Høgskulen på Vestlandet

BACHELOROPPGAVE

Bikubeovervåking

Erling Tellnes

Erlend Matre

Jacob Eide

Bachelor i ingeniørfag–automatisering med robotikk

Institutt for datateknologi, elektroteknologi og realfag

Veileder: Yngve Thodesen

Innleveringsdato: 20. mai. 2025

Dokumentkontroll

Rapportens tittel: Bikubeovervåking	*Dato/Versjon* 20. mai. 2025/1.0
	Rapportnummer: BO25EB-01
Forfatter(e): Erling Tellnes Erlend Matre Jacob Eide	*Studieretning:* AUTB
	Antall sider m/vedlegg 63 + 54
Høgskolens veileder: Yngve Thodesen	*Gradering:* Åpen
Eventuelle Merknader: Vi tillater at oppgaven kan publiseres.	

Oppdragsgiver: Ingvar Bjelland	*Oppdragsgivers referanse:*
Oppdragsgivers kontaktperson(er) (inklusiv kontaktinformasjon): *Svein Haustveit - Svein.Haustveit@hvl.no*	

Revisjon	**Dato**	**Status**	**Utført av**
1.0	20.05.2025	Endelig versjon av rapport	Erling Tellnes, Erlend Matre, Jacob Eide

Forord

Denne rapporten er utarbeidet i forbindelse med vår avsluttende bacheloroppgave ved Høgskulen på Vestlandet, avdeling Bergen, våren 2025. Oppgaven har som mål å utvikle et system som kan integreres i en bikube, hvor systemet skal måle både vekten og temperaturen og vise måledata til brukeren.

Vi valgte denne oppgaven som vårt førsteprioriterte ønske, da vi oppfattet prosjektet som spennende og utfordrende. Oppgaven skilte seg ut ved å involvere utvikling av et fysisk produkt, noe som ga rom for stor kreativitet og selvstendig definering av løsningene. I tillegg har det vært motiverende at produktet var tenkt til å tas i bruk i praksis etter ferdigstillelse.

I arbeidet med denne oppgaven har vi brukt kunnskap og ferdigheter vi har tilegnet oss gjennom studietiden, spesielt innenfor fagområdene kretslære, mikrokontrollere og programmering. Underveis i prosjektet har vi også utviklet ny kompetanse, blant annet innen IT og IoT, samt i bruk av skybaserte databaser.

Vi ønsker å rette en spesiell takk til Ingvar Bjelland, som har definert oppgaven og sendt den inn til HVL som privatperson. Videre vil vi takke Høgskulen på Vestlandet for tilgang til nødvendige komponenter og verktøy, samt HVL Skape for bruk av 3D-printer. En stor takk rettes også til vår veileder, Yngve Thodesen, for verdifulle råd og konstruktive tilbakemeldinger gjennom hele prosjektperioden.

Sammendrag

Oppgaven vår går ut på å utvikle et innovativt system som automatisk overvåker bikuber. Systemet skal samle inn temperatur og vekt til en bikube, og vise dette til brukeren.

I første fase analyserte vi kravspesifikasjonen, utarbeidet en oversikt over hvilke funksjoner systemet måtte ha, og planla hvordan vi skulle løse oppgaven. Vi tok initiativ til å utvide og forbedre prosjektet, og i samråd med veileder og oppdragsgiver ble vi enige om at måledataene ikke skulle vises lokalt på en skjerm, men heller sendes over internett og vises for brukeren via en app eller nettside. Deretter utarbeidet vi en fremdriftsplan, delte prosjektet inn i milepæler, og estimerte nødvendig tid og ressursbruk for de ulike delene av arbeidet.

Resultatet ble et system som automatisk og periodisk innhenter vekt og temperaturdata fra bikuben, og gjør disse tilgjengelige for brukeren via en nettside. Systemet virker uten å være tilkoblet strømnettet eller WiFi, da det har egne løsninger for dette.

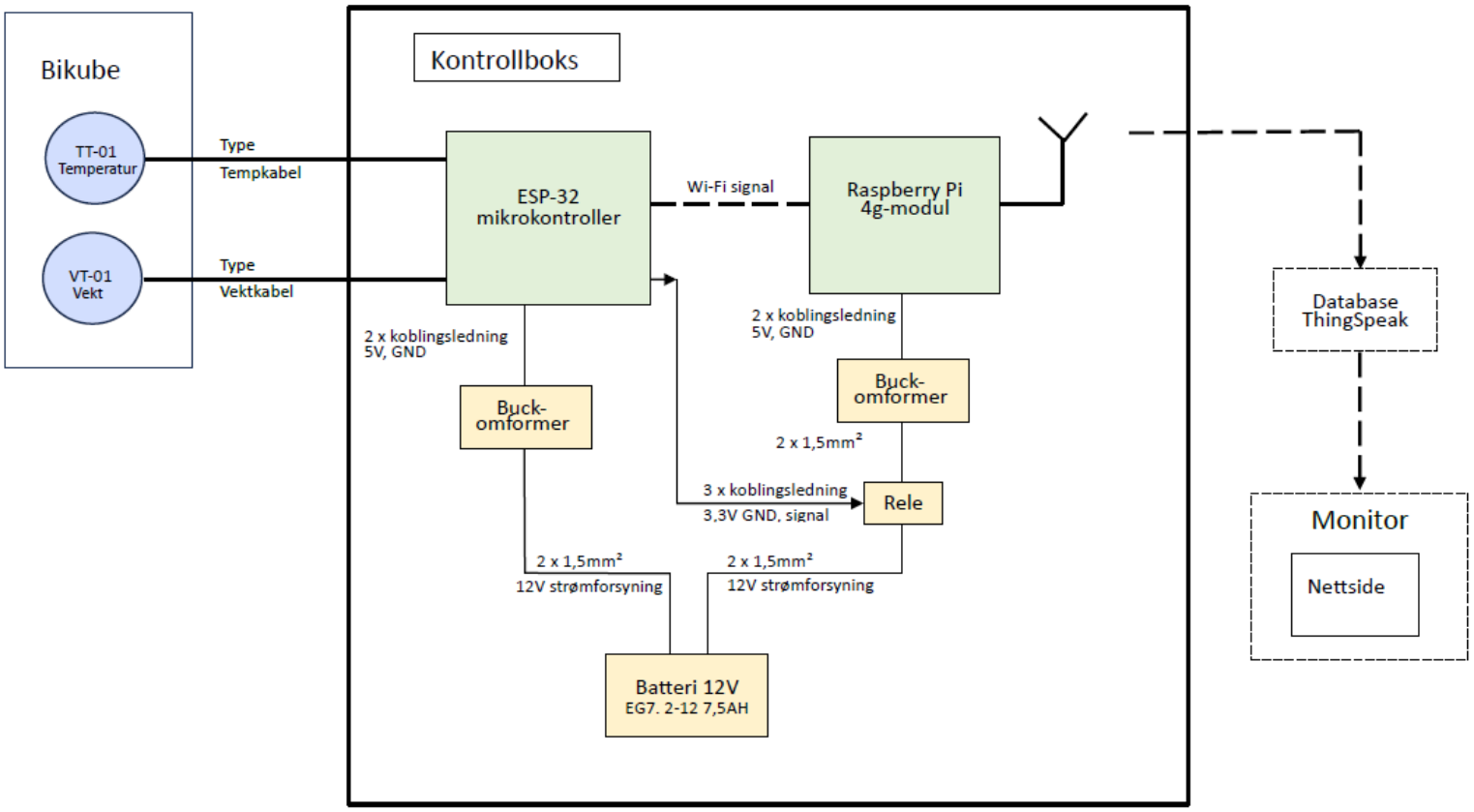
For å måle vekten av bikuben monterte vi fire vektceller under en plate som plasseres under selve kuben. For å måle temperaturen, monterte vi en temperaturføler inne i yngelrommet i bikuben.

Sensorene er koblet til en mikrokontroller, som er plassert i en kontrollboks i nærheten av kuben.

For å sende og vise data, kobler mikrokontrolleren seg til et WiFi-nettverk og overfører måledataene til en database. WiFi-nettverket settes opp av en Raspberry Pi, som er tilkoblet en 4G-modul. Denne gjør det mulig å opprette et lokalt WiFi-nett basert på mobilnettverk.

Måledataene som lagres i databasen hentes deretter ut og vises for brukeren via en nettside.

Prosjektet har resultert i en fungerende prototype som demonstrerer hvordan systemet er ment å fungere, men som har noen begrensninger. På grunn av systemets kompleksitet og behovet for å bestille og teste en rekke ulike komponenter med lang leveringstid, samt flere ganger måtte bytte ut deler og bestille på nytt, har vi ikke klart å ferdigstille et fullverdig produkt som svarer til alle kravene i oppgaven knyttet til innkapsling og utendørsbruk. Prototypen gir likevel et godt grunnlag for videre utvikling, og rapporten beskriver både prototypen vi har laget, og en mer ferdig utgave, i grundig detalj. Vi har blant annet utviklet og bestilt et eget kretskort for å erstatte de mer uoversiktlige koblingsledningene, noe som gir et mer kompakt og ryddig design. Dette har vi ikke fått i posten enda. Det er også utarbeidet en bruksanvisning som gjør det mulig for oppdragsgiver å bygge en egen versjon av systemet.



Figur 1: Blokkdiagram av systemets oppsett.

1 Innhold

Figurliste

1 Innledning

1.1 Oppdragsgiver

Ingvar Bjelland har vært vår oppgavegiver for dette bachelorprosjektet. Han driver med birøkt som hobby på fritiden og kom til HVL med denne oppgaven på eget initiativ. Han ga oss det opprinnelige scenariet og problemstillingen som prosjektet er bygget opp på. I løpet av prosjektet har han kommet med innspill i form av ideer og relevante spørsmål. Selv om han ikke har deltatt aktivt i utvikling av løsningen har hans bidrag vært viktig for å definere retninger vi har tatt og rammene for prosjektet.

1.2 Problemstilling

1.2.1 Bakgrunn

Honningbier er viktig for å opprettholde økosystemer rundt om i hele verden og de bidrar til pollinering av en stor andel av verdens matplanter, noe som gjør dem essensielle for både matproduksjon og biologisk mangfold. Birøkt er derfor veldig viktig, både for pollineringen og for å sikre produksjon av honning og andre biprodukter.

I tillegg til at det er kjekt å vite hvor mye honning som blir produsert er det viktig å overvåke bikubene. Ved å måle både vekt og temperatur i kuben kan birøkteren få innsikt i bienes tilstand og aktivitet. For eksempel kan et plutselig vekttap indikere at sverming er i gang. Sverming er en naturlig del av bienes formeringsprosess, der dronningen og en stor del av arbeiderbiene forlater kuben for å danne en ny koloni. Dette kan føre til redusert honningproduksjon og en svekket bikube. Ved tidlig varsling kan birøkteren gripe inn og forsøke å hente svermen til en annen kube. [1]

1.2.2 Oppgaven

Oppdragsgiver ønsker en løsning for overvåkning av temperatur og vekt av sine bikuber som står på Sotra like utenfor Bergen, et stykke unna der han selv bor. Løsningen som han bruker pr nå, er å manuelt flytte kubene over på en vekt, for så å lese av vekten. Han ønsker seg en smartere løsning.

I oppgaveteksten står det en kort beskrivelse av produktet som oppdragsgiven ønsker seg:

«En elektronisk vekt for bikuber som også måler temperaturen inni bikuben. Den skal kunne stå permanent under kuben (utendørs) og måle kontinuerlig. Den skal vise sanntid vekt og temperatur på et display (ev. via Bluetooth på en smartphone), men også lagre data som kan

vises som kurver (tid langs x-aksen, måleverdi langs y-aksen). Fordi det er dårlig mobildekning der kubene står, er det trolig ikke aktuelt med en wi-fi løsning.»

I samråd med oppdragsgiver og veileder besluttet vi å utvide oppgaven til å innebære sending av måledata til en database, og videre visning på en nettside, og ikke på et display lokalt på bikuben. Fordelene med denne løsningen er at oppdragsgiver kan se på måledata uten å fysisk reise ut til bikuben.

I tillegg ønsker oppdragsgiver en bruksanvisning som viser oppbygningen til systemet, slik at han kan lage en versjon til hvis han ønsker.

1.2.3 Utfordringer

Utfordringene med den utvidede løsningen er at systemet blir mye mer avansert og komplekst. Det er flere utfordringer knyttet til både sending av data og økt strømforbruk.

Siden bikuben er plassert i et område uten tilkobling til strømnettet, er vi avhengige av en batteridrevet løsning. Det er derfor avgjørende å finne en strømeffektiv løsning som er godt egnet for formålet.

I tillegg er mobildekningen i området dårlig, og det er ikke tilgang til WiFi. Dette har gjort det nødvendig å benytte en alternativ metode for å overføre data til internett. Også her må løsningen være energieffektiv, ettersom systemet drives av batteri.

Temperatur og vektsensorene som benyttes må levere presise målinger, samtidig som de må være robust nok til å tåle varierende værforhold og kontinuerlig utendørs bruk.

Bruksanvisningen skal helst være mulig for en som ikke er ekspert på elektronikk og programmering å følge, men kompleksiteten på systemet kan gjøre dette utfordrende.

2 Kravspesifikasjon

2.1 Funksjonelle krav

2.1.1 Måling av vekt og temperatur

For å holde oversikt over tilstanden i bikuben, ønsker oppdragsgiver måling av både vekt og temperatur med nøyaktighet på +/- 100g og +/- 1 ˚C. Oppdragsgiver har også spesifisert at vi kan se vekk ifra ekstra vekt i forbindelse med eventuell snø som lander på bikuben. Systemet skal fungere uavhengig av strømtilgang og være batteridrevet. For å unngå hyppig oppladning, er det ønskelig at systemet kun sender data en gang per dag. Dette er tilstrekkelig for at oppdragsgiver raskt kan reagere dersom det oppstår noe uventet.
Som del av dette arbeidet inngår følgende krav:

- Skisse av den elektroniske løsningen med nødvendige sensorer og mikrokontroller.
- Valg av konkrete komponenter (med vurdering av pris og kvalitet).
- Testing av løsningen for å verifisere at målingene er stabile og nøyaktige over tid.

2.1.2 Visning av data

Opprinnelig ønsket oppdragsgiver et fysisk display, men dette ble endret etter dialog. Det ble besluttet at en nettside ville gi bedre tilgjengelighet og brukervennlighet. Løsningen krever ingen vedlikehold fra brukerens side og gir tilgang fra mobil, PC eller nettbrett.
Dataene presenteres som grafer, som gir et tydelig bilde av utvikling over tid. Dette gir rask innsikt i bikubens tilstand og reduserer behovet for fysisk inspeksjon.

2.2 Tekniske og praktiske krav

2.2.1 Robusthet og værbestandighet

Systemet skal monteres utendørs og må tåle norske værforhold, inkludert regn, snø, vind og temperatursvingninger. Løsningen skal derfor bygges med vanntett kapsling og robuste materialer som beskytter elektronikken mot ytre påvirkning.

2.2.2 Energieffektivitet

Ettersom det ikke finnes strømtilgang på stedet der bikuben står, er systemet avhengig av batteri. Etter dialog med oppdragsgiver ble vi enige om at driftstiden på batteriet måtte vare i minst to uker. Derfor er det avgjørende at løsningen er energieffektiv og bruker minst mulig strøm. Det innebærer at systemet må være aktivt bare når det skal samle inn og sende data,

og resten av tiden være i en strømsparende tilstand. Målet er at batteriet skal vare i flere uker, helst en måned eller mer før det må lades eller byttes. Dette reduserer behovet for vedlikehold og gjør løsningen mer praktisk for oppdragsgiver.

2.2.3 Kostnad

Oppdragsgiver stiller krav om en rimelig løsning, derfor skal alle komponenter velges med hensyn til lavest mulig pris uten å gå på kompromiss med funksjon og kvalitet.
Kostnadene dokumenteres i en oversikt som sammenligner prosjektkostnaden med kommersielle alternativer.

2.2.4 Brukervennlighet og enkel montering

Løsningen skal kunne monteres og tas i bruk uten å være ekspert på elektronikk og programmering. Brukeren skal kunne følge en bruksanvisning med bilder og forklaringer for oppsett. Det legges også vekt på brukervennlig programvare, nettsiden for visning av data skal være intuitiv og enkel. Målet er at oppdragsgiver selv skal kunne montere en ekstra enhet hvis ønskelig, basert på medfølgende dokumentasjon.

2.3 Endringer og vurderinger

Underveis i prosjektet har vi hatt dialog med oppdragsgiver. Blant annet ble ønsket om fysisk display justert til en nettbasert løsning etter vurdering av behov og praktiske hensyn. Det har vært viktig å tilpasse løsningen etter oppdragsgivers faktiske brukssituasjon.

2.4 Mål for prosjektet

Prosjektets mål er å utvikle et robust, kostnadseffektivt og energieffektivt overvåkningssystem som måler temperatur og vekt i en bikube og overfører data til en nettside en gang per dag. Løsningen skal være enkel å montere og bruke, og fungere uten strømtilkobling.
Ferdigstillelse inkluderer:

- Komplett skisse og plan for både elektronisk og fysisk utforming.
- Dokumentert valg av komponenter og anskaffelsesmetode.
- Fullstendig teknisk dokumentasjon som gjør det mulig for oppdragsgiver å reprodusere løsningen ved behov.
- En fungerende prototype.

3 Analyse av problemet

3.1 Problemdefinisjon

Målet med systemet er å overvåke en bikube ved hjelp av sensorer for temperatur og vekt, med dataoverføring en gang om dagen via 4G uten tilgang til fast strømnett. Utfordringene ligger i å balansere strømforbruk, miljørobusthet og pålitelig kommunikasjon i et område med begrenset 4G-dekning og tøft værforhold. Systemet må kunne motstå temperaturer mellom -10 grader og 30 grader, høy luftfuktighet og salt i luften fra sjøen, samtidig som det opprettholder nøyaktighet på pluss minus 0,5 grader for temperatur og pluss minus 100 gram for vekt. Dette krever en solid dekomponering av systemet i underliggende komponenter, der hver del må møte spesifikke krav.

3.1.1 Dekomponering av systemet

Systemet er delt inn i fire hovedmoduler: Strømforsyning med spenningsregulatorer, sensorer, kommunikasjon og design. Strømforsyningen må sikre kontinuerlig drift over lang tid, mens sensorene må gi pålitelige data under tøffe værforhold. Kommunikasjonsmodulen må håndtere varierende 4G-dekning, og det fysiske designet må beskytte elektronikken mot fuktighet og korrosjon. Hver modul er tilpasset oppdragsgiverens krav om lavere kostnader enn allerede eksisterende løsninger, enkelt vedlikehold og mulighet for senere utvidelse til flere bikuber.

3.1.2 Strømforsyning

Ettersom systemet skal operere uten tilkobling til strømnettet og stå ute i lengre perioder uten vedlikehold, er det nødvendig med en løsning som sikrer lang driftstid og stabil spenningstilførsel. Det må derfor benyttes en batteripakke med tilstrekkelig kapasitet, og det er avgjørende at systemet er designet med fokus på lavt strømforbruk. Enkelte komponenter, som kommunikasjonsmoduler og mikrokontrollere, bør støtte dvalemodus og kunne aktiveres kun ved behov. I tillegg må strømforsyningen kunne håndtere variasjon i effektbehov mellom inaktive og aktive perioder. Systemet bør også vurderes for fremtidig bruk av lading via solceller, for å forlenge levetiden ytterligere.

3.1.3 Sensorer og Datainnsamling

Systemet skal kontinuerlig samle inn temperatur- og vektdata med god nøyaktighet. Dette krever sensorer som både er presise og robuste. For temperaturmåling er det nødvendig

med sensorer som fungerer pålitelig i utendørs miljø, med fuktighet og temperatursvingninger. Vektmåling krever god mekanisk stabilitet og mulighet for kalibrering. Sensorene må kunne samhandle med mikrokontrolleren uten behov for høy prosessorkraft, og data må kunne samles inn og behandles raskt slik at energiforbruket reduseres.

3.1.4 Kommunikasjon

Datainnsamlingen har liten verdi uten pålitelig overføring av informasjon til en ekstern server. Systemet må derfor støtte trådløs kommunikasjon som sikrer stabil og effektiv dataoverføring. Dette stiller krav til både maskinvare og programvare. Kommunikasjonsmodulen må enten være alltid tilgjengelig eller kunne startes opp automatisk ved behov, noe som krever styring av strøm og oppkobling via mikrokontroller. Systemet må kunne håndtere ustabile forbindelser og gjenopprette tilkobling automatisk ved oppstart.

3.1.5 Mekanisk design

Systemet skal stå utendørs over lengre tid og må derfor ha et fysisk design som tåler vær og vind, samtidig som det beskytter de elektroniske komponentene mot vann, støv og mekaniske belastninger. Det er behov for en innkapsling som tåler å stå ute og god intern organisering slik at sensorer, strømforsyning og kontrollenheter er strategisk plassert. Temperaturmåleren må plasseres på en måte som reflekterer reel temperatur inne i bikuben, uten av den påvirkes av honning eller andre ting som kan forekomme inne i en bikube. Vekten må kunne monteres på et stabilt underlag og beskyttes mot nedbør, skadedyr og ytre krefter. Hele systemet må være lett å montere og demontere for vedlikehold og justering.

3.2 Valg av komponenter og kostnadsanalyse

Valget av komponenter er basert på en balanse mellom kostnad, strømforbruk og pålitelighet.

Komponent	**Ca. Pris (NOK)**
Raspberry Pi 4B (8gb eller 4gb)	1300kr 850kr
Waveshare SIM7600E-4G-HAT	1000kr
Mikrokontroller	70-100kr
Temperatursensor	50-100kr
Vektceller	50-100kr
Batteri	200-300kr
Relé	100kr
Spenningsregulator	30kr-100kr
Kretskort	100kr
SIM-kort	100kr mnd.
Totalt	**2580kr-3300kr**

3.2.1 Eksisterende løsninger og markedsanalyse

Det finnes i dag en rekke kommersielt tilgjengelige løsninger for overvåkning av bikuber. Disse systemene tilbyr stort sett de samme grunnleggende funksjonene, som overvåkning av temperatur, vekt, fuktighet og lydnivå inne i kuben. Systemene varierer imidlertid i hvordan de er konstruert og hvordan data hentes ut. Noen løsninger er helt batteridrevne og trådløse, mens andre er avhengige av ekstern strømforsyning. Enkelte systemer krever fysisk tilstedeværelse for å lese av data, da de ikke er koblet til internett.

Nedenfor gjennomgås to sentrale kommersielle aktører på markedet:

BroodMinder er en amerikansk anerkjent produsent som tilbyr en rekke modulbaserte sensorer for birøktere. Systemene deres er fleksible og kan benyttes enten separat eller sammenkoblet, avhengig av behov. De vanligste modulene inkluderer:

- Vektsensorer for å overvåke honningproduksjon og kubens generelle helse
- Temperatur- og fuktighetssensorer
- Akustiske sensorer som måler lydnivået og flyaktiviteten inn og ut av kuben

Alle BroodMinder-moduler er batteridrevne, med fokus på lavt strømforbruk og lang batterilevetid. For fjernovervåkning tilbyr de en separat kommunikasjonsmodul kalt «Cell Hub», som bruker mobilnett (4G) til å sende data til en skyløsning. Dette gir mulighet for sanntidsdata og historisk analyse via mobil eller web. Prisene varierer mellom 200 og 500 USD per modul, avhengig av funksjonalitet og tilbehør. [2]

Beehivemonitoring (Norge)

Beehivemonitoring er et norsk firma som spesialiserer seg på brukervennlige overvåkningssystemer for småskala birøktere. Selskapet tilbyr både frittstående komponenter og komplette pakkeløsninger med tilhørende mobil- og webapplikasjoner.

Hovedproduktet deres er Kubehjerte, en batteridrevet mikrokontroller med integrerte sensorer for temperatur, fuktighet og lyd. Kubehjerte kan også utvides med eksterne moduler, som vektsensor og 4G-kommunikasjonsenhet. Dette gjør det mulig å fjernovervåke bikuben i sanntid, noe som er særlig nyttig i områder uten fast strømtilkobling eller WiFi.

- Pris: Komplett pakke med Kubehjerte, vektsensor og 4G-modul koster ca. 6300 kr. Dette inkluderer ett års abonnement på datahåndtering og tilgang til deres digitale plattform.
- Abonnement: Etter første år koster abonnementet 320 kr per år. [3]

3.3 Arbeidsflyt

3.3.1 Prototype

Prototypen må være et kompakt og strømsparende system som integrerer flere teknologier for å samle inn og sende data fra sensorer. Den vil bestå av en mikrokontroller, et batteri, en temperatursensor og en vektmåler. Prototypen vil også bruke en Raspberry Pi 4B med en Waveshare 4G-HAT-modul som ruter for å opprette et WiFi-hotspot.

I designet er mikrokontrolleren programmert til å våkne fra dvale på bestemte tidspunkter for å aktivere et relé som gir strøm til ruteren. Denne setter opp et WiFi-hotspot som mikrokontrolleren kan koble seg til for å kunne sende sensor data til ThingSpeak. Når dataoverføringen er fullført, slår mikrokontrolleren av reléet for å kutte strømmen til ruteren og deretter gå tilbake i dvale.

Dette gjør at systemet kan operere i lange perioder med lavt strømforbruk. Komponentene er nøye valgt og konfigurert for å minimere energibruk, samtidig som de sikrer regelmessig og effektiv dataoverføring.

I tillegg til strømsparingen, er løsningen fleksibel nok til å støtte både WiFi og 4G-tilkobling, og kan tilpasses ulike bruksområder som krever trådløs dataoverføring fra flere sensorer.

3.4 Risikoanalyse og forbedringspotensial

3.4.1 Risiko

Utviklingen av et batteridrevet overvåkningssystem for bikuber innebærer flere mulige risikoer som må vurderes tidlig i prosessen. Den største utfordringen er pålitelig drift over lange perioder uten fysisk tilsyn, noe som stiller høye krav til både maskinvare og programvare. Energiforbruk må holdes på et minimum for å sikre lengst mulig batterilevetid, samtidig som systemet må kunne samle inn og overføre data regelmessig.

Kommunikasjon utgjør en kritisk risikofaktor. Dersom 4G-dekningen er veldig svak, vil systemet bruke lengre tid på å gjennomføre overføringen. Dette fører til økt strømforbruk siden systemet forblir aktivt over lengre tid.

Det finnes også en risiko for fuktinntrengning og kondens, spesielt ved utendørsplassering over tid. Dette kan føre til feil i elektriske komponenter. I tillegg må det tas høyde for mulige programfeil, ustabil sensorlesning og generell slitasje.

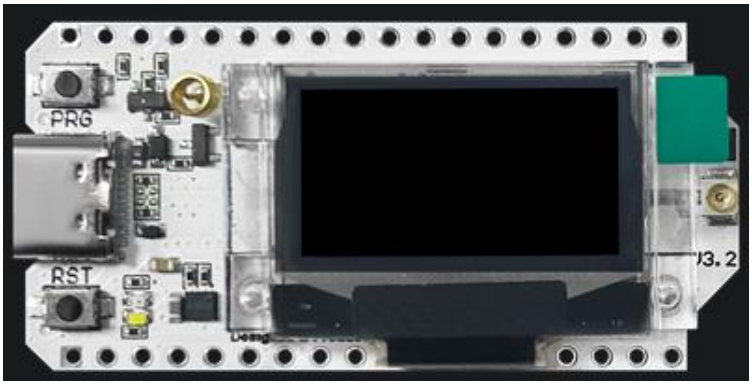
3.4.2 Forbedringspotensial

For å forbedre produktet ytterligere, så kan man investere i et solcellepanel, bruke edge computing som f.eks. lagrer data lokalt for hver time i løpet av hele dagen, og sender gjennomsnittet av dette en gang om dagen. Bruke loRaWan som backup visst 4G svikter. Solcellepanel for å unngå å bytte/lade batteri.

3.5 Utforming av mulige løsninger på sending/lagring av data

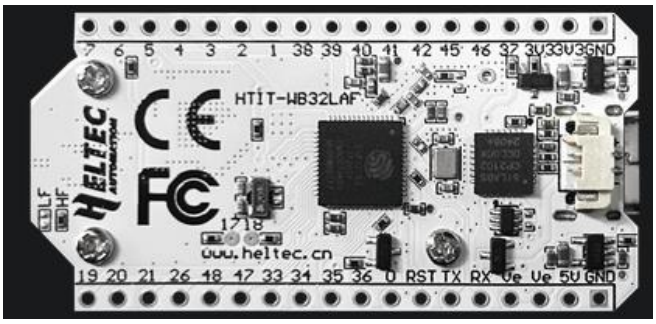
3.5.1 Løsningsalternativ 1: LoRaWAN-system med ESP32

I denne løsningen så vil et LoRaWAN-modem sende data til en LoRaWAN-gateway som «the things network». En gateway fungerer som et mellomledd mellom sensorenhetene og internett, den mottar data fra LoRa-enheter og videresender det til en nettbasert server. Dette kretskortet har ekstremt lavt strømforbruk og kan sende data over større distanser. Denne metoden har begrenset med datakapasitet på hver melding (51 bytes) og er avhengig av en gateway som er innenfor rekkevidde. Det er mulig å bruke andre sine gateways, men visst eieren tar av gateway'en, så vil det ikke virke lengre. Eventuelt så må en sette opp en egen gateway som man har hjemme med konstant strøm. Rekkevidden på enheten kan fort reduseres ved skog og fjell. Noe som gjør at en må ha en gateway som er koblet til strøm innenfor en kortere radius. De fleste gateway'ene som er i vårt område er opptil 30 km unna bikubens plassering og det er mye skog og fjell, noe som gjør at denne løsningen ikke er optimal der bikuben er satt opp.



[4]

Figur 2: LoRaWAN



[4]

Figur 3: LoRaWAN

3.5.2 Løsningsalternativ 2: SMS-basert system med en GSM-modem

Dette systemet er bygget på å bruke en Raspberry Pi Zero med et gsm modem som vil sende data som en SMS (Short Message Service, 160 tegn). Dette systemet vil kunne fungere selv med svakt 2G-signal (svak mobildekning). Med dette systemet så vil en kunne få SMS sendt til sin egen telefon via et kontantkort som står i gsm-modemet. Her vil kostnad på hver SMS ligge på rundt 1kr og det vil ikke være mulig å få noe annet enn tekstdata. Dette systemet ser vi på som et midlertidig alternativ fordi 2G-signalet vil legges ned etter 2027.

3.5.3 Løsningsalternativ 3: Bluetooth med SD-kort

Her vil en mikrokontroller bli brukt til å samle data fra vektcelle og temperatursensor, og lagres på et minnekort som brukeren kan hente data manuelt via bluetooth. Her vil det ikke være noe sending av data, så en må fysisk være i nærheten av bikuben man vil hente data fra. Vi ser mer på dette som en mulig backup løsning enn en hovedløsning med tanke på område bikuben er plassert på.

3.5.4 Vurderinger av komponenter

Underveis i prosjektet ble flere ulike løsninger vurdert for å sikre en stabil og energieffektiv måte å samle inn og sende data fra en avsidesliggende bikube. Det ble tidlig klart at systemet måtte kunne operere uten tilgang til vanlig strøm eller WiFi, og at det skulle bruke minst mulig energi. Med dette som utgangspunkt startet vi utprøving av forskjellige komponenter og kommunikasjonsmetoder.

Et av de første alternativene vi vurderte var å bruke en LilyGO T-Call ESP32 med A7670E-modul. Denne enheten kombinerer mikrokontroller og 4G-modem i en kompakt enhet, og virket derfor som en lovende løsning, særlig med tanke på lavt strømforbruk og enkel maskinvare.

Planen var å bruke den på samme måte som den løsningen vi senere realiserte med Raspberry Pi og 4G-HAT: altså at LilyGO skulle sette opp et hotspot som en mikrokontroller kunne koble seg til for å sende data videre til thingspeak. Vi fikk opp hotspotet på LilyGO og klarte å koble til mikrokontrolleren til nettverket. Likevel støtte vi på problemer med kommunikasjonen videre, selv om tilkoblingen ble etablert fikk ikke mikrokontrolleren sendt data over nettverket. Det virket som det var problemer med IP-adressering eller ruting på nettverket LilyGO satte opp, og vi fant ikke ut hvordan vi kunne utbedre dette. Etter flere forsøk og feilsøking vurderte vi at løsningen ble for ustabil til å kunne benyttes.

Figur 4: LilyGO T-Call A7670E

I tillegg testet vi LoRaWAN som alternativ kommunikasjonsmetode. Dette systemet har ekstremt lavt strømforbruk og egner seg godt til IoT-løsninger over lange avstander. Dessverre var rekkevidden fra bikuben til nærmeste tilgjengelige LoRaWAN-gateway for lang til at det var pålitelig, og det var ikke mulig å sette opp en egen gateway nærmere på grunn av praktiske begrensninger. Dermed måtte vi også forkaste denne løsningen.

Selv om Raspberry Pi har et høyere strømforbruk enn ideelt, er den kun aktiv i korte perioder, noe som holder totalforbruket på et akseptabelt nivå. Til tross for at løsningen med LilyGO og LoRaWAN teoretisk sett kunne vært mer kompakt eller energieffektiv, viste Raspberry Pi-løsningen seg å være mer stabil og gjennomførbar innenfor prosjektets rammer.

3.6 Konklusjon av analyse

Etter å ha vurdert flere løsningsalternativer, ble løsningen med Raspberry Pi 4B og Waveshare 4G-HAT som ruter og mikrokontrolleren Wemos D1 mini valgt som den mest hensiktsmessige. Denne kombinasjonen oppfyller kravene til nøyaktighet i målinger, robusthet i felt og kostnadseffektivitet. Til tross for at strømforbruket til ruteren da er relativt høyt, oppnås god energieffektivitet ved at systemet kun aktiveres en gang daglig for datainnsamling og overføring.

Løsningen gir også en fleksibilitet og stabilitet, og bygger på komponenter med god dokumentasjon og støtte i utviklermiljøer, noe som gjør det mulig å finne liknende eksempler som en kan bruke til veiledning.

Valgt løsning gir dermed en god balanse mellom pålitelighet, funksjonalitet og fremtidige mulige utvidelser. Forbedringer som solcellelading og lokal datalagring kan i omgang ytterlige forbedre systemets bærekraft og autonomi, og løsningen har potensial for bred bruk innen moderne, datadrevet birøkt.

4 Realisering av valgt løsning

Under planlegging og utførelse av oppgaven har vi valgt å rette oss inn mot en modulbasert løsning. Dette ble gjort for å forenkle prosessen underveis og gi oss lettere tilgang til å bytte ut ulike deler og komponenter. Ved å dele systemet opp i moduler for strømtilførsel, datainnsamling, dataoverføring og visuell fremstilling, kunne vi jobbe mer effektivt og målrettet.

4.1 Strømtilførsel

På grunn av manglende strømnett på lokasjonen systemet skal brukes ble vi enige om å bruke en batteridrevet løsning. Vi valgte et 12V, 7.2Ah, oppladbart blybatteri fra Clas Ohlson. Dette gir en god balanse mellom kapasitet, størrelse og pris. Løsningen gjør systemet fleksibelt til plassering og kan brukes hvor som helst uten tilgang til strømnettet.

4.1.1 Batteri

For å optimalisere strømforbruket settes mikrokontrolleren i deepSleep mellom hver syklus. Denne styrer også strømtilførselen til ruteren via et relé, slik at hele systemet går i dvale når det ikke brukes.

Under testing fant vi ut at systemet bruker mellom 83 og 106 sekunder på å våkne fra deepSleep, innhente måledata, koble seg på nett og sende data til databasen. (Les mer i 7.2 Testing av strømforbruk her.)

Vi har gjort et estimat av strømforbruket til systemet for å se hvor lenge batteriet holder før det må lades. Her har vi lagt til grunn at systemet utfører én sending av måledata per dag, der én sending tar i snitt 87.1 sekunder.

Her er en oversikt av strømforbruk når vi måler mellom batteri og resten av systemet.

Tilstand	**Strømforbruk**	**Aktiv tid**	**Energiforbruk**
deepSleep	88mA	23t 59m 32.9s	8.8 mA * 23.9768 t ≈ 211 mAh
Aktiv måling og sending	350mA	87.1 sekunder	350 mA * 0.0242 t ≈ 8.5 mAh
			Total: 219.5 mAh /dag

Tabell 1

Daglig estimert forbruk er 219mAh. Og med et batteri på 7200mAh vil teoretisk driftstid på systemet være 32.79 dager

7200mAh / 219.5mAh ≈ 32.79 dager før batteriet må lades/ byttes.

4.1.2 Spenningsregulator

Systemet består av komponenter som trenger stabil spenning. Ved å koble både mikrokontroller og ruteren på samme spenningsregulator, vil vi risikere å få et større spenningsfall under måling og sending av data, som kan resultere i at en eller begge enhetene slår seg av. For å unngå dette har vi valgt å benytte to separate spenningsregulatorer.

Begge regulatorene er av typen LM2596S, en justerbar spenningsregulator som kan regulere spenning fra 3–40V ned til 1.25–35V, med en oppgitt maksimal belastning på 3A og typisk effektivitet opp til 75–85 % avhengig av belastning. [5]

Til styring og sensordelen benyttes en mikrokontroller, som krever 5V. LM2596S er i dette tilfellet valgt på grunn av lav kostnad, enkel tilgjengelighet og tilstrekkelig ytelse.

Ruteren krever 5V og kan trekker relativt høy strøm under aktiv bruk, spesielt ved dataoverføring. I vårt tilfelle er imidlertid ruteren kun aktiv i omtrent 87.1 sekunder per dag, og er ellers avslått. På grunn av den korte driftstiden vurderes LM2596S som en godt egnet spenningsregulator også for denne delen av systemet.

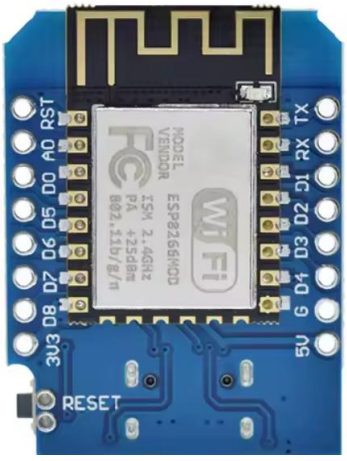
Den korte driftstiden på begge spenningsregulatorene gjør at det ikke er behov for aktiv kjøling, da varmeutviklingen blir minimal.

4.1.3 Relé

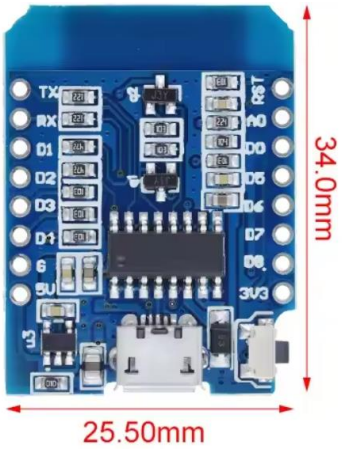
For å spare strøm ønsker vi at ruteren skal være avslått mellom hver måling. Siden Raspberry Pi ikke har innebygd dvalefunksjon, løser vi dette ved å kutte og gjenopprette strømtilførselen ved hjelp av et relé. Vi bruker en Luxorparts 1-kanals relémodul for Arduino, som fungerer utmerket til vår mikrokontroller. Når mikrokontrolleren våkner fra dvale, sender den et signal til reléet, som deretter kobler ruteren sammen med batteriet. [6]

4.2 Wemos D1 Mini (ESP8266)

Hjernen i systemets prosesskontroll er en mikrokontroller kalt Wemos D1 Mini, som er basert på ESP8266-modulen fra Espressif Systems. [7] Dette er en svært kompakt og kostnadseffektiv mikrokontroller som kombinerer digital behandling med trådløs kommunikasjon via Wi-Fi. Den måler kun ca. 34 x 25 mm, noe som gjør den ideell for prosjekter som vårt med begrenset plass. [8]



[9]



[9]

Figur 5: Frontside og bakside av Wemos D1 Mini

Wemos D1 Mini har en 32-bit RISC-prosessor som kjører på opptil 80 eller 160 MHz, og den har 4 MB flashminne for lagring av programkode og data. Den støtter både digital I/O og PWM-signaler, og har flere GPIO-pinner som enkelt kan kobles til sensorer og aktuatorer. Dette gir stor fleksibilitet i utformingen av både enkle og komplekse systemer.

En av de største fordelene med Wemos D1 Mini er dens innebygde Wi-Fi-modul, som gjør det mulig å koble seg direkte til internett eller et lokalt nettverk uten behov for ekstra maskinvare. Dette gjør den ideell for fjernovervåking av bikuben.

Wemos D1 Mini er kompatibel med Arduino IDE, noe som gjør utvikling, testing og feilsøking enkel, selv for brukere med begrenset erfaring. En annen stor fordel er dens brede støtte i utviklermiljøet på nett.

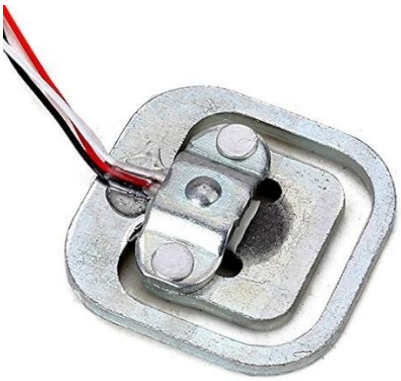
I vårt prosjekt benyttes Wemos D1 Mini til å samle inn og sende data fra ulike sensorer til en ekstern nettside over mobilnettet, takket være tilkoblingen til et 4G-modul. Den lave strømbruken og fleksible strømforsyningen (3.3V / 5V) gjør den også egnet for batteridrevne applikasjoner i områder uten strømtilførsel.

4.3 Måling av data

For å kunne overvåke bikuben på en effektiv måte, er det viktig med kontinuerlig og nøyaktig datainnsamling. Vi har valgt to typer sensorer i prosjektet: en for vekt og en for temperatur.

4.3.1 HX711 modul og vektceller

Vekten måles ved hjelp av fire vektceller av typen YZC-161B, som er plassert i hvert hjørne på undersiden av en plate som bikuben står på. Hver vektcelle har en kapasitet på 50 kg, noe som gir en samlet målekapasitet på 200 kg. [10]



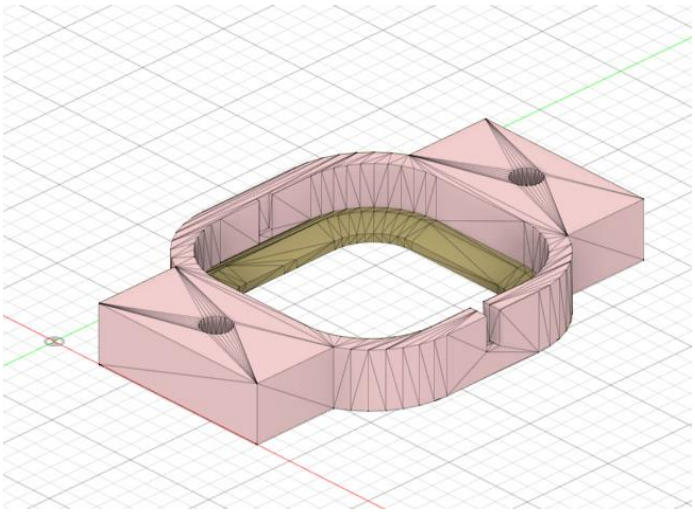
[11]

Figur 6: Vektcelle

Vektcellene fungerer ved hjelp av innebygde strekklapper, som er tynne elektriske ledere med en kjent motstand. Når disse lederne strekkes, blir de lengre og tynnere, noe som øker motstanden. Ved kompresjon blir de kortere og tykkere, noe som reduserer motstanden. Når belastningen på platen endres, påvirkes strekklappene tilsvarende, og den resulterende motstandsendringen kan måles. Denne endringen er svært liten, men målbar, og er proporsjonal med vekten som påføres.

Konstruksjonsmessig består vektcellene av en indre og en ytre metallramme. For at målingen skal fungere, må den indre delen kunne bevege seg relativt i forhold til den ytre. Det er denne relative bevegelsen som skaper strekk i strekklappen. Dersom vektcellen festes direkte på en plate uten tilpasning, vil begge deler hvile mot underlaget og hindre bevegelse, noe som gjør måling umulig.

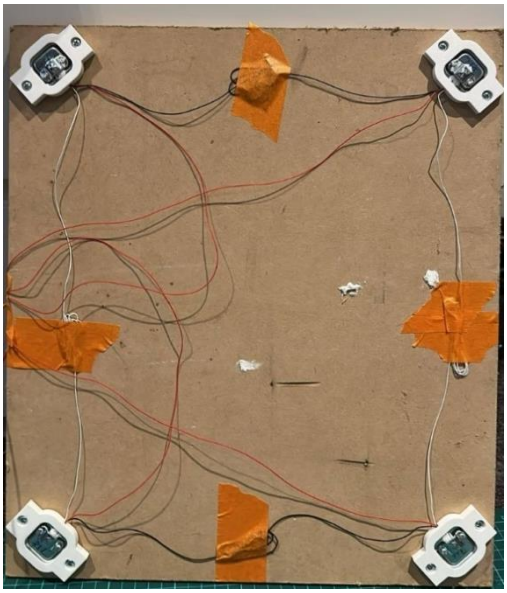
For å løse dette har vi 3D-printet en spesialtilpasset monteringsramme som løfter den ytre delen av vektcellen litt opp. Dette gir den indre delen rom til å bevege seg fritt når det påføres vekt. Monteringsrammen er basert på en åpen kildekode-design funnet på nett, og ble 3D-printet hos HVL Skape. [12]



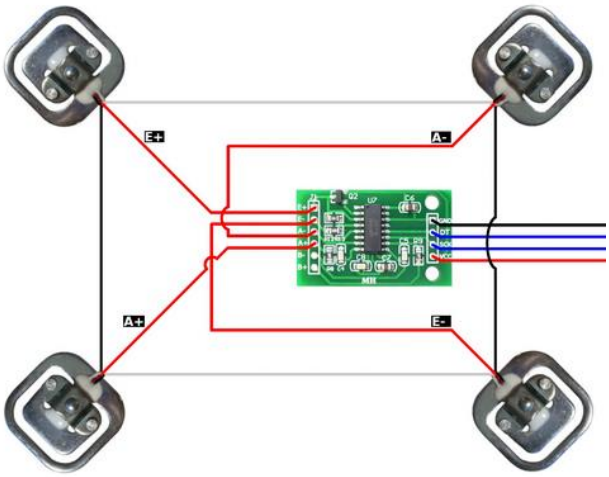
Figur 7: En visuell skisse av beina til cellene

Figur 8: Printer prosessen av beina hos HVL skape

De fire vektcellene kobles sammen slik at strekklappene står i en målebro som er en elektrisk målekrets som er svært følsom for små motstandsvariasjoner. Når vekten på platen endres, fører det til en ubalanse i målebroen, noe som resulterer i en spenningsforskjell.



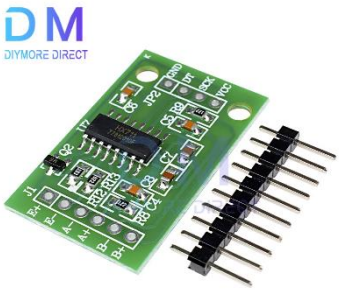
Figur 9: Prototype av oppsett av vektcellen



[13]

Figur 10: Koblingsoppsett av vektceller med HX711 modul

Denne spenningen er svært liten (typisk i millivolt-området) og må forsterkes og digitaliseres før den kan leses av en mikrokontroller. Vi bruker HX711, som fungerer som både en signalforsterker og en ADC (Analog-Digital Converter), slik at vi kan lese og beregne den faktiske vekten på platen nøyaktig med en mikrokontroller. [14]



[15]

Figur 11: HX711 modul

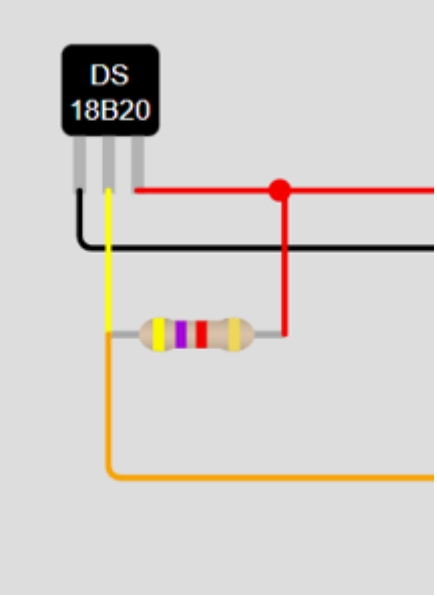
4.3.2 DS18B20 temperatursensor

For å måle temperaturen i og rundt bikuben bruker vi DS18B20, en digital temperatursensor med høy presisjon og god langsiktig stabilitet, som egner seg godt for utendørs og varierende miljøer. Den kan måle temperaturer fra –55 °C til +125 °C, med ±0.5 °C nøyaktighet i det mest relevante området (–10 °C til +85 °C).

Sensoren benytter 1-Wire-protokollen, som lar både data og strøm gå gjennom én enkelt datalinje i tillegg til GND, noe som forenkler koblingen. Temperaturverdiene digitaliseres og lagres automatisk i et internt scratchpad-minne, som mikrokontrolleren enkelt kan lese av.

DS18B20 krever svært få eksterne komponenter. I vårt tilfelle har vi brukt en 4.7 kΩ pull-up-motstand mellom datalinjen og 5V, for å sikre pålitelig signaloverføring. Pull-up-motstanden holder signalet høyt når ingen enhet trekker det lavt, og hindrer ustabil kommunikasjon.

Sensoren er vanntett og innkapslet i rustfritt stål, og er plassert slik at den er godt beskyttet, men likevel eksponert for luftstrømmen inne i kuben. Dette gir verdifulle og nøyaktige målinger som bidrar til å overvåke bienes trivsel og temperaturforholdene i sanntid. Ved hjelp av DS18B20 får vi altså presise og stabile temperaturmålinger i sanntid, noe som er avgjørende for å tolke miljøet inni bikuben og sette dette i sammenheng med vektmålingene fra HX711. [16]



[26]

Figur 12: Kretsoppkobling av temperaturføler *Figur 13: Bilde av valgt DS18B20*

4.4 Mekanisk sammensetning

Vi har lagt vekt på å gjøre den mekaniske og elektroniske sammensetningen så enkel og brukervennlig som mulig, slik at montering og installasjon skal kunne gjennomføres med minimalt arbeid. For å oppnå dette har vi designet et egendefinert kretskort (PCB) som samler de fleste nødvendige komponentene på ett sted. Dette inkluderer blant annet HX711-modulen, spenningsregulatorer, motstander og øvrige nødvendige kretselementer.

Ved å samle komponentene på ett kretskort, blir montering betydelig forenklet og det eneste brukeren trenger å gjøre er å lodde på innganger og utganger til de eksterne komponentene, som sensorer og strømforsyning.

På grunn av lang leveringstid og gjentatte endringer i kretsdesignet underveis i utviklingen, fikk vi dessverre ikke testet det ferdige kretskortet innen prosjektets frist. Likevel har vi sørget for at designet er klart til produksjon, og vi har gjort det mulig å bestille kretskortet direkte fra leverandøren. Vi har vurdert to ulike løsninger for oppsettet av kretskortet for oppdragsgiver, hver med sine egne fordeler og ulemper:

1. Bestille kretskort og komponenter separat:
Denne løsningen innebærer at man bestiller selve kretskortet og komponentene hver for seg, og deretter lodder komponentene på manuelt. Dette gir stor fleksibilitet og kontroll over både produksjonsprosessen og valg av komponenter. En utfordring med denne metoden er at mange komponenter ofte må kjøpes i bulk (f.eks. 10 eller 100 stk), noe som kan føre til overskudd dersom man kun skal lage et enkelt kort. Samtidig gir dette også en fordel i form av reservekomponenter, som kan være nyttige dersom man mister eller skader komponenter under montering.

2. Bestille ferdig loddet kretskort (montert PCB):
Den andre løsningen er å bestille et ferdig loddet kretskort fra en leverandør. Denne prosessen er mer omfattende og krever god innsikt i produksjonsflyten. Det er nødvendig med tett kommunikasjon med produsenten, da det må utveksles produksjonsfiler (som Gerber-filer og BOM-lister), samt godkjenning av eventuelle alternative komponenter, forslag til omplasseringer og andre produksjonsrelaterte avklaringer. Det er viktig å merke seg at selv om kretskortet blir levert delvis montert, er det ingen garanti for at alle komponenter loddes på automatisk. Derfor må man være forberedt på å ettermontere enkelte komponenter selv.

For å sikre enkel videreføring har vi utarbeidet en detaljert bruksanvisning som beskriver hvordan kretskortet kan bestilles, monteres og tas i bruk. Dette gjør det enkelt for andre å videreutvikle, produsere og implementere løsningen.

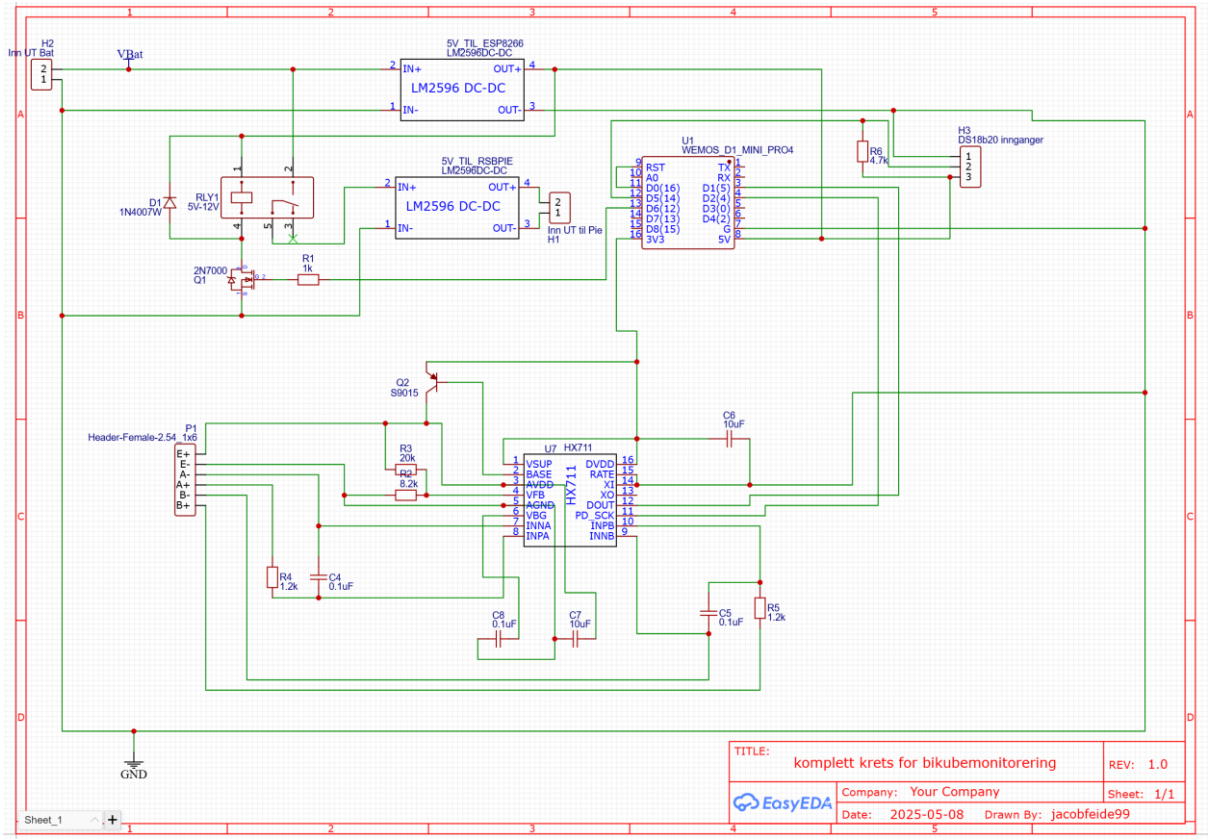
4.4.1 Utvikling av kretskortet

Til design og utvikling av kretskortet benyttet vi den nettbaserte plattformen EasyEDA. Dette verktøyet gir en helhetlig og brukervennlig løsning for elektronikkdesign direkte i nettleseren, uten behov for lokal installasjon av programvare.

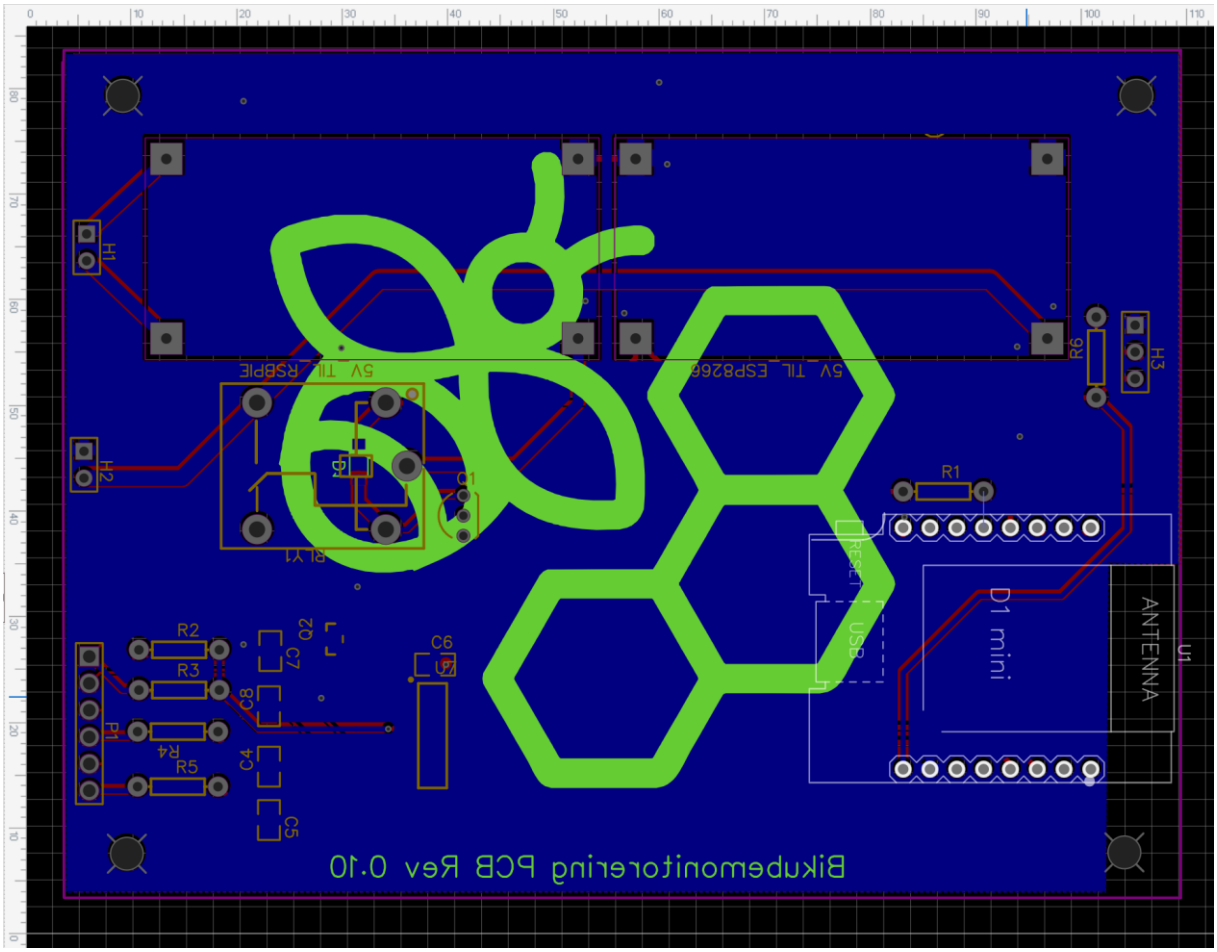
Gjennom EasyEDA fikk vi tilgang til et omfattende bibliotek med elektroniske komponenter, noe som gjorde det enkelt å finne og integrere de nødvendige delene – som HX711-modulen, spenningsomformere, motstander og tilkoblingspunkter. Alle komponentene ble først samlet i et komplett hoved kretsskjema som representerer den logiske koblingen mellom de ulike delene.

EasyEDA tilbyr deretter en funksjon som automatisk overfører kretsskjemaet til et fysisk PCB-design. Her kan man nøyaktig plassere komponentene, rute forbindelser, og optimalisere utformingen av selve kretskortet. Denne funksjonaliteten gjorde det mulig for oss å designe et funksjonelt og plassbesparende kretskort klart for produksjon.

Når designet er ferdigstilt, gir EasyEDA mulighet for å sende kortet direkte til produksjon via tilknyttede PCB/PCBA-fabrikker, noe som forenkler hele prosessen fra ide til fysisk prototype.



Figur 14: Kretsskjema for kretskort



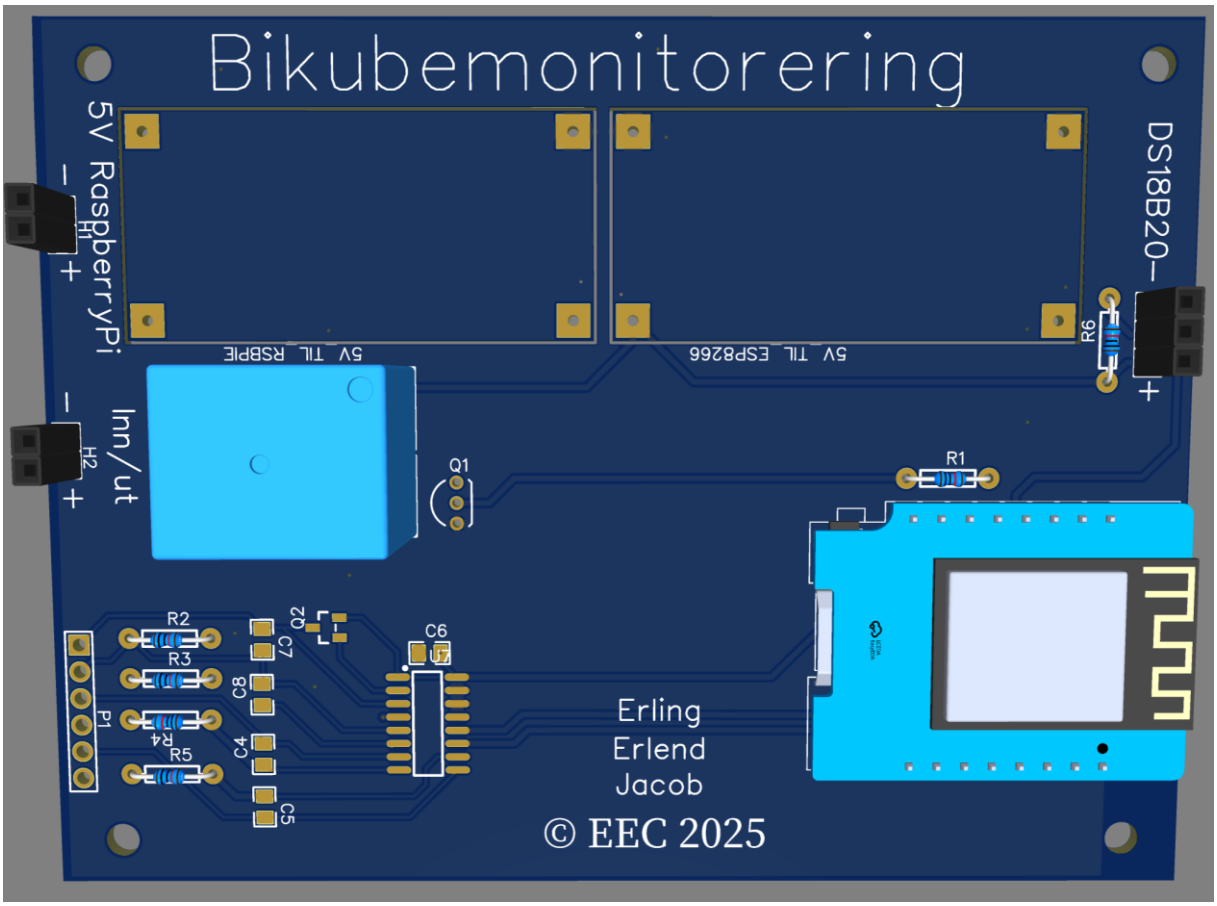
Figur 15: Kretskortdesign(PCB)

4.4.2 Bruk av kretskort

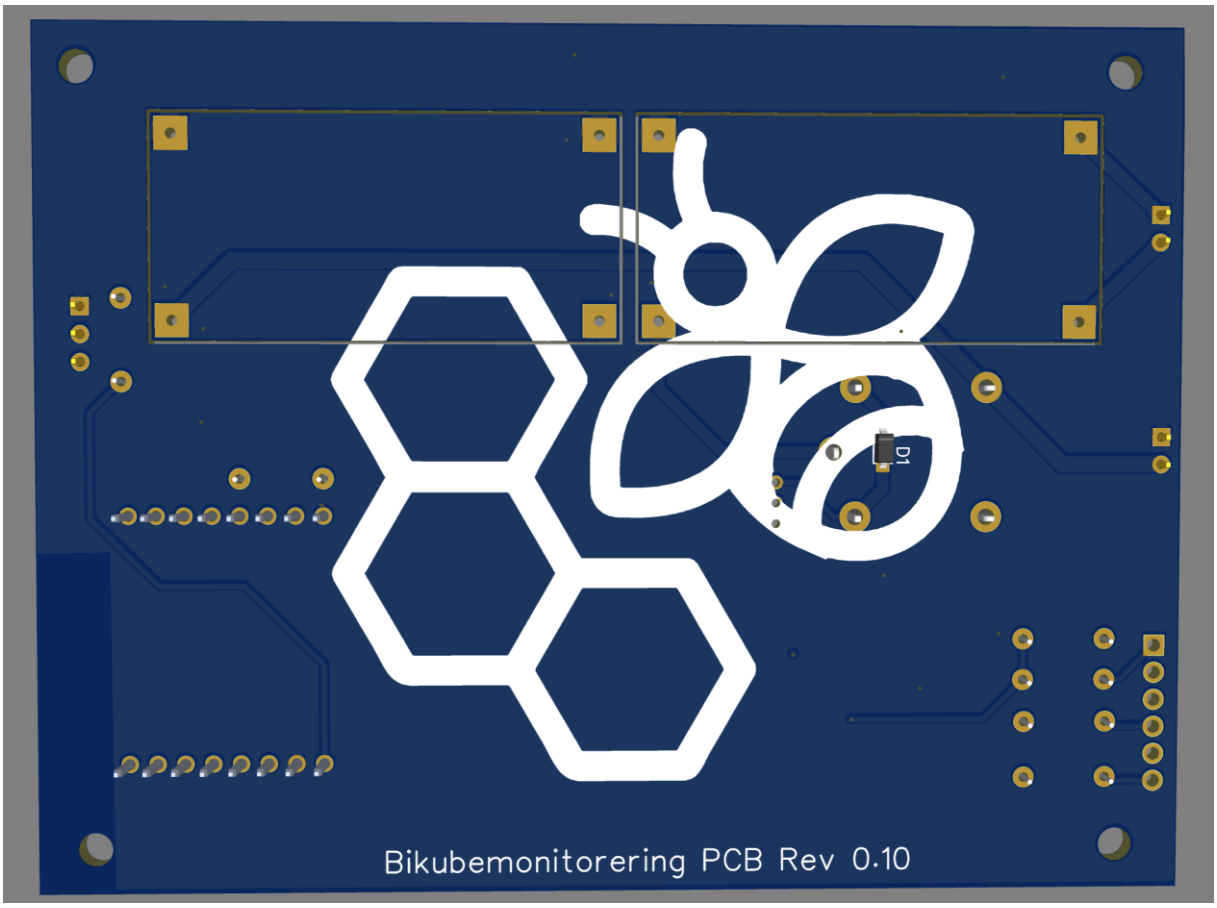
Kretskortet er ferdig utviklet og klart for produksjon, men det er foreløpig kun bestilt og ikke testet. Dette er grunnet mange endringer som gjorde at kretskortet ikke ble levert før prosjektets innleveringsfrist. Vi har samtidig lagt opp til at brukeren selv kan bestille kortet ved behov. Bestillingsprosessen er enkel og brukervennlig, og kan gjennomføres via EasyEDA eller tilsvarende tjenester. Prisen for kortet er omtrent 7 USD, men dette kan variere avhengig av fraktmetode og leveringsadresse.

Kortet er designet med et sterkt fokus på kompakt størrelse og brukervennlighet. Det totale arealet er på kun 105 mm x 80 mm, noe som gjør det svært plassbesparende og egnet for integrasjon i mindre systemer. Til tross for det kompakte formatet, er alle tilkoblingspunkter tydelig merket, noe som gjør monteringsprosessen enkel og reduserer risikoen for feilkoblinger.

Nedenfor vises en 3D-modell av kretskortet slik det er designet i EasyEDA.



Figur 16: 3D visualisering av kretskort ovenfra



Figur 17: 3D visualisering av kretskort nedenfra

B	C	D	E
Name	Designator	Footprint	Quantity
4.7k	R6	R_AXIAL-0.3	1
8.2k	R2	R_AXIAL-0.3	1
20k	R3	R_AXIAL-0.3	1
1.2k	R4,R5	R_AXIAL-0.3	2
0.1uF	C4,C5,C8	0805	3
10uF	C6,C7	0805	2
1k	R1	R_AXIAL-0.3	1
LM2596DC-DC	5V_TIL_ESP8266,5V_TIL_RSBPIE	LM2596 DC-DC	2
Inn UT til Pie	H1	HDR-F-2.54_1X2	1
Inn UT Bat	H2	HDR-F-2.54_1X2	1
DS18b20 innganger	H3	HDR-F-2.54_1X3	1
Header-Female-2.54_1x6	P1	HDR-6X1/2.54	1
2N7000	Q1	2N7000	1
S9015	Q2	SOT-23(SOT-23-3)	1
5V-12V	RLY1	RELAY-TH_SRD-XXVDC-XL-C	1
WEMOS_D1_MINI_PRO4	U1	DIP-16_WEMOS_D1_MINI_PRO4	1
HX711	U7	SOP-16_150MIL	1
1N4007W	D1	SOD-123_L2.8-W1.8-LS3.7-RD	1

Figur 18: BOM for komponenter på kretskortet

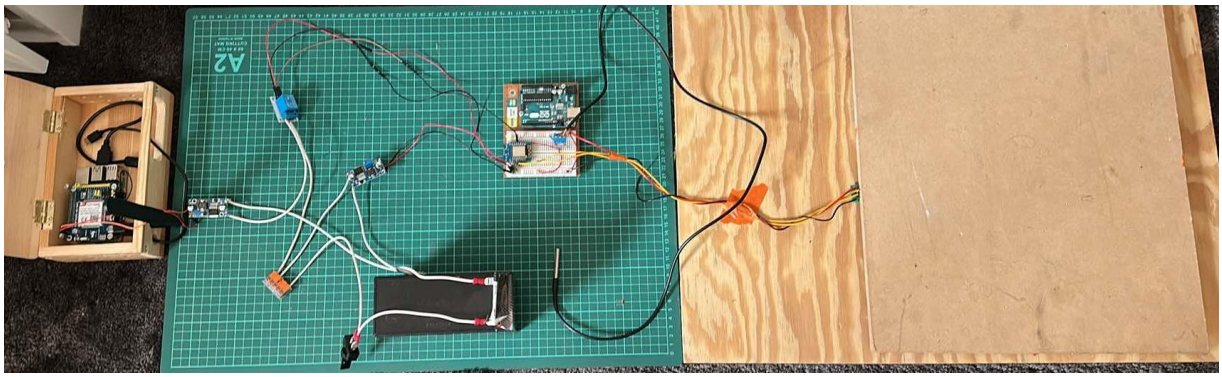
4.5 Vår prototype

Under testfasen benyttet vi løse komponenter og et koblingsbrett for å sette sammen kretsen. Dette oppsettet var utelukkende ment som en midlertidig testplattform og er ikke egnet for utendørs bruk. Hensikten var å verifisere at alle komponentene fungerte som forventet før vi eventuelt overførte designet til et permanent kretskort.

Fordelen med å bruke koblingsbrett var at vi enkelt kunne bytte ut komponenter og gjøre justeringer i koblingene underveis. Dette ga oss fleksibilitet til å utforske og implementere bedre løsninger etter hvert som prosjektet utviklet seg.

Ettersom vi gjorde flere endringer sent i prosjektet, fikk vi dessverre ikke tid til å implementere og teste selve kretskortet. Likevel er det designet for å fungere optimalt og skal tas i bruk så snart det er levert.

Oppsettet bestod av alle nødvendige komponenter koblet sammen på ett hoved-koblingsbrett. Nedenfor vises ett bilde som illustrerer hvordan oppsettet så ut i testfasen.



Figur 19: Bilde av komplett oppsett (Viktig å se vekk fra arduinoen ovenfor koblingsbrett som ikke er en del av oppsettet)

4.6 Programkode og sending av data

Sammen med vår mikrokontroller benytter vi en Raspberry Pi som er konfigurert til å fungere som en ruter. Dette gjør det mulig for mikrokontrolleren å bruke sin innebygde WiFi-modul til å sende data via ruteren. For at dette oppsettet skulle fungere, var det nødvendig at hver IoT-enhet hadde sin egen tilpassede programkode. Nedenfor beskriver vi hvor og hvordan disse programkodene er utviklet og implementert.

4.6.1 Rasberry Pi sammen med Waveshare 4G-HAT

Raspberry Pi-en med Waveshare 4G-HAT fungerer som et autonomt 4G-basert WiFi-hotspot ved å bruke Modemmanager og networkmanager, to standardverktøy i Linux for håndtering av nettverkstilkoblinger. For at 4G-fobindelsen skal fungere, må det settes inn et aktivt SIM-kort i modemet, med gyldig dataplan fra en mobiloperatør. Når enheten skrus på, oppdager modemmanager automatisk 4G-modemet på HAT-en og kommuniserer med det via MBIM-protokollen (Kommunikasjon via USB). Dette skjer i bakgrunnen uten manuell input, da modemmanager er konfigurert til å identifisere SIM-kortet, hente operatørens APN-informasjonen, og etablere en 4G-forbindelse. Networkmanager, som er ansvarlig for nettverkskonfigurasjonen på systemet, tar deretter over og setter 4G-tilkoblingen som hovedforbindelsen til internett. Dette eliminerer behovet for manuell PPP-oppsett (Point-to-point protocol), da modemmanager og networkmanager sammen forenkler kompleksiteten ved mobilnettverkstilkoblinger.

Etter at 4G-tilkoblingen er aktiv, tar manuelt konfigurerte tjenester over for å sette opp WiFi-hotspotet. Hostapd brukes til å transformere Raspberry Pi'ens WiFi grensesnitt (wlan0) til et tilgangspunkt med egendefinert SSID og sikkerhetsinnstillinger. Dette krever redigering av hostapd filer for å spesifisere forskjellige parametere som navnet på hotspotet og passordet. Deretter konfigureres dnsmasq som er en dedikert DHCP- og DNS-server gjennom dns filene der man tydelig definerer IP-addresseområdet, gateway, og DNS-servere. Dette gir full kontroll over nettverksparametere, i motsetning til networkmanager innebygde automatikk for dette. Vi valgte å ikke bruke networkmanagers automatiske håndtering fordi 4G-forbindelsen tidligere ble avsluttet når vi forsøkte å bruke samme tilkoblingen til å sette opp hotspot.

For å sikre korrekt ruting mellom WiFi-enheter og 4G-nettverket, aktiveres IP-forwarding i filene og NAT-regler legges til manuelt i iptables.

Systemd er et program som starter og kontrollerer systemtjenester og prosesser ved oppstart.
Tjenestene hostapd og dnsmasq aktiveres som systemd-enheter for å starte automatisk ved oppstart. Dette krever justering av systemd-filer for å sikre at de kjører etter at 4G-tilkoblingen er etablert. For eksempel legges det til en utsettelse i filene for å unngå timing-problemer.

Selv om modemmanager og networkmanager håndterer 4G-delen automatisk, gir den manuelle konfigurasjonen av hostapd og dnsmasq større fleksibilitet – som å tilpasse DHCP-leietid, blokkere spesifikke MAC-addresser, eller aktivere detaljert logging for feilsøking. Sikkerhet håndteres via WPA2-kryptering i hostapd, kombinert med brannmurregler som blokkerer uønsket trafikk til/fra bikubens sensorer.

Denne hybridtilnærmingen kombinerer styrken til modemmanagers automatiserte 4G-håndtering med presis kontroll over lokalnettverket, noe som er avgjørende for en stabil bikubedrift der både pålitelighet og tilpasningsevne er nødvendig. Alt sammen sikrer at systemet fungerer som en selvkjørende «bro» mellom 4G og WiFi uten avhengighet til grafiske verktøy eller tredjepartsskript. [17]

4.6.2 Programkode til ESP8266(WEMOS D1 mini)

I dette prosjektet benyttes en WEMOS D1 mini, en kompakt mikrokontroller med innebygd WiFi, for å hente inn sanntidsdata fra en temperatursensor (DS18B20) og en vektesensor (HX711 + lastcelle). Dataene sendes til en skylagringstjeneste (ThingSpeak) for videre analyse og visualisering. Mikrokontrolleren er programmert i C++ ved hjelp av Arduino IDE.

Koden er strukturert på en måte som sikrer:

- Minimalt strømforbruk
- Pålitelig kommunikasjon over ustabilt nett
- Robust håndtering av sensoravlesning
- Vedvarende kalibrering ved bruk av EEPROM
- Sikker sending til ThingSpeak

Under følger en forklaring på de viktigste kodekomponentene, inkludert tekniske begrunnelser.

Inkludering av biblioteker

```
1   #include <ESP8266HTTPClient.h>
2   #include <ESP8266WiFi.h>
3   #include <HX711_ADC.h>
4   #if defined(ESP8266)|| defined(ESP32) || defined(AVR)
5   #include <EEPROM.h>
6   #include <OneWire.h>
```

ESP8266WiFi.h og ESP8266HTTPClient.h gir nødvendig funksjonalitet for å koble til WiFi og sende HTTP-forespørsler. Disse er spesialtilpasset ESP8266-arkitekturen og brukes i stedet for generiske Arduino-biblioteker.

HX711_ADC.h er et bibliotek optimalisert for ADC-kommunikasjon mellom mikrokontroller og lastcelleforsterkeren HX711. Biblioteket inkluderer avansert filtrering og automatisk tare-funksjon.

EEPROM.h gir varig lagring av kalibreringsdata, noe som er essensielt siden HX711-kalibrering og offset ikke bør gå tapt ved omstart.

OneWire.h brukes fordi DS18B20 temperatursensoren opererer over én enkelt datalinje (One-Wire-protokoll). Biblioteket håndterer timing og bit-baserte kommandoer automatisk.

WiFi-tilkobling

```
12  #define WIFI_SSID "HyttewifiM"
13  #define WIFI_PASSWORD "Fjelly714"
```

SSID og passord må være hardkodet for å kunne etablere automatisk tilkobling uten brukerinteraksjon. I miljøer uten skjerm eller tastatur (som en bikube), er det ikke mulig å konfigurere nettverk ved oppstart. Derfor er dette en nødvendig forenkling.

ThingSpeak-konfigurasjon

```
18  const char* THINGSPEAK_API_KEY = "REHEU1FGRAGQ9FHU";
19  const char* THINGSPEAK_URL = "http://api.thingspeak.com/update";
```

ThingSpeak krever en unik API-nøkkel for at data skal sendes til riktig kanal.

Sensoroppsett og pinner

```
24  #define ONE_WIRE_BUS 14
25  OneWire ds(ONE_WIRE_BUS);
```

OneWire ds() initialiserer kommunikasjonen med DS18B20. DS18B20 kan kobles parallelt med flere sensorer, men i dette prosjektet brukes én.

```
30  const int HX711_dout = 4;
31  const int HX711_sck = 5;
32  HX711_ADC LoadCell(HX711_dout, HX711_sck);
```

GPIO4 (D2 på mikrokontroller) og GPIO5 (D1 på mikrokontroller) er standardpinner for data- og klokkelinje til HX711. HX711_ADC er en forbedret versjon som filtrerer og glatter ut støy, som er vanlig i analog lastcellemåling.

EEPROM

```
34  const int addr_magic = 0;
35  const int addr_calibration = 4;
36  const int addr_offset = 8;
```

For at lastcellen skal måle nøyaktig over tid, må den kalibreres. Men etter et strømbrudd ville alle kalibrerte verdier vært tapt uten EEPROM. Et “magic number” på adresse 0 sjekkes det om EEPROM er initialisert.

Kalibreringsfaktor og tare-offset lagres på adresse 4 og 8 som float.
Dermed slipper brukeren å kalibrere manuelt hver gang, noe som er viktig i en fjerntliggende installasjon.

setup() – initialisering av hele systemet

```
56  void setup() {
57      Serial.begin(115200);
58
59      pinMode(relayPin, OUTPUT);
60      digitalWrite(relayPin, HIGH);
61
62      connectToWiFi();
63
64      LoadCell.begin();
```

Seriell kommunikasjon gjør feilsøking mulig i tidlig utviklingsfase.

Reléet aktiveres ved oppstart. Dette brukes til å slå på reléet og gir strøm til ekstern ruteren.

connectToWiFi() forsøker å koble til nettverk. Det er viktig å gjøre dette før sensordata samles inn, ellers vil ikke dataene kunne sendes.

loop()

```
112 void loop() {
113     // Les teller fra EEPROM
114     byte wakeCounter = EEPROM.read(addr_wake_counter);
115
116     if (wakeCounter >= max_wake_count) {
117         digitalWrite(relayPin, HIGH);
118         delay(200);
119
120         float temperature = readTemperature();
121         float weight = readWeight();
122         Serial.println("Weight: " + String(weight) + " g");
123
124         if (temperature != -127.0) {
125             Serial.println("Temp: " + String(temperature) + " °C");
126             sendToThingSpeak(temperature, weight);
127         } else {
128             Serial.println("Ugyldig temperatursensor-avlesning.");
129         }
130
131         digitalWrite(relayPin, LOW); // Slå av releet
132
133         // Nullstill telleren etter sending
134         wakeCounter = 0;
135         EEPROM.write(addr_wake_counter, wakeCounter);
136         EEPROM.commit();
137     } else {
138         Serial.println("Sender ikke data denne gangen. Teller: " + String(wakeCounter));
139         wakeCounter++;
140         EEPROM.write(addr_wake_counter, wakeCounter);
141         EEPROM.commit();
142     }
143
144     WiFi.disconnect();
145     delay(1000);
146     Serial.println("Sover i 60 minutter...");
147     ESP.deepSleep(3600000000); // 60 min i mikrosekunder
```

I **loop()** hentes og gjøres om måledata til en enkel streng som benyttes til å sende en strukturert og enkel «setning» til thingspeak hvor data kan bli lest på en forståelig måte for videre tolkning. Det er også inkludert feilmeldinger som gir tilgang til feilsøking under testing.

Mikrokontrolleren bruker svært lite strøm i deepSleep()-modus, noe som gjør den ideell for batteridrevne løsninger. I dette prosjektet våkner mikrokontrolleren automatisk opp med jevne mellomrom, hvert 60. minutt. Men for å spare både strøm og begrense unødvendig datatrafikk, sendes målingene til ThingSpeak kun hver 12. gang enheten våkner. Det tilsvarer omtrent én gang hver 12. time.

Mellom sendingene tar systemet og sjekker om det er tid for å sende, basert på en intern teller lagret i EEPROM. Etter målingen går mikrokontrolleren tilbake til deepSleep igjen.

readTemperature() – DS18B20 måling

```
151  float readTemperature() {
152      byte data[9], addr[8];
153      float celsius;
154      if (!ds.search(addr)) {
155          ds.reset_search();
156          return -127.0;
157      }
158      ds.reset();
159      ds.select(addr);
160      ds.write(0x44, 1); // Start conversion
161      delay(1000);
162      ds.reset();
163      ds.select(addr);
164      ds.write(0xBE); // Read Scratchpad
165      for (int i = 0; i < 9; i++) {
166          data[i] = ds.read();
167      }
168      int16_t raw = (data[1] << 8) | data[0];
169      celsius = (float)raw / 16.0;
170      return celsius;
171  }
```

readTemperature() er en funksjon som leser data fra temperaturføleren. Funksjonen brukes for å finne riktig data pin på ESP8266 og inkluderer feil behandling om temperaturdataen gir en ulogisk eller ugyldig verdi. Funksjonen brukes i **loop()** og returnerer en enkel temperaturverdi.

readWeight() – HX711-vektmåling

```
379  float readWeight() {
380    //Read initial weight some times to stabilice HX711
381
382    int count = 30;
383    while (count > 0)  // repeat until count is no longer greater than zero
384    {
385
386      if (LoadCell.update()) {
387
388        Serial.println("Stabelizing : " + String(LoadCell.getData()) + " g");  // Return weight data from HX711
389        count = count - 1;
390      }
391    }
392
393    LoadCell.update();
394
395    return LoadCell.getData();  // Return weight data from HX711
396  }
```

update(**)-funksjonen henter nye data fra vektsensoren og kjører intern filtrering for å redusere støy og forstyrrelser. **getData() returnerer deretter den sist filtrerte og stabile vekten i gram. Dette gir en jevn og pålitelig avlesning, som er robust mot miljømessige forstyrrelser som vibrasjoner eller vind.

Funksjonen brukes i **loop()** og returnerer en enkel, stabil vektverdi. For å sikre at målingen er nøyaktig, er det gjennomført en stabiliseringsløkke som tar flere målinger over tid. Dette gjør at vekten får stabilisere seg før en gjennomsnittsverdi beregnes og sendes videre til ThingSpeak for visualisering.

sendToThingSpeak() – opplasting av data

```
185  void sendToThingSpeak(float temperature, float weight) {
186      if (WiFi.status() == WL_CONNECTED) {
187          HTTPClient http;
188          WiFiClient client;
189
190          String url = String(THINGSPEAK_URL) + "?api_key=" + THINGSPEAK_API_KEY +
191                       "&field1=" + String(temperature) + "&field2=" + String(weight);
192
193          http.begin(client, url);
194          int httpCode = http.GET();
195          if (httpCode == 200) {
196              Serial.println("✅ Data sendt til ThingSpeak.");
197          } else {
198              Serial.println("❌ Feil ved sending til ThingSpeak.");
199          }
200          http.end();
201      } else {
202          Serial.println("❌ Ikke koblet til WiFi.");
203      }
204  }
```

Det brukes en enkel HTTP GET-forespørsel. Dette er valgt fordi det er mer pålitelig og lettere å feilsøke enn POST i en ustabil nettverkssituasjon. Hvis forbindelsen feiler, sendes det ikke data, og mikrokontrolleren forsøker på nytt i neste syklus. Funksjonen brukes i **loop()**.

Konklusjon

Vi har forsøkt å sette opp programmet/koden for moduler struktur der hver funksjon er adskilt, noe som gir bedre lesbarhet og gjør det lettere å endre og vedlikeholde koden. Koden er også strukturert for å spare mest mulig strøm under kjøring av koden for at batteriet skal ha lengst mulig levetid. Vi har også forsøkt å gjøre koden robust slik at det oppstår så få feil som mulig. Vi har også gjort det slik at om feil oppstår så er de gjenkjennelig og lett å finne fram til. For testing og videre arbeid er det også brukt en del konsollutskrifter som viser hva som skjer underveis og varsler dersom noe går galt.

4.7 Datavisualisering og bruk av ThingSpeak

For visualisering og lagring av måledata har vi valgt å benytte den nettbaserte IoT-plattformen ThingSpeak. ThingSpeak er utviklet av MathWorks og er en skybasert tjeneste som gjør det mulig å samle inn, lagre, analysere og visualisere sanntidsdata fra sensorer og IoT-enheter. Tjenesten er mye brukt i undervisning og hobbyprosjekter, og er godt egnet for vårt behov, da den er gratis ved personlig bruk og tilbyr enkle integrasjoner.

4.7.1 Innsending av data

ThingSpeak fungerer ved at man oppretter en egen kanal, hvor hver kanal kan ha opptil åtte felt for datainnsamling. Tilgangen til kanalen styres ved hjelp av API-nøkler, én for skriving (Write API Key) og én for lesing (Read API Key). I vårt tilfelle sender vi en enkel datastreng via HTTP GET- forespørsler til ThingSpeak sin API, hvor verdiene skilles og tilordnes riktig felt i databasen.

Eksempel på en HTTP GET-forespørsel:

https://api.thingspeak.com/update?api_key=DIN_WRITE_API_KEY&field1=23.5&field2=68.2

Denne forespørselen sender for eksempel en temperaturverdi til field1 og en vektverdi til field2.

4.7.2 Visualisering

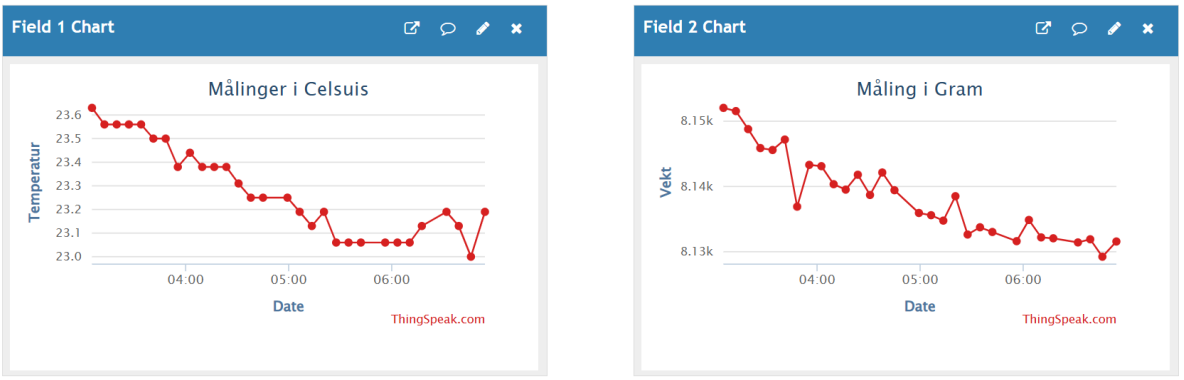
Når dataene er lagret, tilbyr ThingSpeak innebygd støtte for grafisk visning. Brukeren kan enkelt lage linjediagrammer og andre visualiseringer direkte i nettleseren, uten behov for ekstern programvare. Disse grafene oppdateres automatisk når nye data mottas og kommer med tidsstempling som tillater brukeren å se når målingene ble tatt.

Brukeren kan også sette inn filtre, tilpasse akser, vise tidsintervaller og aktivere triggere eller varslinger basert på bestemte verdier. Dette gjør det enkelt å overvåke tilstanden til systemet over tid.

Tilpasning og videre bruk

I bruksanvisningen vår har vi inkludert en steg-for-steg-guide til hvordan man:

- Innlogging av en opprettet ThingSpeak-konto
- Anskaffelse av riktig kanal



Figur 20: Eksempel på temperatur og vektmåling tatt fra ThingSpeak

4.8 Bruksanvisning

Som en del av kravspesifikasjonene var det etterspurt en bruksanvisning som oppdragsgiver kan benytte for å sette opp og gjennomføre flere slike systemer på egen hånd. Vi har derfor utarbeidet en detaljert og trinnvis veiledning som beskriver hvordan systemet skal bygges, hvilke komponenter som må bestilles, og hvordan disse skal kobles sammen. I tillegg er det inkludert instruksjoner for nedlastning og installasjon av nødvendig programvare, samt hvordan systemet programmeres.

Bruksanvisningen er utformet med tanke på brukere uten teknisk bakgrunn, og vi har lagt vekt på å gjøre den så forståelig og brukervennlig som mulig. Innholdet er spesielt tilpasset oppdragsgiverens behov.

Bruksanvisningen er vedlagt som et eget dokument i apendiks C.2.

5 Testing

I løpet av prosjektutviklingen var det nødvendig med omfattende testing for å sikre at alle komponentene fungerte som forventet, både individuelt og i samspill. Vi startet med å teste hver enkelt komponent separat for å verifisere funksjonaliteten og identifisere eventuelle feil eller begrensninger. Dette gjorde det enklere å isolere problemer tidlig i prosessen.

Etter at de individuelle testene var gjennomført, begynte vi gradvis å integrere komponentene. Vi la særlig vekt på samspillet mellom sensorene, mikrokontrolleren og kommunikasjonssystemet, og testet hvordan de oppførte seg i kombinasjon. Dette trinnvise oppsettet gjorde det mulig å avdekke kompatibilitetsproblemer og justere systemet fortløpende, slik at vi til slutt oppnådde en stabil og velfungerende løsning.

5.1 Testing av Hx711 modul og Testing av DS18B20

Oppsettet for testing besto av ett enkelt oppsett der vi koblet til komponentene i en enkel krets. Vi brukte innebygde bibliotek for å forsikre oss om at det ikke var noe feil med koden under testingen. Vi testet komponentene hver for seg og senere samlet vi programmene sammen til ett fullstendig program.

Resultatene av testingen var som forventet og vi fikk de dataene vi ønsket. Vi prøvde ulike forhold og kondisjoner for å forsikre oss at vi fikk så bra resultat som mulig og slik at vi kunne unngå senere problemer og komplikasjoner.

Videre beskrives kjøring av program separat for DS18B20 temperaturføleren og Hx711 modulen. Og resultat med forholdene den ble eksponert til.

5.1.1 HX711 modul test

Vi kjørte et innebygd testprogram som tok nødvendige steg for å initialisere vektcellene. Her begynner man med å tarere vekten slik at den vet hva null er. Videre legger man en kjent vekt på, som den deretter bruker for å finne en kalibreringsverdi som den skal bruke videre. Under er ett bilde av oppstartsprosessen.

```
18:23:20.327 -> ▯Startup is complete
18:23:20.452 -> ***
18:23:20.452 -> Start calibration:
18:23:20.452 -> Place the load cell an a level stable surface.
18:23:20.452 -> Remove any load applied to the load cell.
18:23:20.452 -> Send 't' from serial monitor to set the tare offset.
18:23:27.313 -> Tare complete
18:23:27.313 -> Now, place your known mass on the loadcell.
18:23:27.313 -> Then send the weight of this mass (i.e. 100.0) from serial monitor.
18:32:26.179 -> Known mass is: 1970.00
18:32:28.026 -> New calibration value has been set to: 22.66, use this as calibration value
18:32:28.069 -> Save this value to EEPROM adress 0? y/n
18:32:30.311 -> Value 22.66 saved to EEPROM address: 0
18:32:30.311 -> End calibration
```

Figur 21: Bilde fra terminal av oppstartsprosess og kalibreringsfaktor av Hx711 modul.

Videre testet vi vektcellene med ulike gjenstander. Første test var med batteriet på 1970 g, andre test var når vi fjernet batteriet altså 0g. Tredje test var ved at en av oss på 73 kilo. På denne måten fikk vi testet lave og høye verdier samt om det var null vekt på. Altså en stress test. Ut ifra avleste verdier regnet vi ut gjennomsnittet og de største avvikene. Under er tabellen med resultatene.

Gjenstand	Vekt på gjenstand	Gjennomsnittlig målt verdi	Avvik	Tolkning
BatterI	1970g	1964,41g	-5,59g	Liten offset eller dårlig kalibrering
Ingen	0g	2,21g	+2,21g	Dårlig nullpunkt eller litt støy
Person	Ca. 73000g	73085,45g	+85,45g	Lite presisjon på reel vekt på person. (Badevekt)

Tabell 2

Basert på dataene ser vi at avvikene er svært små og målesystemet egner seg bra til å måle honning produksjon, vektendringer ved sverming, eller tap av bier i bikuben. Siden avviket er under 0.2% på sitt verste tilfelle, er dette mer enn godt til å overvåke disse endringene med høy presisjon.

```
13:51:14.189 -> Load_cell output val: 1968.62
13:51:14.189 -> Load_cell output val: 1967.78
13:51:14.189 -> Load_cell output val: 1966.67
13:51:14.223 -> Load_cell output val: 1964.71
13:51:14.223 -> Load_cell output val: 1963.47
13:51:14.263 -> Load_cell output val: 1965.20
13:51:14.263 -> Load_cell output val: 1964.53
13:51:14.263 -> Load_cell output val: 1964.80
13:51:14.263 -> Load_cell output val: 1965.25
13:51:14.299 -> Load_cell output val: 1964.36
13:51:14.299 -> Load_cell output val: 1962.98
13:51:14.338 -> Load_cell output val: 1961.65
13:51:14.338 -> Load_cell output val: 1962.27
13:51:14.338 -> Load_cell output val: 1962.71
13:51:14.338 -> Load_cell output val: 1962.36
13:51:14.375 -> Load_cell output val: 1963.16
```

Figur 22: Vekt data fra batteri på 1970g

```
13:53:48.052 -> Load_cell output val: -0.04
13:53:48.052 -> Load_cell output val: -1.16
13:53:48.093 -> Load_cell output val: -1.60
13:53:48.093 -> Load_cell output val: -2.71
13:53:48.093 -> Load_cell output val: -1.02
13:53:48.093 -> Load_cell output val: 0.49
13:53:48.129 -> Load_cell output val: 0.84
13:53:48.129 -> Load_cell output val: 2.67
13:53:48.160 -> Load_cell output val: 3.11
13:53:48.160 -> Load_cell output val: 4.04
13:53:48.160 -> Load_cell output val: 4.04
13:53:48.193 -> Load_cell output val: 4.40
13:53:48.193 -> Load_cell output val: 4.22
13:53:48.193 -> Load_cell output val: 5.15
13:53:48.230 -> Load_cell output val: 6.18
13:53:48.230 -> Load_cell output val: 6.75
```

Figur 23: Vekt data fra ingen vekt på vektsensor

```
14:14:38.301 -> Load_cell output val: 73106.78
14:14:38.301 -> Load_cell output val: 73102.91
14:14:38.334 -> Load_cell output val: 73097.14
14:14:38.334 -> Load_cell output val: 73093.27
14:14:38.375 -> Load_cell output val: 73093.67
14:14:38.375 -> Load_cell output val: 73096.30
14:14:38.375 -> Load_cell output val: 73098.07
14:14:38.375 -> Load_cell output val: 73099.19
14:14:38.409 -> Load_cell output val: 73096.78
14:14:38.409 -> Load_cell output val: 73091.41
14:14:38.450 -> Load_cell output val: 73084.43
14:14:38.450 -> Load_cell output val: 73076.26
14:14:38.450 -> Load_cell output val: 73065.41
14:14:38.450 -> Load_cell output val: 73058.30
14:14:38.486 -> Load_cell output val: 73053.91
14:14:38.486 -> Load_cell output val: 73053.33
```

Figur 24: Vekt fra en person på 73Kg

5.1.2 DS18B20 temperatur test

For å verifisere nøyaktigheten til vår DS18B20-temperatursensor gjennomførte vi en sammenlignende test ved hjelp av to eksterne temperatursensorer. Alle tre sensorene ble plassert i to ulike miljøer, et glass med isvann og en termos med nesten kokende vann.

Under vises bilder av oppsettet, samt resultatene fra målingene.

Væsketype	Gjennomsnittlig temperatur fra DS18B20	Ekstern Sensor 1	Ekstern Sensor 2
Isvann	2,51˚C	2 ˚C	4 ˚C
Nært kokende vann	81,37˚C	80˚C	82 ˚C

Tabell 3 Resultat av testing på ulike temperaturfølere

Basert på tabellen over ser vi at DS18B20-sensoren leverer realistiske og konsistente målinger i forhold til de to eksterne referansesensorene. Temperaturavvikene er små og ligger innenfor en akseptabel feilmargin. Dette indikerer at vår sensor fungerer som forventet og gir pålitelige resultater.

Dermed kan vi konkludere med at DS18B20-sensoren er godt egnet for videre bruk i vårt prosjekt, og at den gir et troverdig bilde av temperaturforholdene i både lave og høye temperaturområder.

Figur 25: Bilde av oppsett på måling av nært kokende vann.

```
17:37:09.909 -> 82.81
17:37:11.647 -> 82.75
17:37:13.362 -> 82.75
17:37:15.061 -> 82.75
17:37:16.782 -> 82.75
17:37:18.492 -> 82.69
17:37:20.181 -> 82.69
17:37:21.903 -> 82.69
17:37:23.581 -> 82.69
17:37:25.331 -> 82.63
17:37:27.000 -> 82.63
17:37:28.743 -> 82.56
17:37:30.442 -> 82.56
17:37:32.159 -> 82.50
17:37:33.853 -> 82.50
17:37:35.584 -> 82.50
```

Figur 26: Temperaturdata fra DS18B20 i nært kokende vann.

Figur 27: Bilde av oppsett på måling av isvann

```
17:42:11.649 -> 2.50
17:42:13.378 -> 2.50
17:42:15.036 -> 2.56
17:42:16.763 -> 2.56
17:42:18.459 -> 2.50
17:42:20.113 -> 2.50
17:42:21.852 -> 2.50
17:42:23.535 -> 2.50
17:42:25.191 -> 2.50
17:42:26.915 -> 2.50
17:42:28.620 -> 2.50
17:42:30.308 -> 2.50
17:42:31.962 -> 2.50
17:42:33.686 -> 2.50
17:42:35.348 -> 2.50
17:42:37.056 -> 2.50
```

Figur 28: Temperaturdata fra DS18B20 i isvann

5.2 Testing av Raspberry Pi med Waveshare 4G-HAT

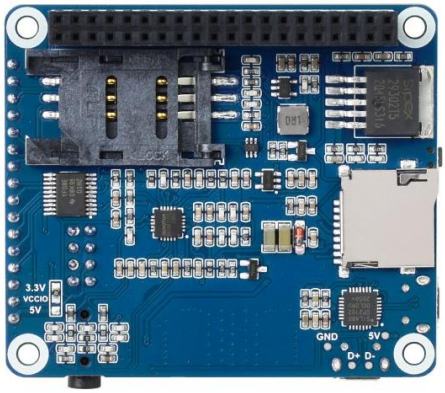
For å verifisere funksjonaliteten til 4G-løsningen i prosjektet, ble raspberry Pi med Waveshare 4G-HAT testet i flere omganger og under ulike forhold. Første test ble gjennomført hjemme, der raspberry Pi ble koblet opp med korrekt APN-konfigurasjon. For å sjekke ytelsen ble en iphone koblet til WiFi hotspoten som ruteren delte, og det ble kjørt en hastighetstest med speedtest by Ookla. Resultatene viste at tilkoblingen fungerte tilfredsstillende, og at systemet klarte å levere stabil ned- og opplastingshastighet mellom 5 Mbps og 15 Mbps.

Etter den innledende testen ble oppsettet medbrakt til en hytte for å se hvordan løsningen fungerte i praksis over lengre tid og i et område uten fast internett. Her ble systemet brukt til å strømme TV via chromecast i 9 dager. Under denne perioden fungerte løsningen stabilt og uten avbrudd, noe som viste at 4G-HAT-modulen leverte tilstrekkelig båndbredde og pålitelighet også i et mer krevende bruksområde.

Etter at systemet ble tatt med tilbake fra hytten, oppstod det imidlertid problemer. Hotspot-funksjonen sluttet å virke, og ruteren klarte ikke lenger å opprette en stabil 4G-tilkobling. Dette førte til en periode med feilsøking og testing som varte i 1-2uker, hvor både konfigurasjonen av ruteren og APN-innstillingene ble gjennomgått og satt opp på nytt. Etter flere nye forsøk og oppstart helt fra bunnen igjen lyktes det å gjenopprette funksjonaliteten, og systemet fungerer nå igjen slik det skal. Problemene som vi hadde fant vi ikke grunnen til, så vi valgte å begynne helt på nytt med oppsettet.

Denne testperioden har gitt verdifull innsikt i både styrkene og svakhetene ved løsningen. Systemet viser seg å være robust under praktisk bruk, men også sensitivt for konfigurasjonsendringer, noe som krever grundig gjennomgang og rutiner for feilsøking.

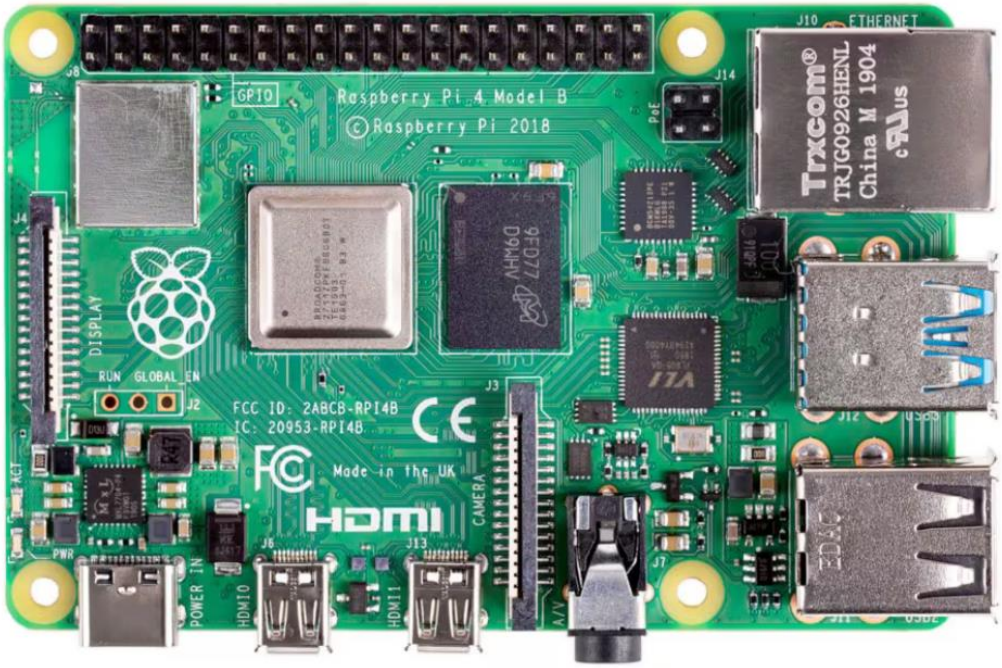
[18]



[18]

Figur 29: Bilde underfra av Waveshare 4G-HAT

Figur 30: Bilde ovenfra av Waveshare 4G-HAT



Figur 31: Raspberry pi 4B sett ovenfra

[19]

5.3 Testing av strømforbruk

For å estimere strømforbruket og levetiden til batteriet, gjennomførte vi målinger av strømforbruket i to hovedtilstander: under aktiv datainnhenting og -sending, samt i dvale (deepSleep). Det ble utført ti målinger hvor vi registrerte maksimal, minimal og stabilisert (gjennomsnittlig) strømbruk for hver test. Disse verdiene ble deretter brukt sammen med tidsmålingene for å beregne et estimert strømforbruk. Les mer om dette på 4.1.1 Batteri.

Basert på resultatene fant vi at det gjennomsnittlige strømforbruket under aktiv bruk var omtrent 0,35 A, med variasjoner mellom 0,25 A og 0,67 A. Gjennomsnittlig varighet for én syklus var 87,1 sekunder.

Vi kan også observere fra stabiliseringsverdien at det ikke oppstår uventede problemer under kjøringen av koden. Strømforbruket holder seg jevnt og konsistent gjennom alle testene, noe som tyder på at systemet oppfører seg forutsigbart og pålitelig under drift. Dette gir en viktig indikasjon på at både programvaren og maskinvaren fungerer som forventet, og at det ikke forekommer unødvendige strømforbrukstopper eller avvik som kan redusere batterilevetiden eller skape driftsproblemer over tid.

Under dvale målte vi et vi i området mellom 0.0084A til 0.0091A som tilsvarer 8.4mA og 9.1mA.

Test Nr.	Tid(sek)	Maks (A)	Min (A)	Stabilisert (A)
1	106	0.50	0.25	0.35
2	84	0.66	0.27	0.35
3	86	0.58	0.26	0.35
4	88	0.60	0.26	0.35
5	91	0.58	0.31	0.35
6	87	0.64	0.31	0.35
7	86	0.67	0.30	0.35
8	83	0.67	0.30	0.35
9	85	0.64	0.30	0.35
10	85	0.67	0.29	0.35
Gjensomsnitt	87.1			
deepSleep Tilstand		0.0091	0.0084	

Tabell 4: Resultat av måling på strømforbruk

5.4 Testing av komplett oppsett

Den siste testen vi gjennomførte ble gjort med et komplett og fullt integrert oppsett, hvor alle komponentene var koblet sammen og systemet opererte som planlagt. Prosessen startet med at ESP-enheten automatisk aktiverte både seg selv og alle tilkoblede komponenter. Deretter ble måledata innhentet, WiFi-tilkobling ble etablert (via en Raspberry Pi), og dataene ble sendt til ThingSpeak-plattformen for sanntidsvisualisering. Etter vellykket dataoverføring gikk ESP-enheten automatisk inn i deepSleep-modus for å redusere strømforbruket. Hele prosessen ble deretter gjentatt i en kontinuerlig syklus.

Under testingen satte vi deepSleep-tiden til 15 sekunder for å kunne observere og verifisere funksjonaliteten og dataflyten i sanntid. Systemet fungerte som forventet med hensyn til kodeutførelse, datainnsamling og kommunikasjon mot ThingSpeak.

Den største utfordringen vi støtte på i dette testløpet var relatert til vektsensoren (HX711). Etter hver omstart av ESP-enheten returnerte sensoren inkonsistente og feilaktige målinger. Det viste seg at sensoren nullstilles ved hvert strømavbrudd, og dermed mister eventuelle referanseverdier, noe som resulterte i unøyaktige data til ThingSpeak.

Vi forsøkte flere tiltak for å løse problemet:

- Lagring av referansevekt i EEPROM (ESP-enhetens interne flashminne), slik at systemet kunne gjenopprette tidligere verdi etter oppstart. Dette ga imidlertid ikke ønsket stabilitet.
- Separat strømtilførsel til HX711 fra batteriet for å unngå at sensoren slås av sammen med ESP-en. Heller ikke dette løste problemet, sensoren leverte fortsatt ugyldige data etter oppstart.

Til slutt fant vi en løsning på det, etter mye testing av fant vi ut at vektsensoren trenger mer tid til å stabilisere seg etter oppstart, og at enkeltmålinger tidlig i prosessen ikke er til å stole på. Løsningen ble derfor å endre koden slik at systemet tar 30 målinger og deretter sender riktig verdi videre. Etter denne justeringen fungerte vektsensoren stabilt, og vi oppnådde nøyaktige og pålitelige målinger som kunne overføres til ThingSpeak.

Et annet praktisk hinder vi møtte underveis var at opplasting av ny kode til ESP-enheten ble tidkrevende. Når ESP benytter deepSleep-modus må enheten fysisk tilbakestilles eller settes i "flash mode" for hver ny opplasting. Dette førte til ekstra arbeid mellom iterasjoner og gjorde utviklingsprosessen mer tungvint enn ønskelig. Vi løste dette ved å koble av ledningen som tillater ESP-en å gå i dvale for å deretter koble til ledningen igjen.

6 Diskusjon

Det ble tidlig klart at vi stod overfor to store utfordringer: området der bikuben skal stå har verken WiFi-dekning eller tilkobling til strømnettet. Hadde vi hatt tilgang til WiFi, så kunne vi enkelt brukt et batteridrevet system med lavt strømforbruk. Og med tilgang til strøm, ville det vært uproblematisk å sette opp en trådløs ruter med WiFi. Det var altså kombinasjonen av manglende nett og strøm som gjorde situasjonen krevende.

Prosjektet har i stor grad fulgt den opprinnelige fremdriftsplanen, men tidlig gjorde vi bevisste valg om å utvide oppgaven og dermed øke kompleksiteten.

Den opprinnelige oppgaven virket relativt enkel - kun å måle temperatur og vekt, og vise data på et fysisk display montert på bikuben. Vi vurderte at dette ikke ga oss tilstrekkelig faglig utfordring eller god nok verdi for oppdragsgiver. Vi valgte derfor å utvide prosjektet til også å inkludere lagring og sending av data til internett via 4G-forbindelse, slik at dataene kunne overvåkes eksternt. Dette viste seg å bli den mest omfattende delen av prosjektet, og vi brukte mye tid på å utforske ulike løsninger. Vi la ned mye arbeid i planleggingen av hvordan dataene skulle sendes og vises, og vi brukte mye tid på feilsøking. Etter mange forsøk og mye feilsøking fant vi til slutt en løsning som fungerte godt: å bruke en Raspberry Pi som ruter og sende data via denne.

Selv om vi ikke utarbeidet en formell risikoanalyse i forprosjektet, ble vi tidlig bevisste på mulige utfordringer som strømforbruk, ustabil kommunikasjon og maskinvare kompatibilitet. Flere av disse viste seg å være reelle problemstillinger, særlig i forsøkene med LilyGo T-Call-modulen. Her støtte vi på problemer med IP-konfigurasjon, som gjorde det vanskelig å sende data. Vi måtte til slutt forkaste denne løsningen til fordel for et mer stabilt, men mer strømkrevende oppsett med Raspberry Pi og 4G-HAT.

I ettertid ser vi at oppgaveutvidelsen både krevde mer arbeid og skapte flere muligheter. Hadde vi holdt oss til den opprinnelige, enklere oppgaven, ville systemet vært lettere å ferdigstille, men vi ville gått glipp av verdifulle erfaringer innen systemdesign, feilsøking og kommunikasjonsteknologi. Prosjektet har derfor utviklet seg fra et enkelt lokalt sensorsystem til en delvis autonom IoT-løsning for birøkt i felt, noe vi ser på som en oppgradering.

7 Konklusjon

Valg av komponenter for vekt- og temperaturmåling ble gjort tidlig i prosessen. Logikken bak oppsettet ble raskt utarbeidet, og vi laget et script til en mikrokontroller hvor vi testet begge sensorene. Testene ga gode og stabile målinger for både vekt og temperatur.

Vi bestemte oss tidlig for å sende måledata over internett og det ble utarbeidet en løsning for dette.

Mot slutten av prosjektet, under testing av det komplette oppsettet, oppdaget vi at systemet bruker mer strøm enn forventet i deepSleep-modus. Vi hadde opprinnelig estimert et strømforbruk på omtrent 1 mA i denne tilstanden. Det viste seg imidlertid at virkningsgraden til spenningsomformeren ble betydelig redusert ved lave strømmer, noe som førte til et jevnt strømforbruk på rundt 8 mA i dvale.

Denne økte strømbruken har en merkbar innvirkning på systemets totale driftstid, og reduserer den med ca. 180 dager sammenlignet med det opprinnelige estimatet. Likevel vurderer vi at kravet til driftstid er oppfylt, ettersom systemet fortsatt opererer i over 32 dager som er godt innenfor den spesifiserte kravet på minst 14 dager.

For å forlenge driftstiden ytterligere, kan det vurderes å benytte en spenningsomformer med høyere effektivitet ved lave strømforbruk. Et annet forbedringstiltak vi ønsket å implementere, dersom vi hadde hatt mer tid, er et tilkoblet solcellepanel som kontinuerlig kan lade batteriet og dermed øke systemets levetid betraktelig.

Vi hadde opprinnelig planer om å gjøre systemet vanntett og klart for utendørs bruk. Likevel valgte vi å prioritere tiden vår på å løse utfordringene knyttet til sending av data og vektmålinger, som vi arbeidet intensivt med helt fram til innleveringsfristen. Vi utsatte derfor å gjøre systemet klart for utendørsbruk, både fordi det ville tatt verdifull tid fra feilsøkingen, og fordi vi ikke ville bruke tid på å gjøre prototypen vanntett, men vente til vi hadde den mer kompakte løsningen med kretskort. Det ble designet og bestilt, men vi rakk ikke å motta det i posten før fristen på oppgaven. Vanntetting kunne vært løst ved å plassere komponentene, som vist i blokkskjemaet, i en vanntett kontrollboks med tilstrekkelig ventilasjon.

Systemet kan enkelt utvides til å overvåke flere bikuber med kun litt konfigurasjoner. For å legge til en ny bikube trenger man bare et batteri, nødvendige sensorer og en ESP8266.

Denne enheten kan benytte WiFi-nettverket som allerede er satt opp ruteren i det eksisterende systemet. Så lenge den nye enheten befinner seg innenfor rekkevidde av dette nettverket, kan den hente måledata og sende dem til samme database som den første enheten. På denne måten kan flere bikuber overvåkes effektivt uten behov for flere komplette systemer.

Referanser

Bibliografi

[1] P. S. O. honningbie, «snl.no,» Store norske leksikon, 23 4 2025. [Internett]. Available: https://snl.no/honningbie. [Funnet 15 5 2025].

[2] BROODMINDER, «BROODMINDER,» BROODMINDER, [Internett]. Available: https://broodminder.com/. [Funnet 19 05 2025].

[3] B. Norge, «Beehivemonitoring Norge,» Beehivemonitoring Norge, [Internett]. Available: https://beehivemonitoring.no/. [Funnet 19 05 2025].

[4] Heltec Automation, «heltec,» Heltec, 2023. [Internett]. Available: https://heltec.org/project/wifi-lora-32-v3/. [Funnet 10 5 2025].

[5] Umw-ic, «umw-ic,» 2022. [Internett]. Available: https://www.umw-ic.com/en/product_details/UMW+LM2596S-ADJ. [Funnet 19 05 2025].

[6] Velleman, «Velleman Group nv,» Velleman, 2025. [Internett]. Available: https://cdn.velleman.eu/downloads/29/vma406_a4v02.pdf. [Funnet 2025].

[7] ALLDATASHEET.COM, «ALLDATASHEET.COM,» [Internett]. Available: https://www.alldatasheet.com/datasheet-pdf/pdf/1132995/ESPRESSIF/ESP8266.html. [Funnet 19 05 2025].

[8] wemos.cc., «WEMOS,» WEMOS, [Internett]. Available: https://www.wemos.cc/en/latest/d1/d1_mini.html. [Funnet 19 05 2025].

[9] L. Shenzhen TuoZhanHong Technology Co., «AliExpress,» Shenzhen TuoZhanHong Technology Co., LTD, [Internett]. Available: https://www.aliexpress.com/item/1005004910195132.html?src=google&pdp_npi=4%40dis!EUR!2.19!2.19!!!!!%40!12000038023397749!ppc!!!&src=google&albch=shopping&acnt=298-731-3000&isdl=y&slnk=&plac=&mtctp=&albbt=Google_7_shopping&aff_platform=google&aff_short_key. [Funnet 18 05 2025].

[10] «electropeak,» [Internett]. Available: https://electropeak.com/learn/interfacing-yzc-161b-weighing-load-cell-sensor-with-arduino/. [Funnet 19. mai 2025].

[11] «Amazon,» [Internett]. Available: https://www.amazon.com/YZC-161B-Scale-Sensor-Human-Weighing/dp/B01NAN6HIU. [Funnet 19. mai 2025].

[12] patrick3345, «Thingiverse,» 04 11 2017. [Internett]. Available: https://www.thingiverse.com/thing:2624188. [Funnet 19 05 2025].

[13] I. Lukk, «Circuitjournal,» [Internett]. Available: https://circuitjournal.com/50kg-load-cells-with-HX711. [Funnet 19 05 2025].

[14] ALLDATASHEET.COM, «ALLDATASHEET.COM,» [Internett]. Available: https://www.alldatasheet.com/datasheet-pdf/pdf/1132222/AVIA/HX711.html. [Funnet 19 05 2025].

[15] DIYMore Direct Store, «AliExpress,» Shenzhen Rongbo Jiachuang Technology Co., Ltd., [Internett]. Available: https://www.aliexpress.com/item/1005008130234531.html?spm=a2g0o.productlist.main.13.3f6874a6cl4BEM&algo_pvid=14ea47cc-655a-45f2-ba52-3ae4c83862a5&algo_exp_id=14ea47cc-655a-45f2-ba52-3ae4c83862a5-12&pdp_ext_f=%7B%22order%22%3A%227%22%2C%22eval%22%3A%221%22. [Funnet 16 5 2025].

[16] ALLDATASHEET.COM, «ALLDATASHEET.COM,» [Internett]. Available: https://www.alldatasheet.com/datasheet-pdf/pdf/58557/DALLAS/DS18B20.html. [Funnet 19 05 2025].

[17] Raspberry Pi (Trading) Ltd., «Raspberrypi.com,» Raspberry Pi, 2024. [Internett]. Available: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf. [Funnet 2025].

[18] Waveshare International Limited, «Waveshare,» Waveshare, [Internett]. Available: https://www.waveshare.com/sim7600e-lte-cat-1-hat.htm?amazon. [Funnet 10 05 2025].

[19] Kjell&company, «Kjell&company,» Raspberry pi, 2025. [Internett]. Available: https://www.kjell.com/no/produkter/data/ettkortsdatamaskin/raspberry-pi-4-model-b-ettkortsdatamaskin-8-gb-ram-p88159. [Funnet 12 5 2025].

[20] cisco, «cisco,» cisco, [Internett]. Available: https://www.cisco.com/c/en/us/tech/wan/point-to-point-protocol-ppp/index.html. [Funnet 12 5 2025].

[21] U. Zieniüté, «nordvpn,» nordvpn, 25 9 2023. [Internett]. Available: https://nordvpn.com/no/blog/hva-er-dhcp/. [Funnet 12 5 2025].

[22] L. Klusaité, «Nordvpn,» nordvpn, 29 1 2022. [Internett]. Available: https://nordvpn.com/no/blog/hva-er-nat/. [Funnet 12 5 2025].

[23] Nord Security, «nordvpn,» Nord Security, 2025. [Internett]. Available: https://nordvpn.com/no/cybersecurity/glossary/ip-forwarding/. [Funnet 12 5 2025].

[24] one, «one,» one, 2025. [Internett]. Available: https://www.one.com/no/domener/hva-er-dns. [Funnet 12 5 2025].

[25] HiB, UiB, NHH, UiO og Nasjonalbiblioteket, «Søk og Skriv,» 12 12 2014. [Internett]. Available: http://sokogskriv.no/. [Funnet 12 12 2014].

[26] E. Importøren, «Elektro Importøren,» [Internett]. Available: https://www.elektroimportoren.no/?Article=94402&Referer=googleshopping&gad_source=1&gad_campaignid=21154348948&gbraid=0AAAAAD2QI3QHloDbkhWAbwO71gZ-ljDAE&gclid=CjwKCAjwravBBhBjEiwAIr30VGyTVmylXuHpTc41wJ_XeE3QVRgCKKKHiJd-J4gWlHYCwoGPMk-aDhoCBa4QAvD_BwE&gcls. [Funnet 19 05 2025].

[27] C. OpenAi, «(kodehjelp og rettskriving - ikke innhold),» OpenAi, [Internett]. Available: https://chatgpt.com/. [Funnet 20 05 2025].

[28] Tim, «core-electronics,» Core electronics, [Internett]. Available: https://core-electronics.com.au/guides/raspberry-pi/raspberry-pi-4g-gps-hat/. [Funnet 3 2025].

Appendiks A Forkortelser og ordforklaringer

Begrep	Forklaring
ADC	Analog-til-digital konverter som gjør om analoge signaler til digitale verdier.
API	Grensesnitt som lar ulike programmer kommunisere ved hjelp av regler og funksjoner.
APN	Tilgangspunktnavn brukt av SIM-operatøren for å koble til 4G-nett. Varierer mellom operatører.
DHCP	Protokoll som automatisk tildeler IP-adresser og nettverksparametere til klienter i nettverket.
DNS	Tjeneste som oversetter domenenavn til IP-adresser, som en internett-telefonbok.
deepSleep	Strømsparingsmodus for mikrokontrollere der mest funksjonalitet slås av og vekkes ved tid/hendelse.
dnsmasq	Kombinert DNS- og DHCP-tjener brukt i små nettverk for enkel adressetildeling og navneoppslag.
EEPROM	Ikke-flyktig minne som beholder data etter at strømmen er slått av.
hostapd	Programvare som gjør en Linux-maskin til et Wi-Fi-hotspot.
IP-forwarding	Gjør det mulig for en maskin å videresende datapakker mellom nettverksgrensesnitt, som en ruter.
iptables	Linux-verktøy for å konfigurere brannmurregler og kontrollere nettverkstrafikk.
MBIM-protokoll	Protokoll som forenkler kommunikasjonen mellom datamaskin og mobilt bredbåndsmodem. Erstatter eldre metoder som AT-kommandoer og PPP.
NAT	Metode som gjør at flere enheter på et lokalt nettverk kan dele én offentlig IP-adresse via en ruter.
PPP	Punkt-til-punkt-protokoll brukt for kommunikasjon mellom to enheter, f.eks. datamaskin og internettleverandør.
SSID	Navnet på et trådløst nettverk som vises når du søker etter Wi-Fi.
systemd	Systemverktøy i Linux som håndterer oppstart, tjenester og prosesser.

Appendiks B Prosjektledelse og styring

B.1 Prosjektorganisasjon

Vi ble som regel raskt enige om hvordan oppgavene skulle fordeles, og tok hensyn til hverandres styrker slik at alle fikk jobbe med noe de følte seg komfortable med. God kommunikasjon var avgjørende, spesielt siden mange deler av prosjektet måtte fungere sømløst sammen. For eksempel måtte programkoden fra de ulike gruppemedlemmene integreres korrekt, og vi hadde derfor jevnlig dialog for å sikre at alt fungerte som det skulle.

Vi hadde ingen formell prosjektleder, men det utviklet seg naturlig at hvert medlem tok ansvar for ulike deler av prosjektet:

- **Jacob Eide** hadde ansvar for mikrokontrolleren og programkode.
- **Erlend Matre** var ansvarlig for Raspberry Pi.
- **Erling Tellnes** hadde ansvar for strømtilførsel og kritiske komponenter som spenningsregulatorer og reléer.

Vi hadde alle god innsikt i hverandres arbeidsområder, noe som var avgjørende for at komponenter, kode og systemet som helhet skulle fungere sammen.

Appendiks C Bruksanvisning

Et krav i oppgaven var å lage en detaljert bruksanvisning. Denne er svært lang og omfattende. På grunn av dette valgte vi å lage et eget dokument der vi har inkludert bruksanvisningen. Denne filen blir lagt ved som et vedlegg.

Vi har også brukt github et brukerverktøy for samling av ulike type filer. Lenken er åpen for alle og kan finnes HER.

Appendiks D Bill Of Materials

D.1 Bill Of Materials

Komponent	Verdi / Beskrivelse	Antall	Estimert pris/stk (NOK)	Total (NOK)
Motstand	4.7kΩ	1	0	0
Motstand	8.2kΩ	1	0	0
Motstand	20kΩ	1	0	0
Motstand	1.2kΩ	2	0	0
Kondensator	0.1µF	3	0	0
Kondensator	10µF	2	0	0
Motstand	1kΩ	1	0	0
Spenningsregulator	LM2596 DC-DC	2	15	30
Terminalblokker	Inn/Ut til Pi	1	0	0
Terminalblokker	Inn/Ut til batteri	1	0	0
Temperaturføler	DS18B20	1	100	100
Pin-header	Female 2.54 mm, 1x6	1	0	0
Transistor	2N7000	1	0	0
Transistor	S9015	1	0	0
Mikrocontroller	WEMOS D1 Mini Pro v4	1	50	50
Vekt-sensorforsterker	HX711	1	15	15
Diode	1N4007W	1	0	0
Koblingskabler Hann–Hunn	Jumper wires	1	0	0
Kabel 1.5 mm^2 10 meter	Strømkabel	1	209	209
Blybatteri	12V 7.2Ah	1	210	210
Relémodul	Luxorparts, 1-kanals	1	100	100
Wemos D1 Mini Utviklingskort	ESP8266	1	150	150
SIM7600E-H 4G HAT	LTE Cat-4 + GNSS, for Raspberry Pi	1	720	720
Raspberry Pi 4 Model B – 8GB RAM	Ettkortsdatamaskin	1	1300	1300
Total Sum				2884