NTNU
Kunnskap for en bedre verden

Institutt for elektroniske systemer

IELET2001 - Datakommunikasjon

Smart Hønsehus

Gruppe 10

Forfattere:
Kato Mørland Abrahamsen, Magnus Storvold Hagen, Erlend Haldorsen,
Aleksander Opheim-Larsen & Erlend Nesheim Steinnes

11. mars 2024

Sammendrag

Prosjektet er en en helhetlig løsning for overvåkning og styring av et smart hønsehus, med formål å sikre hønenes velferd til enhver tid. Løsningen implementerer kjente og muliggjørende teknologier for å danne et omfattende produkt. Dataoverføring til Raspberry Pi og Node-RED skjer trådløst via MQTT og WiFi. Lydsensoralarm bruker PyAudio for deteksjon og Twilio for varsling. Automatisk mating stilles inn i UI på Node-RED, og viser matingshistorikk. Luftkvalitet måles med en BME680-sensor, og vannkvalitet blir målt med en pH-sensor. Temperatur kan styres og overvåkes. Kameraovervåkning benytter ESP32CAM. Brukeren kan bruke RFID og kode for å få adgang til hønsehuset. Antall egg kan loggføres og innsamlet data kan inspiseres i terminalen og i brukergrensesnittet på løsningens nettside.

Innhold

Figurer

Nomenklatur

DNS : Domain Name System

DUC : Dynamic Update Client

$FTDI$: Future Technology Devices International Limited

$HTML$: Hypertext Markup Language

$HTTP$: Hypertext Transfer Protocol

$I2C$: Inter-Integrated Circuit

IP : Internet Protocol

LED : Light Emitting Diode

$MQTT$: Message Queuing Telemetry Transport

$PICO$: Peripheral In Controller Out

$POCI$: Peripheral Out Controller In

QoS : Quality of Service

$RFID$: Radio Frequency Identification

RPi : Raspberry Pi

SMS : Short Messaging Service

SPI : Serial Peripheral Interface

UI : User Interface

VOC : Volatile organic compound

$WiFi$: Wireless Fidelity

Terminologi

MQTT: Nettverksprotokoll for dataoverføring

SSID: Navn på lokalt nettverk

DNS: Internettjeneste for å knytte et navn til en IP-addresse

Mikrokontroller: Integrert krets med prosessor og minne.

ESP32CAM: ESP32 mikrokontroller med innebygd kameramodul

HTTP: Kommunikasjonsprotokoll som brukes til å utveksle informasjon.

Base64: Binær data representert i ASCII tekst.

KortUID: Kortets unike identifikasjons nummer.

Buss: Kommunikasjon hvor data fra flere komponenter sendes på samme høyhastighetslinje

VOC: Flyktige organiske forbindelse. Forurensing av luft.

Hash: Algoritme for å konvertere data for sikker lagring av passord.

1 Innledning

Dyrevelferd er viktig i matindustrien og i private husdyrhold. Fjørfe, spesielt høns, utgjør en stor andel av dette markedet, både nasjonalt og globalt. “Dyr skal behandlast godt og vernast mot fare for unødige påkjenningar og belastningar”. (Landbruks- og matdepartementet 2009). For å sikre velferden til hønene, kreves betydelige ressurser. Bakgrunnen for valget av prosjektoppgaven er å utvikle en smart og innovativ løsning som kan bidra til at disse kravene møtes, og å bedre levestandarden for høns.

Prosjektets mål er å utvikle et automatisert hønsehus som sørger for at kravene til livskvalitet, derav kvalitet på luft, vann og mat møtes. Et enkelt betjent brukergrensesnitt et viktig for å sikre funksjonaliteten til systemet. En nettside med brukergrensesnitt for dataovervåking og styring er ønskelig. Dette vil i tillegg muliggjøre at antall egg funnet i hønsehuset kan sees i sammenheng med leveforhold. I andre rekke er overvåkning og deteksjon av rovdyr, og andre eksterne farer, et viktig punkt i smart-hønsehuset.

2 Teori

2.1 ESP32

ESP32 er en programmerbar mikrokontroller med 32-bit mikroprosessor. Mikrokontrolleren bruker WiFi og Bluetooth over 2.4GHz-båndet. Den kan programmeres med språket C++ (Arduino C).

2.2 RPi

RPi er en ettkortsdatamaskin som kjører et Linuxbasert operativsystem. RPi bruker lite strøm, er energieffektiv, og egner seg til serverbruk med kontinuerlig drift.

2.3 FTDI programmer

FTDI programmer er en komponent som brukes for USB-til-seriell konvertering. Ved å koble RX og TX utgangene på FTDI programmeren til henholdsvis UOT og UOR på ESP32CAM kan det lastes opp kode til mikrokontrolleren.

2.4 Edge Computing

Edge Computing handler om å prosessere data nærmest mulig kilden. Ved edge computing blir unødvendig dataoverføringer unngått og systemer kan fungere mer pålitelig og effektivt.

2.5 Kommunikasjon over MQTT

MQTT er en effektiv meldingsprotokoll som baserer seg på publisering og abonnering på emner. QoS er kvaliteten på sending av meldinger over MQTT. QoS 0 sender maks en melding fra klient til megler, QoS 1 sørger for at minst en melding blir levert, og QoS 2 garanterer kun en melding sendt og mottatt, og protokollen har en header på 2 byte og en binær payload.(Cope 2018). Mosquitto er brukt som megler, og er installert på RPi.

2.6 DNS

DNS knytter domenenavn til en IP-adresse. Når brukeren skriver inn en URL blir den assosierte IP-adressen forespurt i et hierarkisk system av servere (Kurose og Ross 2016). Rotserveren forespør TLD-serveren, som henviser til serveren der adresse og domenenavn er koblet. Adressen blir cachet og datoene sammenlignes, for å sjekke om klienten har nyeste versjon.

2.7 Twirlio

Twirlio er et kommunikasjonsverktøy som brukes for å sende eller motta telefonsamtaler og SMSer. Det kan implementeres i både Python og Node-RED med tilleggsbibliotek.

2.8 Node-RED

Node-RED er et flow-basert utviklingsverktøy på nett, bygget på Node.js (OpenJS 2023). Programmet er designet for integrasjon av IoT-enheter, og knytter sammen noder som styrer datastrømmen.

2.9 SPI

SPI, en seriell kommunikasjonsprotokoll som benytter fire tilkoblinger: SCK, POCI, PICO, og CS. I denne protokollen sender mikrokontrolleren et konstant klokkesignal (SCK) til SPI-enheten den er tilkoblet, noe som muliggjør synkronisert kommunikasjon mellom de to enhetene på hver sin kommunikasjonsbuss (POCI og PICO) (Grusin 2023). CS(Chip Select) brukes for å angi hvilken periferienhet som skal motta eller sende data. Ved bruk av SPI protokollen kan flere enheter kobles på samme buss, og CS brukes til å fastsette hvilken enhet som snakkes til eller fra. Dette aspektet av SPI tillater "full-duplex" kommunikasjon, hvor data kan sendes og mottas samtidig.

2.10 I2C

I2C er en seriell kommunikasjonsprotokoll som bruker to tilkoblinger, klokke og data over buss. Den fungerer ved at en master styrer kommunikasjonen og genererer et klokkesignal som sendes på bussen (SFUPTOWNMAKER 2023). Slaver tilkobler seg bussen, og adressering sørger for at mange slaver kan tilkobles en master på samme buss. Den største fordelen med I2C er få (to) tilkoblingspunkter.

2.11 RFID

RFID er en teknologi som ofte brukes i adgangskontrollkort og for kontaktløse betalinger med bankkort. I denne teknologien sender en RFID-leser radiosignaler gjennom en antenne. Signalene blir mottatt av RFID-taggen, som bruker energien fra signalet til å aktivere den interne elektronikken, deretter sender taggen data tilbake til leseren (Amsler 2021). Denne prosessen gjør det mulig for systemet å identifisere og behandle informasjonen knyttet til taggen, som kan inneholde identifikasjonsdata eller annen relevant informasjon. I dette prosjektet blir KortUID nummer hentet fra kortet, for å identifisere kortet.

2.12 Rotasjonsgiver

Rotasjonsgiver(eng. Rotary encoder) er en roterende innretning som bruker spenning og klokkesignal til å måle vinkel, dreieretning og dreiehastighet (The Geek Pub 2019). Utgangssignalet som blir generert kan brukes til å manuelt justere og styre verdier. Rotasjonsgivere kan også kombineres med trykknapp for å gi ytterligere bruksmuligheter. Funksjonalitet må i utgangspunktet tilegnes manuelt.

3 Metode

3.1 Organisering av arbeid

Ved oppstart ble arbeidsoppgaver fordelt til gruppemedlemmer etter interesse. Hvert gruppemedlem er ansvarlig for minst en undermodul. Gjennom regelmessige møter er det sørget for at alle har innsikt i hverandre sitt arbeid. Oppgaver som er kritiske for gjennomføring av andre arbeidsoppgaver, eksempelvis oppsett av RPi server, ble prioritert. Dette har bidratt til at alle medlemmene har vært i stand til å hjelpe hverandre, og dermed sikre en god og jevn fremdrift.

3.2 Node-RED

I prosjektet brukes Node-RED til å sette opp et brukergrensesnitt der brukeren kan interagere med funksjonalitetene. Node-RED kjøres på RPi, som er koblet til ett lokalt nettverk. På gjeldende ruter er det åpnet for portviderekobling. Port 1880-1883 er konfigurert til å være åpne gjennom internettleverandør. Dette muliggjør tilgang til Node-RED og kommunikasjon over MQTT fra utenfor det lokale nettverket.

Innloggingssiden til Node-RED er konfigurert i settings.js filen på RPi. Det er lagt til to kontoer: "admin" med redigeringstilgang, og "guest" med lesetilgang, og samme passord. Hashen til passordene er generert gjennom kommandopromptvinduet i RPi. Domenenavnet "cluckwatch360.ddns.net" administreres av en gratis DNS-tjeneste, "noip.com". IP-addressen blir dynamisk oppdatert av programmet sin dynamiske oppdateringsklient (DUC).

MQTT in-noder tar inn verdier fra programmer som kjører på ESP-enheter. Data sendes til UI-siden, som er delt inn i flere seksjoner. "Egg statistikk" har historikken over sensormålinger sammen med antall egg samlet inn i en graf. "Mating" og "Lydalarm" viser historikken over tidspunktene for de respektive hendelsene. Temperatur og luftfuktighet kan leses av under "Luft".

3.3 Luft, ventilasjon, og vann

Sensorverdier for luft måles gjennom BME680, og sendes over I2C-protokollen til ESP32. Noen av de innsamlede dataene behandles lokalt i form av grafisk visning av temperatur på OLED-skjerm, innstilling av ønsket temperatur ved bruk av rotasjonsgiver og utregning av luftkvalitet. All innsamlet data blir sendt trådløst til server ved gitte, innstilte intervall. Se Vedlegg 7.2 og 7.3 for koblingsskjema og flytdiagram.

Mikrokontrolleren som brukes i varme-/kjølebatteriet spiller en nøkkelrolle i temperaturreguleringen. Denne enheten henter inn data om den reelle temperaturen og den ønskede temperaturen, som blir målt av styringsenheten for luft- og ventilasjonssystemet. Mikrokontrolleren analyserer deretter temperaturforskjellen. Avhengig av om det er et negativt eller positivt avvik, aktiveres varme-/kjølebatteriet tilsvarende for enten å varme opp, eller kjøle ned luften - og forsøker således å opprettholde den instilte temperaturen. Kjøling vises som blå LED, og varming vises som rød LED. Vedlegg 7.5 og 7.4 viser koblings- og flytskjema for varme- og kjøle-batteriet.

Kalibrering av sensor for pH og temperatur i vann gjøres ved å teste målinger imot kjente verdier. Nivåmåleren krever ikke kalibrering. Gjennom stiv spenningsdeling vil 5V spenningsforsyning til

sensorer omgjøres til 0-3.3V for analog avlesning av ESP32. Denne rekkevidden blir lineært transformert til måleområdet til sensoren. Variabelen til de transformerte verdiene blir omgjort til en streng og publisert ved faste intervaller. Programmet består av en kombinasjon av eksempelkode for WiFi og MQTT fra Arduino. Koblings- og flytskjema er opptegnet i Vedlegg 7.12 og 7.13.

3.4 Terminal

Terminalen fungerer som en sentralenhet i hønsehuset. Den har funksjonene: loggføring av egg som samles i hønsehuset, kontroll av adgang til området, samt visning av data fra ulike sensorer. Terminalen er satt sammen av 4 komponenter, en RFID-leser for identifikasjon, en LCD-skjerm for visuell informasjon, et 4x4-tastatur for brukerinteraksjon, og en ESP32-mikrokontroller som styrer operasjonene. LCD-skjermen og RFID-leseren er tilkoblet ESP32 mikrokontrolleren med SPI-protokollen, koblingsskjema vises i vedlegg 7.6.

Programvaren som driver terminalen er strukturert som en tilstandsmaskin. Dette betyr at terminalens oppførsel er avhengig av dens nåværende tilstand og brukerinput. Denne tilnærmingen sikrer en effektiv og organisert håndtering av forskjellige operasjoner. For en mer detaljert forståelse av terminalens funksjonalitet og oppbygging, se flytskjemaet i vedlegg 7.7.

Hjemskjerm 1 Hjemskjerm 2 Sensor skjerm

Figur 1: Hvordan noen av skjermene ser ut

Hver tilstand i koden har sitt eget skjermbilde. Skjermbildene designes i Photoshop, og bildene gjøres så om til en matrise med pikselverdier som mikrokontrolleren kan sende til skjermen. Hvordan noen av bildene ser ut, vises over i figur 1

Terminalen har også en funksjon for adgangskontroll. Denne funksjonen er designet slik at brukeren skanner sitt RFID-kort på toppen av terminalen. Programmet vil deretter sammenligne KortUID bit for bit med en liste over kjente kort. Hvis kortet gjenkjennes, vil skjermen be brukeren om å taste inn sin firesifrede personlige kode. Programmet kontrollerer så om koden matcher med den som er tilknyttet det gjenkjente RFID-kortet. Hvis både kortet og koden er korrekte, vil adgang bli innvilget. Hvis ikke, vil døren forbli låst.

Kapslingen til terminalen ble designet i 3D og 3D-printet ut som en prototype. Bruk av 3D-printing for denne typen prototyping representerer et kostnadseffektivt alternativ sammenlignet med andre produksjonsmetoder. I tillegg kan 3D-modellen sendes til eksterne produsenter, slik at kapslingen kan støpes i plast dersom det blir aktuelt med masseproduksjon senere.

3.5 Lydsensoralarm

Lydsensoralarmen er utviklet for å detektere potensielle rovdyr og farer. Ideen bak lydalarmen er å lytte til lydnivået og utløse en alarm dersom nivået går over en viss grense. Programmet er skrevet i Python, med bruk av flere tilleggsbiblioteker for å muliggjøre funksjonaliteten.

PyAudio er et viktig bibliotek som brukes for å ta kontinuerlige lydmålinger. PyAudio tar målinger fra én lydkanal, med samplingsrate på 44100Hz. I koden blir volumnivået skalert fra 0 til 100, slik at det kan defineres en grense for støy. Gjennom if-setninger sjekker koden om lydnivået er over støygrensa. Antall høye lyder telles, samt antall sekunder fra første høye lyd. Dersom det oppstår flere høye lyder enn grensen som er satt, samtidig som tiden det tar er under tidsintervallet, vil alarmen gå og kunden vil bli varslet. Se Vedlegg 7.8 for programlogikk.

For å kunne varsle kunden brukes Twilio. I Pythonkoden vil Twilio sende SMS og ringe dersom alarmen går. Twilio tilbyr egendefinerte SMSer og telefonsamtaler ved bruk av text-to-speech og syntetisk stemme. I varslingsfunksjonen blir en egendefinert melding lagt inn der blir sagt at det er en potensiell fare, samt tidspunktet for denne faren. For å unngå at alarmen går flere ganger på kort tid, er det satt en time.sleep() i programmet på 30 min etter at alarmen går.

Alarmtidspunkt vil også bli sendt til brukergrensesnittet i Node-RED. For å oppnå dette integreres MQTT i Pythonkoden. Paho-MQTT er biblioteket som blir brukt for å oppnå dette. Ved alarmutløsning vil det bli publisert en melding over MQTT slik alarmtidspunktet vises i brukergrensesnittet.

3.6 Kameraovervåkning

Det er utviklet to løsninger for video-overvåkning til Node-RED dashbordet med ESP32CAM. Den første løsningen bruker HTTP og en webserver som kjører lokalt på nettet. Den andre løsningen sender videostrømning direkte til dashbordet med MQTT.

For å programmere ESP32CAM er det i dette prosjektet en bro, FTDI programmerer, mellom ESP32CAM og pc-en, for oppkobling se vedlegg 7.9. Kamera brukes til å fange bilder og sende binær bildedata - til en HTTP server, eller via MQTT. Noe som midlertidlig egner seg godt til kommunikasjon via MQTT er servomotorene, og det er brukt i begge video-løsningene. De beveger kamera i x og y retning etter beskjed fra bruker i Node-RED dashbordet.

På kameraet er det koblet to servomotorer slik at kameraets posisjon kan styres i x og y retning. I Node-RED dashbordet gjøres dette manuelt med å skru på to slidere. I tillegg kan bruker trykke på en bryter i dashbordet for å aktivere selvstyrt modus, dette får kamera til å panorere automatisk i x retning.

Å sende bildedata til MQTT via det globale internett er ressurskrevende for ESP32CAM og MQTT/RPi. Bilder tas med ESP32CAM og er opprinnelig en binær buffer. Å sende bilder gjennom MQTT tar mye maskinkraft av ESP32CAM, og for å redusere belastningen er det brukt client.publish_P() istedenfor client.publish() i koden. Dette gjør at ESP32CAM bruker flash-minne istedenfor ram. Bufferen gjøres om til en base64 streng i Node-RED for å enklere vise bildeinnholdet i dashbordet.

Endringer av kamerainnstillinger gjøres også i dashbordet med denne løsningen. Når brukeren jus-

terer på UI-nodene i dashbordet endres verdier i et JSON-dokument, som blir sendt til ESP32CAM gjennom MQTT. Verdiene i det dynamiske JSON-dokumentet setter innstillingene i kameraet.

I den andre løsningen benyttes en webserver som mottar bildedata fra kamera. Store deler av koden knyttet til webserveren er hentet fra Espressif Systems sin dokumentasjonskode av ESP32CAM. På webserveren kan kamerainnstillingene endres på samme vis som for MQTT løsningen. Videostrømmen hentes med en HTML template node for å integrere i Node-RED dashbordet, og ikke bare på webserveren. Kriteriet for at dette skal funke er at ESP32CAM og RPi kobles til det samme nettverket.

3.7 Matingssystem

Mating av hønsen bestemmes i brukergrensesnittet. I Node-RED er det satt opp en "ui-time-scheduler", slik at dato og tid for mating kan innstilles for regelmessig mating. Ved matingstidspunkt blir det sendt melding til matings-ESP enheten, som deretter simulerer matingen med en LED. Etter dette vil ESP32 sende melding tilbake over MQTT, slik at matingstidspunkt vises i brukergrensesnittet. I Node-RED blir tidspunktet hentet og lagt til i en liste, og deretter lagt til på brukergrensesnittet i en vertikal liste. Se Vedlegg 7.11 for programstrukturen på matings-ESP enheten.

4 Resultater

4.1 Brukergrensesnitt i Node-RED

En sentral del av sluttløsningen som ble utviklet er implementasjonen av Node-RED. Node-RED gjør det mulig å utvikle et brukergrensesnitt der brukeren kan interagere med hønsehuset, ved å se og styre tilhørende funksjonaliteter. Det er satt opp en portviderekobling til RPi, og konfigurert en innloggingsside med to registrerte brukere. Brukeren har innsyn på sensorverdier for klima, samt antall egg funnet i hønsehuset. Dette blir vist i en sammenhengende graf, slik at korrelasjon kan analyseres. Det tilbys i tillegg kameraovervåkning og varsling fra lydsensoralarm. Brukeren har også mulighet til å innstille et regelmessig matingstidspunkt til hønene.

4.2 Luft, ventilasjon og vann

Måling av vannkvalitet er gjennomført med en sensor for vannivå, og en sensor for pH og vanntemperatur. Signal fra sensorer mottas som et analogt spenningssignal. En mikrokontroller fungerer som en enkel sensornode, og publiserer med intervall på 5 sekunder verdi til RPi for avlesning i brukergrensesnittet. Sensor for pH og temperatur gir tilfeldige feil i målinger. Nivåmåleren viser det reelle nivået fra en nedre til ei øvre grense.

Løsningen for luft, varme og ventilasjonsstyring er delt opp i flere deler med flere mikrokontrollere. Styreenheten måler lufttempereratur, luftfuktighet, lufttrykk og forurensing (VOC). Her stilles også inn ønsket temperatur. Dette gjøres ved bruk av en rotasjonsgiver for manuell innstilling. En OLED-skjerm viser den reelle temperaturen og innstilt ønsket temperatur, som endres ved betjening av enheten. Maksimal og minimal temperaturverdi er begrenset til fornuftige verdier for å hindre at feilbetjening utsetter dyrene for fare. Basert på all innsamlet data utregnes en samlet verdi for luftkvalitet. Alle dataene blir sendt til en server hvor de blir grafisk fremstilt i brukergrensesnittet. Dataene blir også sendt til terminalen hvor det blir grafisk fremstilt på LCD-skjerm.

4.3 Terminal

Virkemåten til terminalen er at en bruker kan komme seg inn i bygget ved å benytte et kjent RFID kort og kode som tilhører kortet. Kjente KortUID og pin-koder er lagret på tilhørende ESP32. Inne i hønsehuset kan brukeren benytte terminalen til å se sensordata, hvor viktig data er fargekodet i henhold til verdier (grønn = bra, rød= dårlig). Antall egg funnet i hønsehuset kan tastes inn, og sendes til webserver som lagrer antallet sammen med loggført sensordata.

Under utviklingen av prosjektet oppstod det en mangel på viktige komponenter. Terminalens funksjoner ble kombinert med adgangskontrollenhetens oppgaver for å effektivisere ressursbruken. Til tross for utfordringene det medførte, sikret denne tilnærmingen at prosjektets kjernebehov ble ivaretatt, samtidig som det demonstrerte prosjektets fleksibilitet under begrensede forhold.

4.4 Lydsensoralarm

Løsningen som ble utviklet for lydsensoralarmen er et fungerende Pythonprogram som kjører på en Windows PC. Datamaskinen er installert i hønsehuset, og benytter seg av en mikrofon til å ta kontinuerlige lydmålinger. Dersom lydnivået går over en viss grense vil hendelsen bli lagret som støy. Hvis antall hendelser går over en grense, innen en satt tid, vil alarmen bli utløst. Disse tre parametere kan bruker velge. Når alarmen blir utløst vil bruker bli varslet på telefon, der alarmtype og alarmtidspunkt blir lest opp ved bruk av syntetisk tale. En SMS vil også bli sendt, samt opplasting til brukergrensesnittet med MQTT.

4.5 Kameraovervåkning

Det ble utviklet to løsninger for kameraovervåkning. Løsningene benytter to ulike applikasjonsprotokoller - overføring via MQTT, og overføring via HTTP. Sending av bildedata gjennom MQTT gjør at det kan sendes bildedata fra kamera til Node-RED, uavhengig av hvilket nettverk ESP32CAM er koblet til. Overføring med HTTP blir gjort ved å lage en webserver lokalt på nettet. Av den grunn er RPi og ESP32CAM tilkoblet det samme nettverket for å hente videostrømning med en template node i Node-RED.

Videostrømming ved bruk av MQTT viser en redusert ytelse på ESP32CAM. Med HTTP protokollen er det enklere og raskere å få bildedata opp på dashbordet i Node-RED.

Samtidig med videostrømning brukes styring av servomotorene koblet til kamera. Med webserver og HTTP klarer ESP32CAM å håndtere oppgavene raskere og bedre.

4.6 Matingsystem

Løsningen for matingssystemet gjør at kunden kan bestille mating via brukergrensesnittet. Regelmessig matingstidspunkt kan innstilles, og matingen vil simuleres ved at en LED lyser 5 sekunder ved det angitte matingstidspunktet. Etter at matingen er simulert vil bekreftelse på mating publiseres i brukergrensesnittet, slik at matingstidspunktet ses i en vertikal liste. Reset-knappen gjør at historikken tømmes.

5 Diskusjon

5.1 Sikkerhet i Node-RED

Sikkerhetsrisikoen ved portviderekobling er delvis adressert med oppsett av brukerkontoer. Det ble oppdaget at flows forsvant underveis i prosjektet, og etter adgangskontroll ble impementert var ikke dette lenger ett problem. For å styrke sikkerheten ytterlige kan tilgang kun gis til dedikerte enheter via IP-adresse, og sette en grense for antall innloggingsforsøk. Sendte data antas å ikke være sensitive og det er derfor ikke nødvendig med kryptering.

5.2 Luft, ventilasjon og vann

Måling av pH og temperatur viser tilfeldige målefeil, og vurderes som mest nyttig for å få en grov oversikt. Kvaliteten på sensoren vil være avgjørende for hvor gode målingene blir. Krav om nøyaktighet bør måles opp mot kostnad for å finne en hensiktsmessig balanse. Nivåsensoren gir repeterbare resultat og vurderes som nøyaktig. Kommunikasjon over MQTT med QoS 1 garanterer at data blir mottatt minst en gang, og er en balanse mellom pålitelige overføringer og hastighet med sending. For videre automatisering kan sensordata senere sammenlignes med ønskelige verdier og reguleres. I utgangspunktet kan vannmålinger settes i sammenheng med mating, og programmene kan utføres på samme mikrokontroller. Modulene er skilt med tanke på skalerbarhet.

Luftstyreenheten fungerer etter hensikt. Den måler reelle verdier i luften av trykk, fuktighet, VOC og temperatur. Rotasjonsgiveren fungerer bra, og dens funksjonalitet er intuitiv. I en fast installasjon, hvor komponentene er ordentlig terminert vil betjeninga ytterligere forbedres. Data sendes med intervall på fem sekunder, for å minimere strømforbruk. Dette gjør at det er litt forsinkelse i overføringa. For bruk i et virkelig anlegg, kan datasendinga foregå enda sjeldnere, da et kjøle- og varmesystem uansett er veldig tregt. Grafisk fremstilling av data på nett er heller ikke avhengig av så hyppig datasampling. For kjøle/varmebatteriet er proposjonalregulering av temperatur greit for formålet, men kan forbedres med andre reguleringsmetoder. For å kunne benytte bedre reguleringsmetoder er det nødvendig med ekte varme- og kjøleelement for innstilling og tilbakekobling.

5.3 Terminal

Terminalen burde ha vært delt i to. Den ene delen, utstyrt med RFID, LCD og tastatur, brukes til å få tilgang til bygningen. Den andre terminalen, som har LCD og tastatur, brukes til å loggføre egg, vise sensordata og mer, inne i hønsehuset. Terminalens funksjoner ble kombinert med grunnlag i tidsbegrensing og komponentmangel.

Programmet for adgangskontrollsystemet er basert på en kombinasjon av KortUID og en tilhørende PIN-kode, som begge er lagret som variabler i mikrokontrollerens programvare. Denne metoden gjør det utfordrende å administrere brukeradgang, da det å legge til eller fjerne brukere krever manuell endring i programmet. En mer effektiv løsning ville vært å lagre kjente KortUID-er og deres tilsvarende koder på en sentral server. Dette ville ikke bare forenkle brukeradministrasjonen, men også muliggjøre implementeringen av flere adgangskontrollenheter og tilby mer skreddersydde adgangsnivåer til ulike brukere.

I implementeringen av SPI-kommunikasjonen ble det benyttet dedikerte SPI IO-porter på ESP32 mikrokontrolleren. Tidlige forsøk på å benytte en felles SPI-buss for både RFID-leseren og LCD-skjermen førte til kommunikasjonskonflikter. Dette problemet ble spesielt merkbart under testfasen, hvor uforutsette signalforstyrrelser påvirket systemets pålitelighet.

For å adressere disse utfordringene, ble det besluttet å separere RFID- og LCD-enhetene på to forskjellige SPI databusser. Denne tilnærmingen sikret mer stabil og konfliktfri kommunikasjon mellom mikrokontrolleren og de to enhetene. Den endelige konfigurasjonen av terminalen, inkludert den spesifikke koblingen av komponentene, er detaljert i det vedlagte koblingsskjemaet, se Vedlegg 7.6.

5.4 Lydsensoralarm

Programmet er utviklet i Windows, og må derfor kjøres på en Windows PC. Det ville vært ønskelig å kjøre denne koden på RPI, da RPI er kontrollenheten i hønsehuset. Forsøk i å kjøre denne koden på RPi resulterer i problemer, da RPi kjører Linux operativsystem. Flere av bibliotekene og funksjonene støttes ikke, og mye av koden måtte endres. På grunn av begrenset tid og kapasitet, ble det valgt å kun la programmet kjøre på Windows. Dette er noe som hadde blitt prioritert videre ved forbedring av prosjekt.

Videre kunne funksjonaliteten til lydsensoralarmen utvides ved at den programmeres til å gjenkjenne bestemte lyder, eller bestemte kombinasjoner av frekvenser. På denne måten kan forskjellige eksterne farer kategoriseres ved bruk av akustisk analyse. Kunstig intelligens kan brukes til dette og til å gjenkjenne lyden til ulike dyr.

5.5 Kameraovervåkning

Videostrømming på lokalnett med HTTP-protokollen egner seg bedre til å sende en kontinuerlig bildestrøm. For å få til dette må RPi og ESP32CAM være tilkoblet det samme logiske nettverket. Bildedata blir sendt til en webserver, og deretter vist i Node-RED-dashbordet.

Med videostrømming utenfor det lokale nettet blir kommunikasjon gjennom MQTT brukt. Fordelen med dette er direkte sending fra ESP32CAM over til Node-RED-dashbordet. Likevel går dette utover hastighet, da ESP32CAM ikke er direkte egnet til dette.

Begge metodene fungerer, men det er enklere å sende store mengder bildedata gjennom HTTP enn det er gjennom MQTT med komponentene som er brukt. Med kraftigere maskinvare eller flere ESP32er kunne MQTT løsningen fungert bedre. Blant annet kan det brukes en egen ESP32 for styring av servoene, mens ESP32CAM bare prosesserer bilder.

5.6 Matingssystem

Løsningen for matingssystemet kan ta i bruk flere parameterverdier innstilt i brukergrensesnittet. Matingen er relativt simpel, da det kun er matingstidspunkt som innstilles. Ønskelige parameterverdier er eksempelvis mengden med fôr, og hvilken type fôr. ESP-en kan gi ulike simuleringer ut ifra mengde med fôr og hvilken type. Mengden fôr kan simuleres med lengden LED lyser, og ty-

pen fôr indikeres med forskjellige LED. Med denne implementeringen bør også matingshistorikken inneholde disse parameterene.

For å utvikle en helhetlig løsning, er det ønskelig å erstatte LED simuleringen med en fysisk fôrautomat. Denne ville bestått av en forbeholder med en dispenseringmekanisme. Dispenseringen kunne vært oppnådd ved en servomotor som brukes til åpne og lukker en luke, eller en rotasjonsskive som lar mat falle gjennom ved bestemte vinkler. Fokuset i prosjektet er derimot datakommunikasjonen, og det er derfor lagt mest vekt på utvikle et fungerende brukergrensesnitt der utveksling av data til ESP32 oppnås.

6 Konklusjon

Utviklingen av et funksjonelt grunnlag tidlig i prosessen muliggjorde videre utvidelse av funksjonalitet på modulene. En positiv side med dette er at systemet er skalerbart, lett å videreutvikle, og lett å replikere for bruk i andre tilsvarende problemer.

Målet ved utviklingen av smarthønsehuset var først og fremst å sikre velferden til hønene ved måling av inneklima, samt detektere og varsle om eksterne farer.

Den utviklede totalløsningen tar i bruk sensorer og mikrokontrollere til video- og lydovervåkning, samt måling og behandling av luft og vannverdier. Verdiene brukes så til å kontrollere forholdene i hønsehuset, og sammenligne med statistikk over egginnsamling. Systemet i seg selv garanterer ikke at hønenes velferd blir bedre, men det gir muligheter og verktøy som kan hjelpe for å oppnå dette. Sluttløsningen som ble utviklet oppfyller kravene og målene som ble satt ved starten av prosjektet.

Det er deler av systemet som kan forbedres. Systemet er avhengig av en operatør som holder anlegget under oppsyn. En forbedring kan derfor være å implementere helautomatiske løsninger, der automatisk styring av vann og luftkvalitet implementeres. Dette vil bedre sikkerheten da det minsker muligheten for menneskelige feil. Noen av enhetene og komponentene i systemet kan settes sammen til en enhet, dette gjelder i hovedsak terminal og temperaturstyring. Denne løsningen vil i tillegg bruke færre komponenter.

Gjennom bruk av MQTT og WiFi blir det oppnådd pålitelig dataoverføring til RPi. Hovedfokuset i prosjektet er datakommunikasjonen, og prioritering er dataoverføring og -behandling, ikke det fysiske anlegget. Prosjektet er i hovedsak et konseptbevis.

Referanser

Amsler, Sarah (2021). *RFID (radio frequency identification)*. URL: https://www.techtarget.com/iotagenda/definition/RFID-radio-frequency-identification (sjekket 08.12.2023).

Cope, Steve (2018). *Understanding the MQTT Protocol Packet Structure*. URL: http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/#:~:text=MQTT%20is%20a%20binary%20based,encoded%20as%20UTF%2D8%20strings. (sjekket 15.11.2023).

Grusin, Mike (2023). *Serial Peripheral Interface (SPI)*. URL: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all (sjekket 08.12.2023).

Kurose, James F. og Keith W. Ross (2016). *Computer Networking, A Top-Down Approach 7th edition*. Pearson. Kap. 2.4.1 Services Provided by DNS, s. 135. ISBN: -10: 0-13-359414-9.

Landbruks- og matdepartementet (2009). *Lov om dyrevelferd*. URL: https://lovdata.no/dokument/NL/lov/2009-06-19-97 (sjekket 10.12.2023).

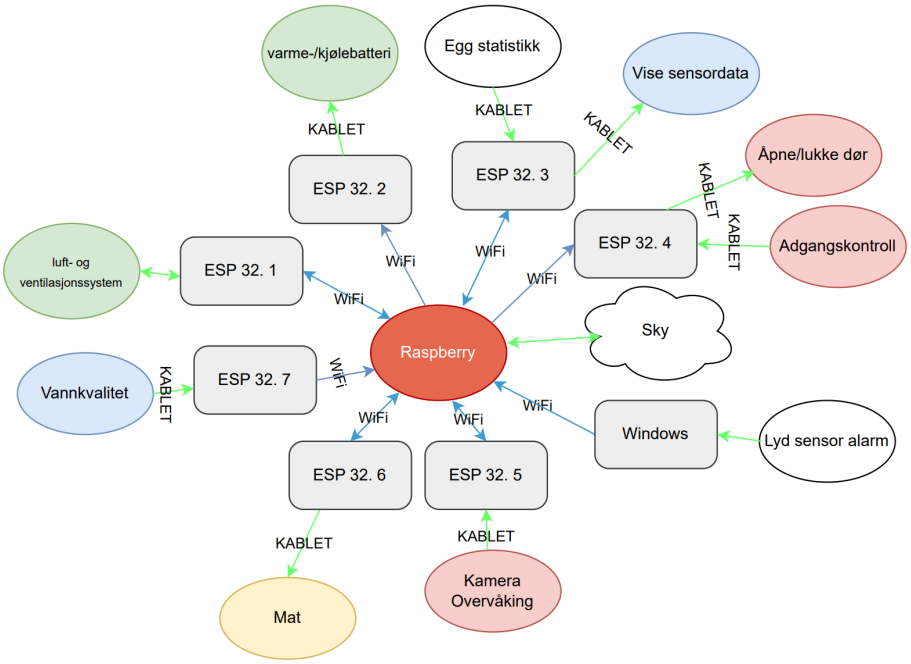
OpenJS (2023). *Node-RED*. URL: https://nodered.org/about/ (sjekket 12.12.2023).

SFUPTOWNMAKER (2023). *I2C*. URL: https://learn.sparkfun.com/tutorials/i2c/all (sjekket 10.12.2023).

The Geek Pub (2019). *How Rotary Encoders Work – Electronics Basics*. URL: https://www.thegeekpub.com/245407/how-rotary-encoders-work-electronics-basics/ (sjekket 10.12.2023).

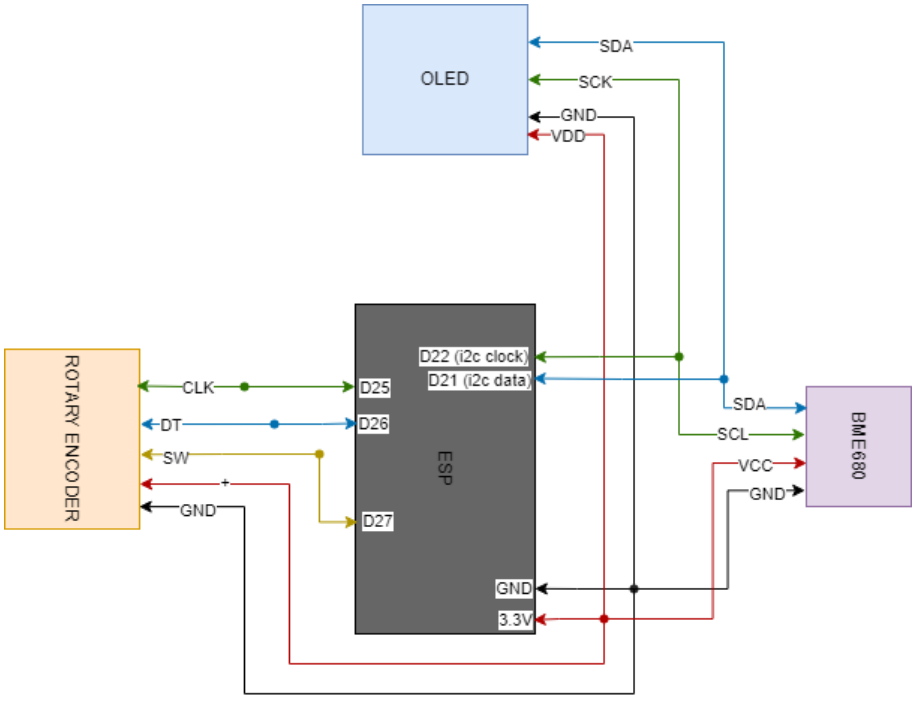
7 Vedlegg

7.1 Topologisk fremstilling av systemet



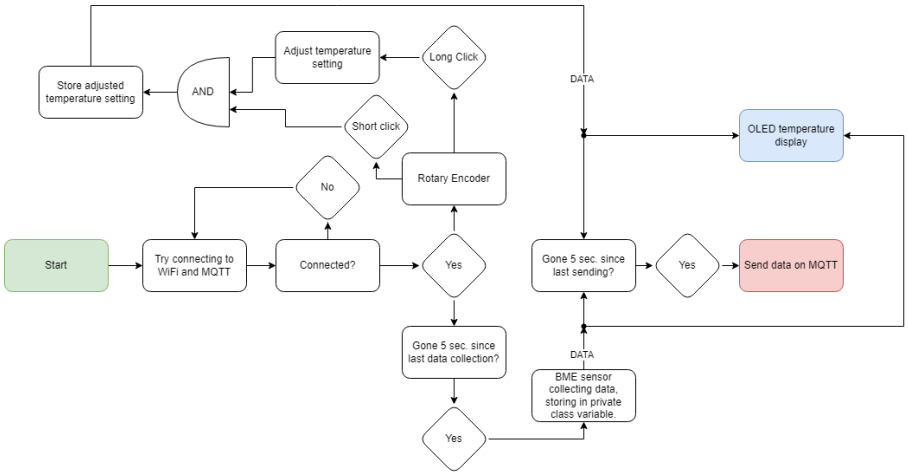
Topologisk fremstilling

7.2 Koblingsskjema for styreenheten til luft og ventilasjon



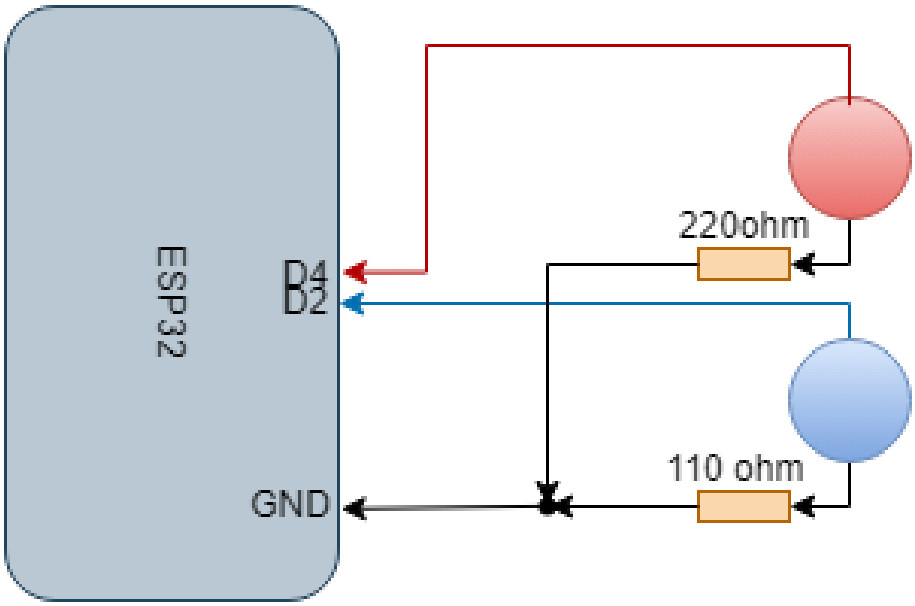
Koblingsskjema for styreenheten til luft og ventilasjon[ESP32. 1]

7.3 Flytskjema for programmet til styreenhet til luft og ventilasjon



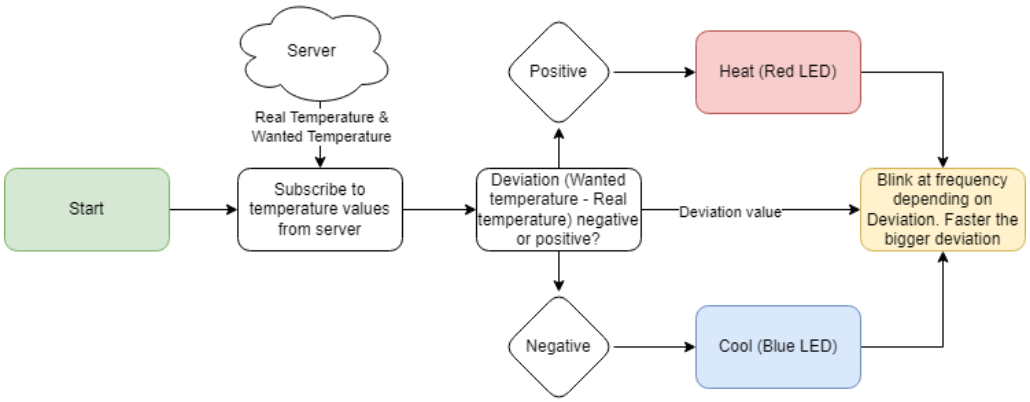
Flytskjema for programmet til styreenhet til luft og ventilasjon[ESP32. 1]

7.4 Koblingsskjema for kjøle/varme-batteri



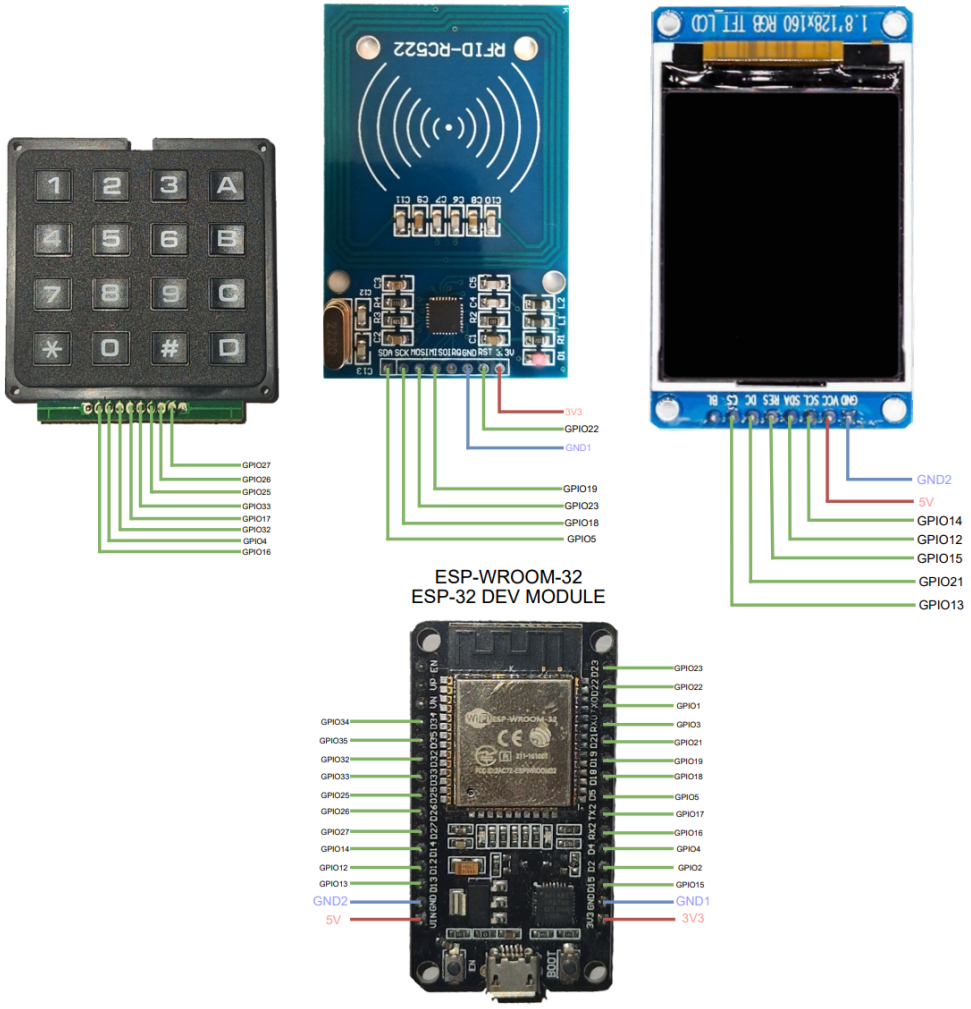
Koblingsskjema for kjøle/varme-batteri[ESP32. 2]

7.5 Flytskjema for programmet i kjøle/varme-batteri



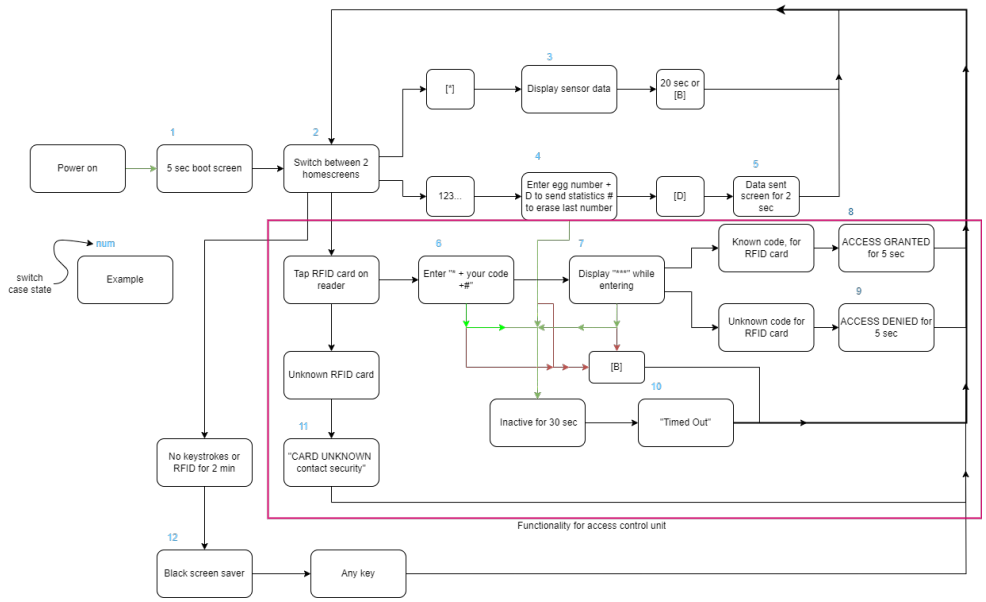
Flytskjema for programmet i kjøle/varme-batteri[ESP32. 2]

7.6 Koblingsskjema for terminalen og adgangskontrollenhet



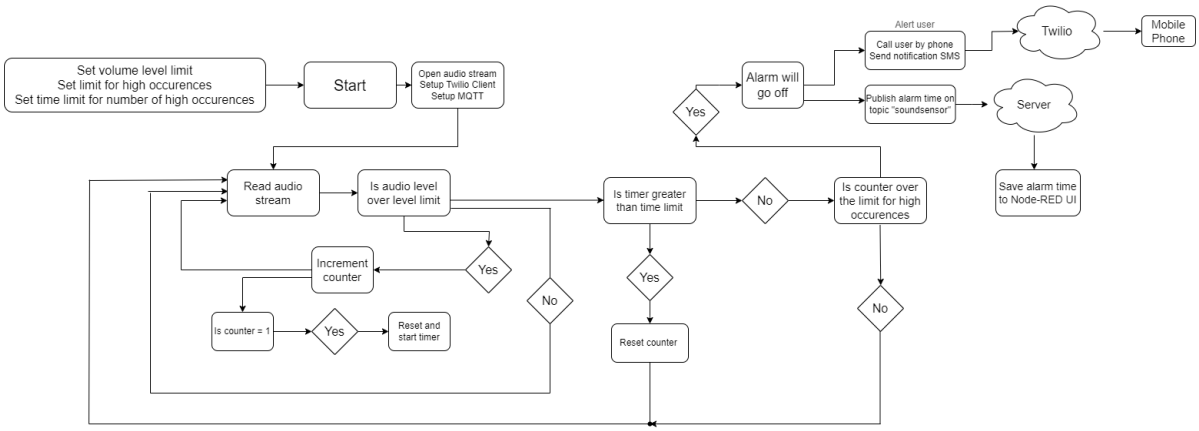
Koblingsskjema for terminalen og adgangskontrollenhet[ESP32. 3,4]

7.7 Flytskjema for programmet terminalen og adgangskontrollenhet



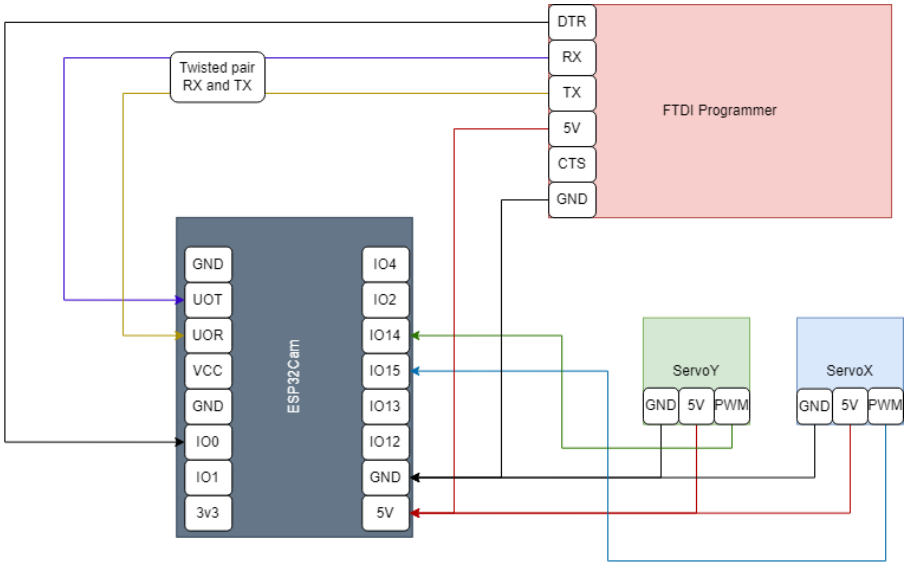
Flytskjema for programmet terminalen og adgangskontrollenhet[ESP32. 3,4]

7.8 Flytskjema for programmet til lydsensoralarm



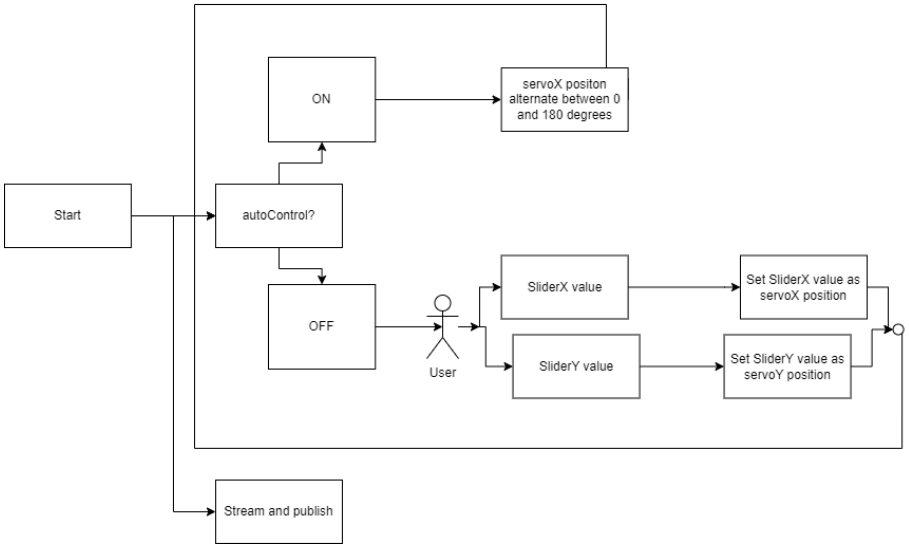
Flytskjema for programmet til lydsensoralarm

7.9 Oppkopling av ESP32Cam, FTDI Programmer og servomotorer



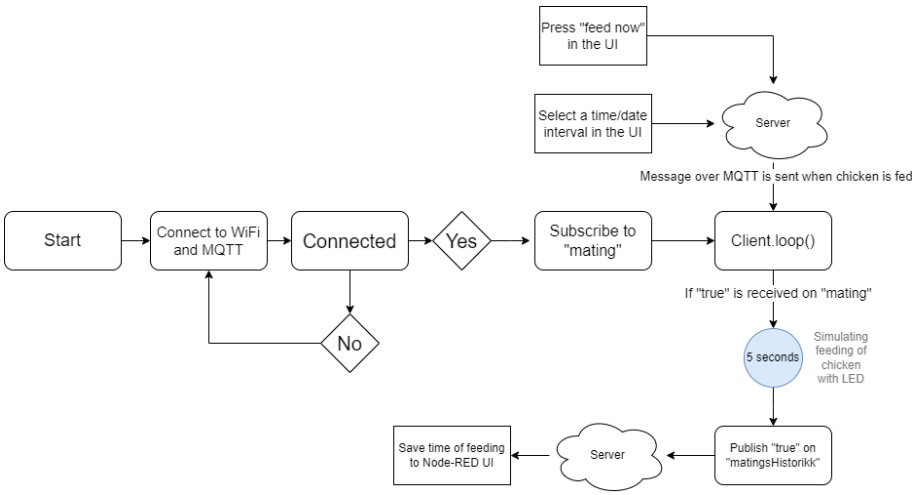
Oppkopling av ESP32Cam, FTDI Programmer og servomotorer[ESP32.5]

7.10 Flytskjema for ESP32CAM servoer



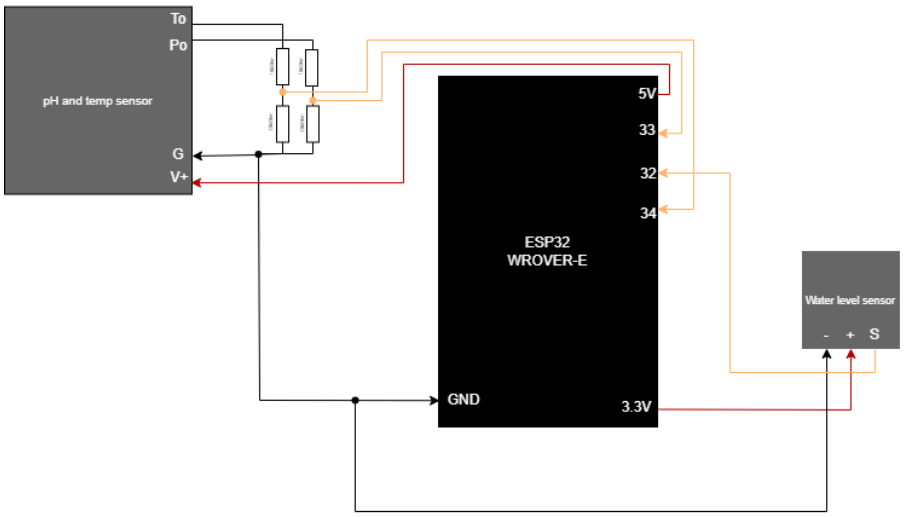
Flytskjema for kode av ESP32CAM servoer [ESP32.5]

7.11 Flytskjema for programmet til mating



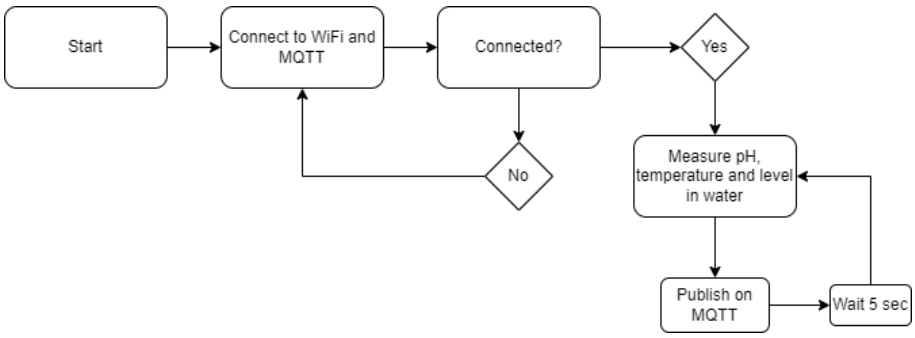
Flytskjema for programmet til mating[ESP32.6]

7.12 Oppkobling av vannkvalitetsmålinger



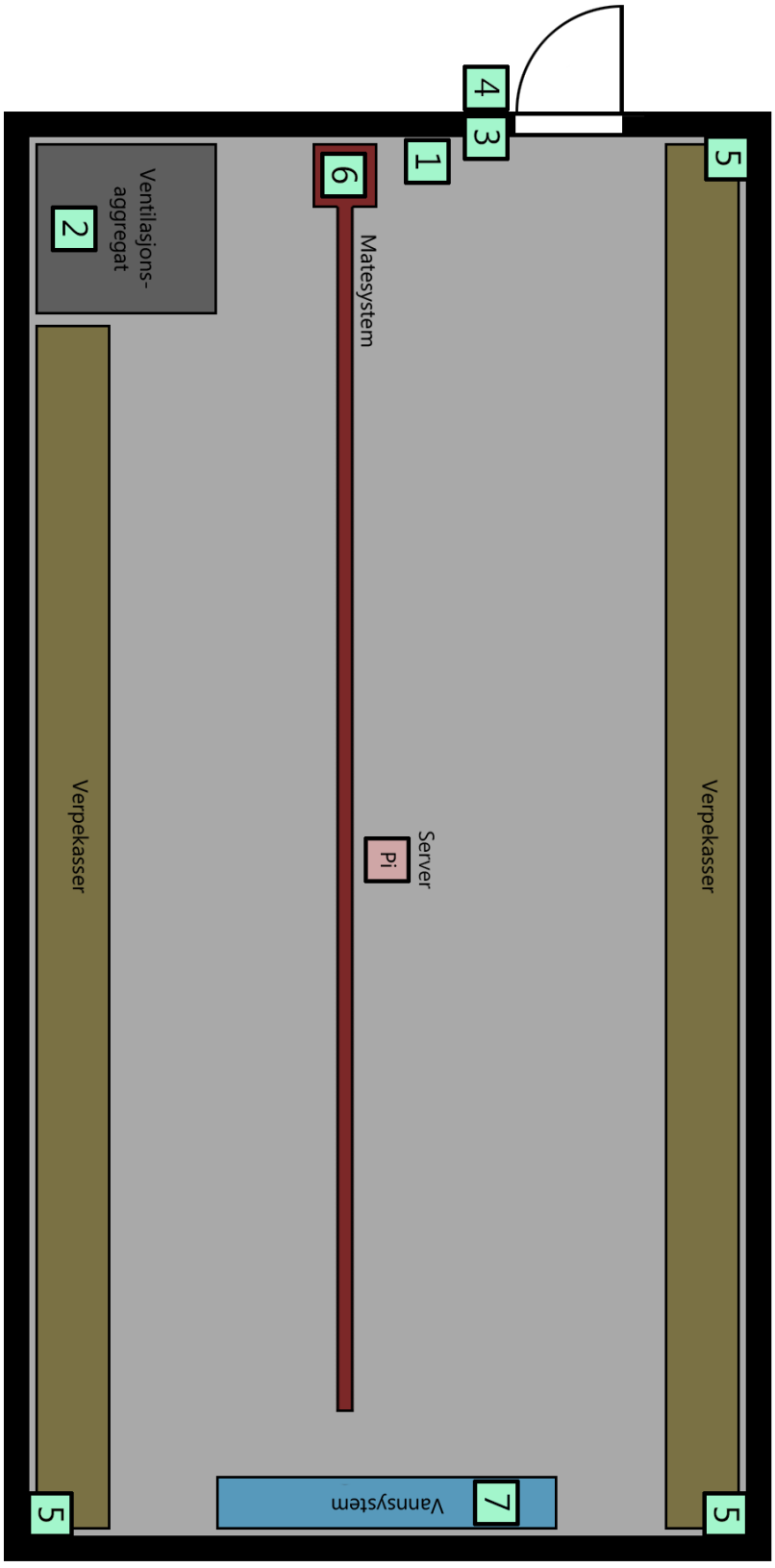
Kobling av vannkvalitetsmålinger[ESP32.7]

7.13 Flowskjema for kode av vannmålinger



Flytskjema for kode av vannmålinger[ESP32.7]

7.14 Eksempel på plassering av komponenter i et hønsehus



1: Styreenhet for luft of temperatur. **2**: Kjøle/varme-batteri. **3**: Terminal.
4: Adgangskontrollenhet. **5**: Kameraovervåking. **6**: Matingsenhet. **7**: Vannkvalitetsmåler

Kameraovervåking (5) kan plasseres én eller flere steder.

Nummereringa tilsvarer den topologiske oversikten