennende system med mye potensial.

7 Drøfting & konklusjon

7.1 Konklusjon

Prosjektet har ført fram til en totalløsning som ansees som gjennomtenkt og fornuftig. Systemets funksjon er som ønsket og den originale planen er i stor grad fulgt. Valget med å dele funksjonaliteten over flere enhetstyper har klare fordeler. Selv om løsningen totalt sett ansees som god, er der likevel aspekter ved prosjektet som ville blitt løst på en annen måte dersom det skulle gjenskapes.

Gruppen har gjennom prosjektet fått erfaring med utvikling av komplekse system, der fler enheter og plattformer skal kommunisere sammen for å skape en god totalløsning. Gruppen har gjennom prosjektet bygd opp kunnskap om lettvekts nettverks-kommunikasjon, adgangskontrollnøkler og kryptering, samt objektorientert programmering og automatisk uthenting av eksterne data fra relevante webtjenester. Gruppen har videre fått erfaring med utvikling av systemer for automatisering og forenkling av manuelle oppgaver.

7.2 Tilbakemeldinger

Gjennomføringen av prosjektet har opplevdes som interessant og utfordrende, og gruppedellemmene har hevet sin kompetanse på mange fagfelt.

Selv om prosjektet har opplevdes som interessant og spennende, er der flere kritikkverdige aspekter ved opplegget og planleggingen fra administrasjonen sin side. Først og fremst er prosjektet det ble lagt opp til gjennom prosjektbeskrivelsen, alt for omfattende og arbeidskrevende til å gjennomføre i et enkelt emne. En studieuke ansees som inntil 50 arbeidstimer fordelt over 30 studiepoeng. Emnet IELET1002 vekter 5 studiepoeng per semester, noe som vil resultere i overkant av 8 arbeidstimer per uke. Mellom publisingsdato av prosjektbeskrivelsen og innleveringsfrist var det i underkant av 9 uker, der en av ukene inkluderer påske. I løpet av 9 uker har et enkeltindivid derfor ikke mer enn ca 75 timer til rådighet til prosjektet, dersom det ikke skal gå på bekostning av andre emner. For å komme i mål med prosjektet, er det likevel gruppemedlemmer som har brukt nermere 300 timer bare de siste 6 ukene av prosjektet. Å legge opp til et så omfattende prosjekt rett før en eksamsperiode ansees derfor som uansvarlig, da det fører til at studenter må nedprioritere alle andre emner.

Videre er det kritikkverdig å lansere et prosjekt, men ikke publisere publisere prosjektbeskrivelsen før tre uker inn i prosjektet, slik at gruppene ikke har noe håndfast å arbeide med. I tillegg er det for dårlig å ikke formidle hva som er forventet med tanke på rapport, videopresentasjon, innlevering osv. før en uke før utsatt innleveringsfrist.

References

- [1] Aga, Lady. Overview <https://learn.adafruit.com/tmp36-temperature-sensor>. (Lastet ned 06.05.2021)
- [2] Andersen, Paul. LDR. <https://snl.no/LDR-motstand>. (27.09.2010)
- [3] Anderson, Jim. An Intro to Threading in Python, <https://realpython.com/intro-to-python-threading/>, (lastet ned 06.05.2021)
- [4] Andreas Ipsen, Student mentor ved NTNU Gløshaugen, Trondheim.
- [5] ArduinoGetStarted.com, 11.05.2021, [Internett, arduinogetstarted.com], <https://arduinogetstarted.com/tutorials/arduino-rfid-nfc>, lastet ned 21.05.2021
- [6] ArduinoGetStarted.com, 11.05.2021, [Internett, arduinogetstarted.com], <https://arduinogetstarted.com/tutorials/arduino-rfid-nfc>, (lastet ned 21.05.2021)
- [7] Espressif, [Internett, www.espressif.com] <https://www.espressif.com/en/products/devkits/esp-eye/overview>, lastet ned 07.04.2021 Espressif, [Internett, www.espressif.com] [Datablad Esp32 Wroom 32 pdf](#), (lastet ned 07.05.2021)
- [8] Fernando Koyanagi, 09.02.2018. [Intrenett, www.instructables.com], <https://www.instructables.com/ESP32-With-RFID-Access-Control/>, lastet ned 21.05.2021
- [9] Mark Stanley, Alexander Brevig, 18.09.2015 [Internett, www.arduino.cc], <https://playground.arduino.cc/Code/Keypad/>, lastet ned 20.05.2021
- [10] Ravi Teja, 24.02.2021, [Internett, www.electronicshub.org], <https://www.electronicshub.org/esp32-oled-display/> Lastet ned 16.05.2021
- [11] Raspberrypi. Hardware. <https://www.raspberrypi.org/documentation/hardware/raspberrypi/README.md> (lastet ned 10.05.2021)
- [12] The HiveMQ Team. Introducing the MQTT Protocol, <https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/>, (publisert 12.01.2015, lastet ned 07.05.2021)
- [13] techtutorialsx.com, 10.05.2018, [Internett, techtutorialsx.com], <https://techtutorialsx.com/2018/05/10/esp32-arduino-mbed-tls-using-the-sha-256-algorithm/>, lastet ned 27.05.2021
- [14] <http://esp32.net/> (lastet ned 10.05.2021)
- [15] <https://components101.com/motors/mg90s-metal-gear-servo-motor> (lastet ned 07.05.2021)
- [16] Aliexpress <https://www.aliexpress.com/item/32914503301.html> (lastet ned 07.05.2021)

- [17] Openweathermap <https://openweathermap.org/> (10.05.2021)
- [18] <https://randomnerdtutorials.com/what-is-mqtt-and-how-it-works/> (10.05.2021)
- [19] <http://www.steves-internet-guide.com/mqtt-topic-payload-design-notes/> (08.05.2021)
- [20] <https://pubsubclient.kolleary.net/> (16.05.2021)
- [21] Bodmer - TFT eSPI (17.05.2021)
https://github.com/Bodmer/TFT_eSPI

8 Vedlegg

1. Vedlegg 1 I2C kommunikasjon [I2C.pdf](#)
2. Vedlegg 2 Esp32 Datablad [Esp32datasheet.pdf](#)
3. Vedlegg 3 Tssp4038 Datablad [tssp40.pdf](#)
4. All kode ligger i mappa "Programmer" på [github](#)