

3. Semesterprojekt

*Malte Egelykke Schwarz, Marcus Frederiksen, Nicolai Beltner
og Noor Jasper Rasmussen*

3. Semester

A

5

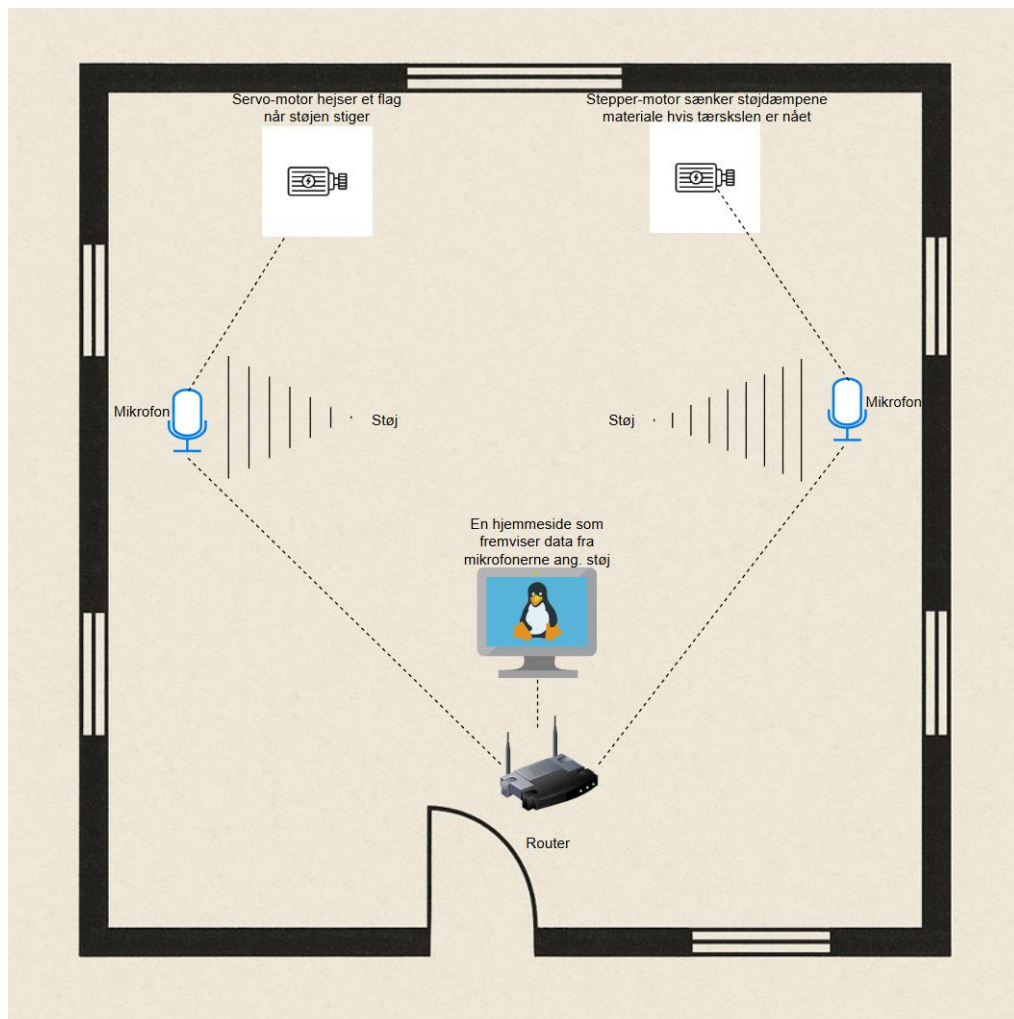
Støjforurening på arbejdspladsen

*Bo Hansen, Kevin Lindmark Holm, Tahseen Uddin og Charlie
Demasi*

79.284

Støjforurening på Arbejdspladsen





1. Resumé

I dette projekt har vi undersøgt, hvordan støjforurening i arbejdsområder kan bekæmpes og dermed nedsænkes. Denne problemstilling er et udbredt fænomen, som kan være komplekst at modvirke, pga. forskellige oplevelser af støjniveauet, begrænset kontrol eller sociale hensyn, men ikke desto mindre kan et negativt støjmiljø øge sundhedsmæssige komplikationer og nedsat arbejds effektivitet. Desangående kan en kompleks problemstilling virke uoverskuelig at komme til livs, derfor har vi valgt at skabe en løsning som kan hjælpe, ved at skabe opmærksomhed og overblik.

Som løsning har vi installeret to mikrofoner, som sender lydmålinger til en server, som fremviser dette data i realtid på en hjemmeside, så man kan følge med i lydniveauet. Dertil kommunikerer mikrofonerne med to aktuatorer, den ene løfter et flag ud og den anden nedsænker lyddæmpende materiale, hvis en prædefineret tærskel er blevet overskredet i et bestemt antal minutter. Derved bliver støjniveauet både fysisk og virtuelt fremvist igennem hjemmesiden, som gør det mere overskueligt og nemmere at blive opmærksom på, hvis det stiger.

Vores løsning nøjes ikke med at visualisere støjforureningen, men skaber opmærksomhed og bekæmper det i form af vores tekniske løsning, som resulterer i et fuldkomment projekt.

2. Indholdsfortegnelse

1.	Resumé.....	4
2.	Indholdsfortegnelse	5
3.	Indledning.....	7
4.	Problemformulering.....	8
5.	Indledende undersøgelse	10
5.1.	Ideudvikling.....	10
5.2.	Research (interview)	10
5.3.	Business Case	11
6.	Indlejrede Systemer	12
6.1.	Blokdiagrammer	12
6.2.	Diagrammer	13
6.3.	Beregninger og målinger.....	14
	Udregning for INMP441:	14
	Udregning for 28BYJ-48 & ULN2003A:	15
	Udregning for SG90-servo:	16
	Støjmålinger	17
6.4.	Teori af relevante elektroniske blokke	18
6.5.	Lovgivning	20
6.6.	Bæredygtighed.....	21
7.	Programmering	22
7.1.	Teori udvalgte elementer	22
	Profiling	22
	Asyncio vs. threading	23
	Pytest.....	24
7.2.	Kriterier for programkvalitet for udvalgt kode inden for de udvalgte teori elementer	25
8.	Netværk.....	28
8.1.	Router/switch-opsætning og interfaceforbindelser	28
8.2.	IP – plan	34
8.3.	Teori af relevante netværk og server-blokke	39
8.4.	GDPR.	41
9.	Linux	42
9.1.	Hardeningliste.....	49
10.	Krav, prioritet, accepttestprocedure og resultater	51
10.1.	Brugertest og resultater	64
11.	Praktisk projektplanlægning og -ledelse	65
11.1.	Scrum.....	66
12.	Konklusion.....	68
13.	Projektførløbet	69
14.	Perspektivering	70

15.	Litteraturliste.....	71
-----	----------------------	----

3. Indledning

Deltagelse i større forsamlinger er for mange ikke længere et valg, men en forpligtigelse. Stigningen i akademisk uddannede (SMVdanmark, 2022) skaber efterspørgsel på kontorpladser i de åbne kontorlandskaber. De åbne kontorer giver frihed for nem mobilisering af kontorpladser, men på bekostning af dem, som opholder sig i dem. Larm fra kollegaer, gæster, computere, kaffemaskiner osv. møder ingen støjmur, de kan slå ind mod. Dertil er ørerne skrøbelige, og ingen behandling findes, for støjinduceret høretab (Bjerg Groth, 2020).

En usynlig gene, som ikke melder sin ankomst i form af fysisk smerte, men psykisk. Livskvalitetsforringelser som hovedpine, migræne, stress, træthed, depression, er blot nogle af de konsekvenser, der følger med et usundt lydmiljø.

Problemet støj er en svær faktor at have med at gøre, for det er en nuanceret sag. Forbrugersamfundets afhængighed har fået mange til at vælge "Convenience over comfort". Tekøkkenet er i forlængelse af kontoret, frem for i separat rum. Det er nemmere at kunne give en besked på tværs af et rum, når der ikke er skillevægge mellem parterne (Perrin Jegen & Chevret, 2018).

Vores tekniske løsning vil skabe opmærksomhed på, hvornår et støjmiljø bliver usundt i arbejdsområder, hvor størstedelen af vågne timer bliver tilbragt; skoler, kontorer, hospitaler. Fundamentalt set skal der gøres opmærksom på støj, da det for mange er et grænseoverskridende samtaleemne.

Brug af fysiske markører, en støjindikator som fortæller, at støjniveauet er for højt. Aktuatorens som er styret af lydsensorer, skal give visuelle beskeder om, hvornår der er brug for en "lydudluftning" i et arbejdsmiljø. Grænseværdier skal bestemme, hvornår der er brug for understregning og sænkning af støjniveauet og støjabsorberende materiale skal nedsænkes fra loftshøjde, for at absorbere lyd.

4. Problemformulering

Hvad er problemet?

Problemet med støjforurening i indendørs miljøer er, at selv relativt lave støjniveauer kan forstyrre arbejdsopgaver og koncentrationen. Over halvdelen af kontoransatte er udsat for forstyrrende støj i mere end 1/4 af arbejdstiden (Arbejdstilsynet, u.d.). Dette kan påvirke produktiviteten og i værste fald have helbredsmæssige konsekvenser såsom stress (Arbejdsmiljø, godtarbejdsmiljø.dk, 2025).

Hvorfor er det et problem?

Støj på arbejdspladsen er et stigende problem i takt med udbredelsen af åbne kontorlandskaber (Castellum, 2024). Undersøgelser viser, at medarbejdere i åbne kontorlandskaber oplever markant flere afbrydelser og højere støjniveauer end tidligere. Da en stigning på blot 3 dB svarer til en fordobling af støjniveauet, kan selv små ændringer have stor indvirkning på arbejdsmiljøet. Støj fra samtaler, telefoner, kontormaskiner og mødelokaler trænger nemt ud i fællesarealer og skaber et konstant belastende lyd miljø.

For hvem er det et problem?

Støjforurening påvirker både medarbejderen og virksomheden. Medarbejderen kan opleve koncentrationsbesvær, stress, hovedpine og nedsat arbejdsglæde. For virksomheden kan det føre til faldende produktivitet, flere fejl i arbejdet og øget risiko for sygefravær. I kontormiljøer, hvor samarbejde og fokuseret arbejde skal gå hånd i hånd, bliver støj hurtigt en barriere for effektivt arbejde (Arbejdsmiljø, godtarbejdsmiljø.dk, 2025).

Hvordan vil man kunne løse problemet på teknisk vis?

Ved hjælp af vores tekniske løsning, kan man med data få kortlagt støj tendenserne, som giver mulighed for at tilføre mere støjdæpende materiale på arbejdspladsen, for at hjælpe akustikken. Yderligere bliver det anvist når støjniveauet er for højt via en anvisning fra en fysisk genstand i lokalet bestående af et flag, som bliver hejst, og derved skaber man opmærksomhed omkring støjniveauet. Alene opmærksomheden omkring støjniveauet vil i sig selv være med til at reducere lydniveauet (Erika Martínez Cantón, Gonzalez, & I. Ibarra-Zarate, 2019).

Forventet udbytte?

Udbyttet ved implementeringen af vores løsning, er at man kan kortlægge mængden af støjforurening. Denne kortlægning kan man bruge til at forbedre lyd miljøet på arbejdspladsen.

Hvis vi tager udgangspunkt i en case study, hvor netop dette var et fokus og flere støjdæpende metoder blev indført som følge af målinger af for meget støj på arbejdspladsen, så kom der et fald på mellem 5-10 dB i forhold til året før (Arbejdsmiljø, godtarbejdsmiljø.dk, 2024).

Vi forventer at nå samme fald af støjniveau med vores løsning.

5. Indledende undersøgelse

Vi opfangede rundt omkring på vores uddannelsesinstitution, at rigtig mange grupper ville udvikle en pille-dispenser. Vi blev hurtigt enige om, at det kunne være sjovt at udvikle noget andet. Snakken faldt på støj. Vi har alle oplevet at støj er til stor gene i undervisningssituationer og researchede derfor videre, om problemet med støj er et generelt problem i andre arbejdsmiljøer. Det tager ikke lang tid at finde artikler omhandlende støj, som har været et stigende problem, i takt med at åbne kontorlandskaber efterhånden er et valg som mange firmaer vælger at indrette sig i.

Den stigende mængde støj, som nu finder sted i mange arbejdsmiljøer, kan være med til at udvikle stress for den implicerede part og deraf medføre et tab af individuel arbejdsglæde for den enkelte og tabt effektivt og produktivitet for arbejdsgivere.

Vi blev enige om, at det ville være spændende at udvikle en løsning der kan være med til at sætte fokus på- og i bedste fald afhjælpe det stigende problem med støj i arbejdsmiljøet.

5.1. Ideudvikling

Vores idéudvikling foregik som en brainstorm i teamet, hvor vi tog udgangspunkt i arbejdsmiljøet. Et produkt, som hurtigt blev bragt på banen, var det allestedsværende øre (Soundear, u.d.), som man finder på mange skoler, læger, hospitaler og skadestuer, som visuelt viser, hvilket støjniveau der pt. er i det givne lokale. Et andet produkt vi snakkede om, var fuglen der "falder ned" når den måler at indeklimaet i det lokale den er hængt op i, er for dårligt (Birdie, u.d.).

Vi fandt på en idé, som er en kombination af de to produkter; en mikrofon, som måler lydniveauet i et lokale og som ved for højt lydniveau, aktiverer enten et flag, som rejser sig, eller lyddæmpende loftsplader, som sænkes.

5.2. Research (interview)

Vi tog udgangspunkt i et kort spørgeskema til Jane Bjerg Groth, som har skrevet en ph.d.-afhandling, der omhandler problematikken i forhold til helbredsmæssige påvirkninger fra støj (Bjerg Groth, 2020). Vi tog ydermere udgangspunkt i de artikler og andet research vi fandt frem til under projektets indledende fase. En stor del af vores research er derfor baseret på artikler og ph.d.-afhandling af Jane Bjerg Groth. Den viden blandet med vores egne idéer, er det hele idéen til vores løsning udspringer fra.

5.3. Business Case

I denne business case tages der udgangspunkt i at dokumentere støjforurening på arbejdspladsen og fastlægge om og hvornår der er behov for fokus på støjniveauet. Vi har taget udgangspunkt i en ph.d.-afhandling af Jane Bjerg Groth, som viser, at støj forårsager et midlertidigt høretab og akut tab af nerveceller i hørenerven (Bjerg Groth, 2020).

Begrundelser

Der har de sidste år været et stigende støjniveau, det kan f.eks. observeres på de danske hospitaler, hvor der er sket en stigning i støjniveauet fra 55 dB til 72 dB fra året 1960 til 2003, og yderligere konkluderes at det efterfølgende ikke er blevet bedre. Støjforurening er helbredsmæssigt skadeligt for mennesket. Det kan føre til forhøjet blodtryk, høreskader, hjertekarsygdomme, og medføre stress, angst og søvnforstyrrelser (Arbejds miljø, godtarbejds miljø.dk, 2024).

Forretningsmuligheder

Hvis vi vælger at gøre et minimum, så laver vi et system som kan måle dB til at fastlægge støjen.

Gør vi medium, kan dette data ses på en hjemmeside og læses tydeligt. Der vil blive aktiveret et flag og sænket støj dæmpende materiale fra loftet, hvis støjniveauet vurderes skadeligt.

Hvis vi gør maksimum, kan vores system alt det ovenstående. Derudover vil der blive sendt en rapport med data, som viser timestamp og andet brugbar information, samt vil systemet ved brug af frekvensmålinger kunne lokalisere, hvor i lokalet støjen kommer fra.

Forventet udbytte

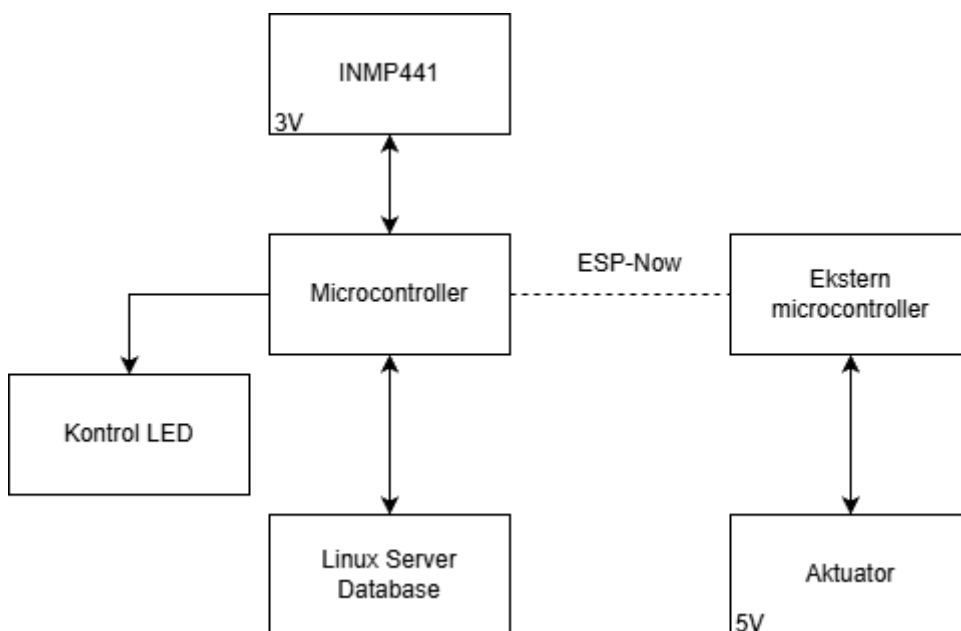
Der forventes at man ved implementering af systemet kan skabe overblik over støjniveauet i det pågældende lokale. Overblikket over et evt. for højt støjniveau kan man udnytte til at skabe løsninger, som kan forbedre lydniveauet og dermed arbejdsmiljøet på den givne arbejdsplads.

6. Indlejrede Systemer

I dette afsnit vil vi beskrive vores brug af indlejrede systemer faget, samt relevante diagrammer, beskrivelser og beregninger. Løsningen gør brug af en blanding mellem kendt læring fra tidligere semestre, læring fra 3. semester, samt et nyt emne, I2S.

6.1. Blokdiagrammer

Figur 6.1: Blokdiagram af vores løsning

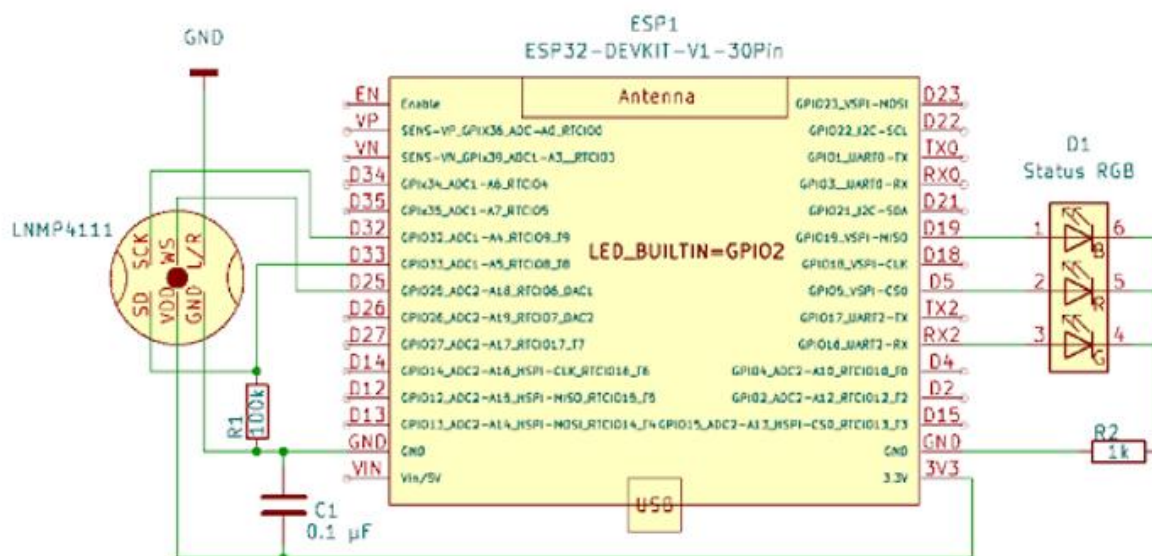


Vores kredsløb gør brug af ESP32-WROOM 32D microcontrollere. Lige netop disse microcontrollere er optimale at bruge, da de åbner op for flere måder at viderekommunikere til andre enheder. ESP32-WROOM 32D kommer med WiFi-antenne, Bluetooth, samt indbygget kommunikationsprotokol i ESP-NOW.

6.2. Diagrammer

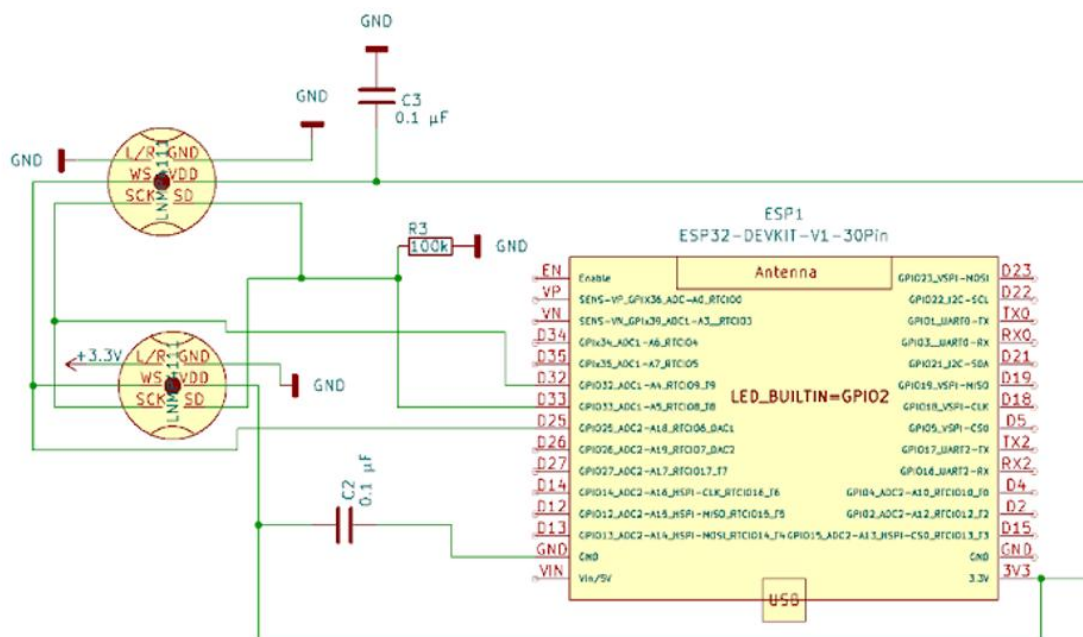
De elektriske diagrammer er udarbejdet i KiCad. Medtaget i rapporten, er den første version og den endelige version af kredsløbet med ESP32-WROOM-32D og INMP441-sensoren. De elektriske diagrammer for aktuatorkredsløbene findes i bilag 6.1 og 6.2.

Figur 6.2: Elektrisk diagram af første udkast



Det første udkast til vores projektløsning gør kun brug af en INMP441-sensor. Da Micropython ikke understøtter at køre I2S-modul med stereo (endnu), kan der kun gøres brug af monoton lyd. Vores første udkast til kredsløbet var derfor kun med en INMP441-sensor, forbundet med GPIO på ESP32 mikrokontrolleren.

Figur 6.3: Endeligt elektrisk diagram



Det endelige elektriske diagram gør brug af to INMP441-sensore, forbundet til de samme GPIO'er. Programmeringsmæssigt er begge mikrofoner sat til samme I2S-bus, clock og datalinjer. INMP441'erne giver stadig monoton lyd, dog fandt vi at de målinger i dB vi fik ud, var langt mere præcise når vi sammenlignede med en dB-måler fra vores telefoner.

Vores elektriske diagram har fulgt databladet for INMP441, hvor det beskrives, at ved brug af flere mikrofoner, skal der tilhørende sættes en 100kΩ pull-down modstand på, samt skal VCC være afkoblet til stel med 0,1 μF kondensator.

6.3. Beregninger og målinger

Udregning for INMP441:

INMP441 bliver forsynet med ESP32's 3,3 V output pin. Ifølge databladet (IvenSense, 2014) har INMP441 mikrofonen et strømforbrug på 2,2 - 2,5 mA. Vi kan bruge Ohms lov, da vi kender U og I, kan vi regne P ud med

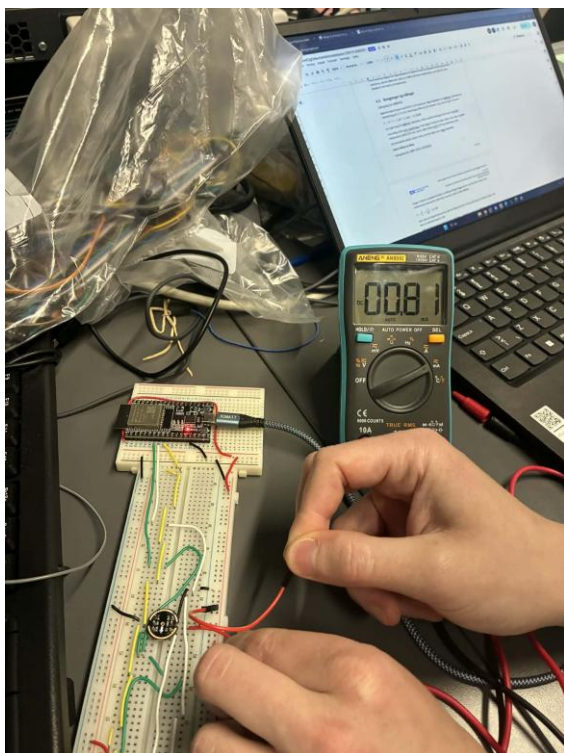
$$P = U * I = 3,3V * 2,5 \text{ mA} = 8,25 \text{ mW}.$$

Da vi gør brug af to INMP441-mikrofoner, vil det samlede forbrug for de to være 16.5 mW.

Ved måling af det reelle strømforbrug, er det vigtigt at huske at strøm måles i serie, ikke parallelt. Multimeteret sættes til DC mA, og der måles mellem ESP32 3.3 V og mikrofonens VDD.

Ved måling viser multimeteret en strøm på 0,81 mA, hvilket svarer til ca. den ampere vi regner med.

Figur 6.4: Måling af DC mA



Udregning for 28BYJ-48 & ULN2003A:

Steppermotorens datablad fortæller, at driftspændingen ligger på 5 V, samt har hver spole en DC-modstand på omkring 50 Ω . Med Ohms lov kan vi derfor finde effekten (Electronics).

$$P = \frac{U^2}{R} = \frac{5^2 V}{50 \Omega} = 0,5 W$$

Hver spole på stepper motoren har altså en effekt på 0.5 W. Ved normalt brug af to aktive spoler, vil vi have et strømforbrug på 1 W.

28BYJ-48 aktuatoren er styret af vores ULN2003A. ULN2003A består af 7 darlington-transistorer, som ifølge databladet er rated til 500 mA. Bruger vi Ohms lov igen, kan vi finde frem til, at vores steppermotors strømforbrug.

$$I = \frac{U}{R} = \frac{5 V}{50 \Omega} = 0,1 A = 100 mA$$

Hver spole har altså et strømforbrug på 100 mA, hvilket gør det sikkert at bruge ULN2003A sammen med 28BYJ-48.

Ifølge databladet (Instruments) har ULN2003A dog en intern mætning voltage på ca 1,2 V. Spolerne modtager derfor ikke helt 5V, men lavere.

$$U = 5,0 V - 1,2 V = 3,8 V$$

Derfra kan vi få den reelle spænding og effekt over hver spole.

$$I = \frac{3,8 V}{50 \Omega} = 76 mA$$

$$P = 3,8 V \cdot 0,076 A = 0,29 W$$

Hver spole har en reel effekt omkring 0,29 W, og ved brug af to spoler vil dette være 0,58 W ca.

Udregning for SG90-servo:

Databladet specificerer at servo-motoren har et typisk strømforbrug ved normal belastning på 250 mA, og ved maksimal belastning 800 mA (MG90S).

Ved normal belastning

$$P = 5 V \cdot 0,25 A = 1,25 W$$

Ved maksimal belastning

$$P = 5 V \cdot 0,7 A = 3,5 W$$

I demoen for dette projekt, vil servo motoren blive monteret med et klassisk fødselsdagsflag.

Belastningen af dette giver en effekt under den normale belastning.

De byggede kredsløb vil have en samlet effekt, som lyder

$$P_{total} = 0,0165 + 0,58 + 1,25 = 1,85 W$$

Energiforbruget i kWh vil være $kWh = \frac{(watt \cdot tid)}{1000} = \frac{(1,85 \cdot 8)}{1000} = 0,0148$.

Dette vil sige at ved en kWh pris på 1,37 kr, vil det koste 0,02 kr per dag at have IOT kredsløbene tændt.

Støjmålinger

Eftersom vores kredsløb gør stor brug af dele, som udleder elektrisk støj. Både internt i kredsløbet skabes der støj, samt er kredsløbet eksponeret overfor eksternt elektrisk og radio interferens.

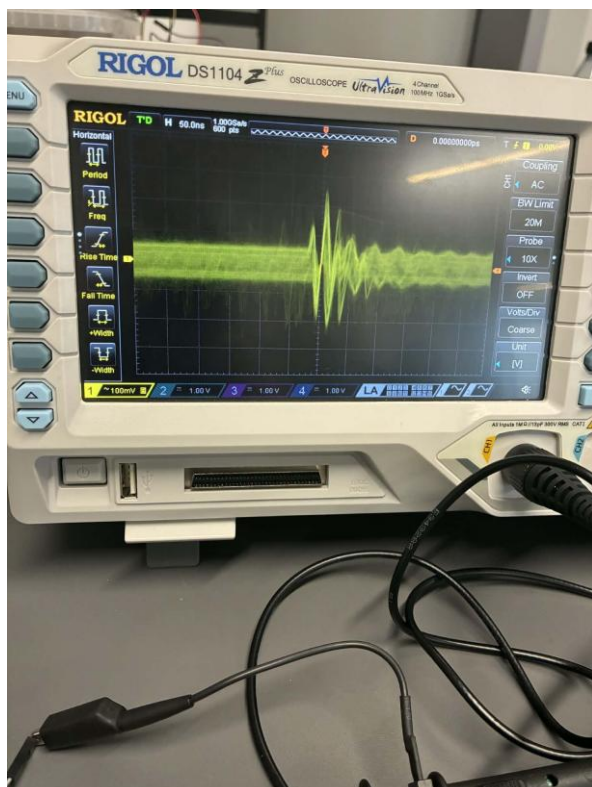
Vi har især oplevet radio interference når vi har gjort brug af ESP-NOW protokollen. Arbejdslokalet projektet har udformet sig i, var smækfyldt med routere som står og broadcaster, mini-pc'er som kører, og switche som bipper. Resultatet af alt dette elektriske støj var en ustabil forbindelse mellem INMP441 og aktuator kredsløbene.

For at mindske støj på selve INMP441 kredsløbet, er der sat to 0,1 μF kondensatorer fra hver forsyningspin til stel. Dette kaldes en afkobling, og det lader AC passere igennem, imens det spærrer for DC. På den måde kan vi få mindsket ledningsbåret støj.

En sample optagelse med oscilloskop bruger vi denne opstilling:

Channel 1, coupling AC, BW limit = 20 M, Probe = 10 X, Volts Coarse, Unit V, 100 mV, 50,0 ns

Figur 6.5: Billedet af Oscilloskop måling



Målingerne er lavet på forsyningen af INMP441 kredsløbet. Vi kan se et cirka toppunkt på omkring 200 mV over meget få nanosekunder. Det er svært at vurdere, men hvis man tæller tern, ligner det et udsving der varer omkring ~ 10 nanosekunder.

Denne slags støj kan komme fra mange kilder, da målingen er lavet med et fumlebræt. Vi vurderer derfor at støjen på kredsløbet ikke har betydning for den færdige løsning.

6.4. Teori af relevante elektroniske blokke

Der gøres brug af radioteknologier i form af WiFi og ESP-NOW. Vores opsætning fungerer som punkt-til-multipunkt, da vores mikrocontrollere kommunikerer videre til en central router, som derfra kommunikerer videre. De fungerer ikke selv som videreformidler af meddelelser. Mikrokontrolleren ESP32-WROOM 32D kommunikere kun på 2,4 GHz bølgelængden. Ved at bruge formelen ' $\lambda = c / f$ ' hvor c er lysets hastighed på 299.792.458 m/s. Kan vi regne ud hvor stor en Wi-Fi bølge vores ESP32'er udsender.

Figur 6.6: Bølgeformlen

$$\lambda = \frac{c}{f} = \frac{299.792.458 \text{ m/s}}{2,4 * 10^9 \text{ Hz}} = 0,125 \text{ m}$$

Det er relevant at vide for vores projekt, da interferens og støj vil kunne skabe problemer for andre enheder i løsningen. For at mindske dette, placeres mikrocontrollers antenne-modul, højt og frit fra materiale såsom betonvægge, som ville lave store dB tab for radioforbindelsen.

Vores analoge måling i projektet er "analog" støj. Vi måler lyd i dB fra vores INMP441, som har en indbygget analog til digital converter. Denne sensor gør brug af et I2S data interface, som er en synkron seriel kommunikations protokol. I2S er unik på den måde, at den er lavet specielt til lyd. I2S overfører digitale lydsamples op til 32 bit opløsning, hvorefter man kan programmere dette. Da det er en synkron seriel kommunikation, muliggør det at man kan lave stereo lydoptagelser, hvis man har flere mikrofoner forbundet til samme bit clock, og sætter det rigtige word select slot i sin programmering.

Vores I2S opsætning gør ikke brug af stereo lydformat; kredsløbet vi har lavet, gør brug af to INMP441-sensore programmeret til monoton lyd-input mode. Begge INMP441-mikrofoner forbindes til samme data pin, derfor skal data output linjen have en afkobling i form af en 100 kΩ pull-down modstand. Ydermere skal begge INMP441 afkobles fra VDD til GND med en 0,1 μF kondensator, dette er for at mindske interferens mellem de to mikrofoner, når de sidder relativt tæt.

Bloknavn: 28BYJ-48 Stepper Motor



Steppermotor bliver i vores projekt brugt til at hæve og sænke støjdæmpende materiale. Dette bliver gjort på data fra dB og vil hjælpe med akustikken inden i rummet hvor der bliver målt.

Bloknavn: Mini 9G Servo



Mini 9G Servo er en servomotor, som i projektet bliver sat sammen med en ESP32, og vil fungere som et flag, der tydeligt skal markere, hvornår der bliver larmet. Dette sker ved at motoren bliver aktiveret og hejser flaget.

Servoer styres via PWM, hvor pulsbredden definerer, hvor mange grader motoren skal dreje og holde.

Bloknavn: ESP32-WROOM-32D



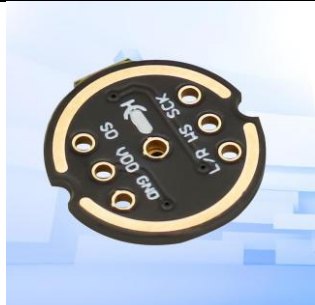
ESP32-WROOM-32D er en mikrokontroller, som er idéel for IOT-projekter. I vores projekt gør vi brug af ESP'ens mange GPIO forbindelser, samt dens kommunikationsmuligheder. Vi gør brug af dens WiFi-modul, for at kommunikere videre til vores netværk, og ud til andre ESP'er, som styrer vores aktuatorer.

Bloknavn: UNL2003



UNL2003 er et integreret kredsløb, hvori der sidder 7 darlington transistorer på, dette kredsløb bliver brugt som driver til at styrer vores steppermotor.

Bloknavn: INMP441



INMP441 er et omnidirektionelt mikrofonmodul, som fungerer med I2S, som er en synkron serial kommunikationsprotokol designet til at overføre lyddata.

6.5. Lovgivning

Da vores kredsløb forsynes fra stikkontakter, og derfor altid vil have adgang til 230V, er det vigtigt at alle komponenter er af høj kvalitet samt har markeringer, som angiver at forskellene regelsæt fra EU bliver overholdt. Komponenter af høj kvalitet sikrer ikke blot et velfungerende system, men også sikkerhedsmæssige tiltag. Devices som skal have høj uptime, som vores, skal ikke have en billig strømforsyning, da den let kan overophede. Markeringer som CE, RoHS og EMC er til for at sikre omgivelserne omkring opsætningen af løsningen.

Styklisten kan benyttes til tjekliste for at sikre at komponenter, eller samlede elektroniske komponenter, overholder lovsæt. CE-markeringer på alle komponenter skal sikre at vores løsning må handles frit inden for EØS. Ydermere skal RoHS-mærket sikre både os som producerer kredsløbet, samt nærmiljøet som produktet står i (Miljøstyrelsen, u.d.).

Eftersom vi anvender radio, er det vigtigt at vi tager retmæssige forholdsregler, når kredsløbet samles til en endelig løsning (Retsinformation, u.d.). Vi skal overholde EU's EMC-lovgivning, for at sikre at vores produkt ikke skaber interferens med andre enheder. Denne lovgivning er en ensrettet vej, da vi også ønsker at andre producenter overholder dette, for ikke at sende støj til vores kredsløb.

For testning af vores løsning hvad angår EMC-Lovgivning, skal den samlede løsning sendes til et lab, som er udstyret til at opfange elektrisk støjemission. Der findes private virksomheder i Danmark, der yder denne service, f.eks. Bolls i Stenløse (Bolls, u.d.).

6.6. Bæredygtighed

En hjørnesten for nutidens samfund er bæredygtighed. Ekskludering af en bæredygtig tænkning i en produktudvikling er direkte umoderne. Elektronik og IT er en meget forurenende branche; serverparker kræver enorme mængder energi for at opretholde driften, produktion af komponenter bruger tungmetaller, og logistik udleder store mængder kuldioxid.

Vores initiativer til at gøre vores produkt bæredygtigt, tager udgangspunkt i tre kerneområder indenfor elektronisk bæredygtighed.

For at sikre lang levetid på vores produkt, sikres det at alle komponenter på styklisten har CE og RoHS markeringer. Kvalitetskomponenter reducerer behovet for at lave udskiftninger af enten komponenter eller hele kredsløb.

Større komponenter, såsom aktuatorer, skal designmæssigt være konstrueret, så de er let tilgængelige. Plads omkring komponenter hjælper både temperaturmæssigt, samtidig med at det giver plads til udskiftning af enkelte komponenter. Hvis vi tager udgangspunkt i Mentech Eco's sigter vores produkt efter punkterne 1, 3, 4 og 5 (mentech). Komponenternes placering på kredsløbet hjælper genanvendelsen af tungmetaller i chips og komponenter ved afmontering. WEEE, mærket skal gøre opmærksom på korrekt bortskaffelse af produktet (EU).

På sigt kan der laves ændringer for at lave en hurtigere og mere bæredygtig løsning af vores produkt. Brug af mere effektive kodesprog som C og C++, som ikke kompileres imens det læses. Det giver mulighed for langt mindre processorer, som bruger færre bits, og derfor også lavere energiforbrug.

7. Programmering

<https://github.com/nico7634/3-semester-projekt.git>

7.1. Teori udvalgte elementer

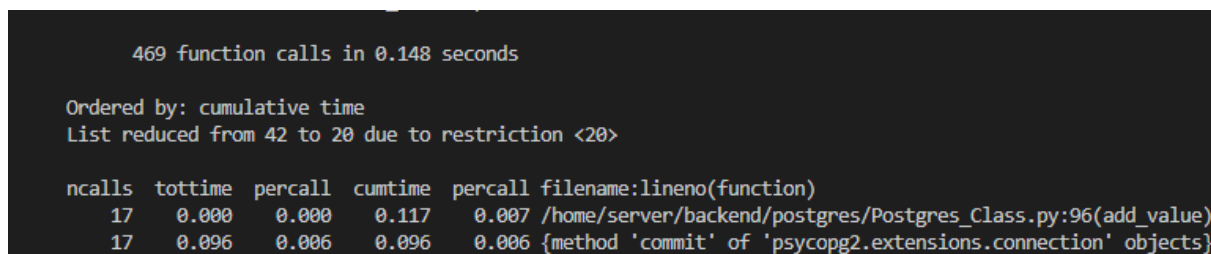
Vores løsning er bygget op af flere forskellige programmeringsrelevante elementer, som alle spiller en vigtig rolle i funktionaliteten af systemet. Dette afsnit vil beskrive, hvordan de respektive elementer er opsat og eksekvere deres funktioner.

Profiling

På Ubuntu-serveren er der opsat en hjemmeside, hvorpå en live opdatering af dB-niveauet kan observeres. For at opnå en liveopdatering af dB-niveauet har vi opsat en PostgreSQL database, som modtager data fra en ESP32 igennem WiFi-protokollen WebSocket. Derigennem kan vi fremvise de nyeste målinger fra mikrofonen, som er tilkoblet en ESP32.

For at teste funktionens effektivitet, har vi anvendt Python-biblioteket cProfile. En Profile er et statistisk overblik, som viser hvor hurtigt dele af et program har kørt, herefter kan dette analyseres ved hjælp af udvidelsen pstats (Python).

Figur 7.1: cProfile på websocket server



Billedet viser et eksempel efter en udført cProfile-test af vores WebSocket-server, og det tilhørende kode, som håndterer data tilsendt fra sensoren og lagrer det i vores PostgreSQL. Det fremviser cProfile-statistik, som hjælper med at skabe et overblik og identificere potentielle flaskehalse. Som udgangspunkt er det acceptabelt at køre 460 funktionskald på 0,148 sekunder, da selv helt små PC'er kan klare dette pres. Hertil skal det dog tilføjes, at der bliver kørt mere kode på serveren end den udvalgte sektion, men fordi koden stadig er relativ let, vil vores server (HP Elite Mini 800 G9) stadig kunne håndtere dette pres, eftersom den bl.a. har 16 GB RAM og 512 GB SSD (HP).

cProfile-statistikken er længere i dets naturlige format. Vi har valgt at fokusere på de to øverste, da resten af funktionerne er en del langsommere end de næste funktioner. Begge funktioner, og de efterfølgende, tilhører psycopg2. Den første statistik er et metodekald fra vores PostgreSQL-klasse og det andet er commit til databasen. Det fortæller os at WebSocket-programmet bruger mest tid på at indsætte data i vores PostgreSQL og committe dette, derfor ville det være oplagt at forbedre disse to,

hvis man ville optimere sin kode. Dette kan man opnå ved f.eks. at samle data fra sensoren i en dictionary, og så committe flere på en gang, i stedet for hver enkelt. Til sidst skal det også pointeres at testen kun er kørt med WebSocket-forbindelse til en ESP32 som sender sensordata, og at løsningen i praksis vil køre med to ESP32'ere. Testen giver, på trods af dette, stadig et overblik over effektivitet og funktionalitet.

Asyncio vs. threading

Vi har hovedsageligt gjort brug af Python-biblioteket Asyncio i vores kode, pga. logikken og funktionalitet i vores kode. Asyncio opererer på en enkelt tråd, og derfor bruger den i de fleste tilfælde færre ressourcer end threading, som opretter en ny tråd, der kører samtidigt, og det kan skabe problemer med Python's GIL (Global Interpreter Lock). GIL tillader kun en tråd af gangen at kontrollere Python-fortolkeren, derfor kan der kun være en tråd i det eksekverende stadie.

Asyncio eksekverer funktioner kaldet coroutines, som kan køre asynkront uden at blokere resten af programmet, som f.eks. funktioner som sleep-funktionen ville gøre. Dette gør den ved at køre en konstant løkke, som holder styr på, hvornår en coroutine kan blive eksekveret eller skal vente, uafhængigt af hinanden (Python R. , 2025). Derved holder den funktionaliteten indenfor en tråd, modsat threading. Yderligere arbejder netværksforbindelser godt sammen med Asyncio, fordi den effektivt kan håndtere samtidige opgaver ved hjælp af asynkrone programmeringsmønstre (Geeks, 2025).

På grund af disse elementer i Asyncio og threading har vi anvendt dem i forskellige situationer. Threading har vi dog kun anvendt i vores håndtering af forbindelsen til vores PostgreSQL database.

Figur 7.2: AutoClosePostgres-klasse

```
def __init__(self, db_conn:str, timeout_seconds:int):
    self.db = db_conn
    self.timeout = timeout_seconds
    self.last_use = 0
    self.lock = threading.Lock()
    self.active = False
    threading.Thread(target=self._watchdog, daemon=True).start()
```

Vi har anvendt threading her fordi det fungerer godt sammen med denne slags Watchdog-funktionalitet. Klassen skal køre udenom Asyncio, og være uafhængig af resten af programmet, derfor er det optimalt at adskille klassen fra resten af programmet. Dermed fortsætter klassen med at holde øje, selvom Asyncio's begivenhedsløkke stopper. Klassen sørger for at lukke forbindelsen til databasen. Derudover opretter Psycopg2 en synkron forbindelse til PostgreSQL-databasen, og derfor ville det

blokere Asyncio's begivenhedsløkke når den kom til klassen, dermed ville den stoppe alle andre coroutines i begivenhedsløkken (Lib, u.d.). I dette tilfælde er GIL ikke en udfordring, fordi den optager CPU i et meget lille og kort omfang, og derfor ikke bliver påvirket. GIL bliver først en udfordring, hvis threading-koden optager mere CPU og tid (Codecademy, u.d.).

Pytest

Der er undervejs i kodeudviklingen blevet anvendt Pytest, for at teste logikken i vores kode og at den agerer som vi forventer. Vi har især anvendt den til at teste koden, som interagerer med vores PostgreSQL.

Figur 7.3: Pytest med mock

```
5  @pytest.fixture
6  def mock_postgres():
7      # Lav PostgresClass objekt
8      pg = PostgresClass("user", "pass", "host", "5432", "db")
9
10     # Mock cursor og connection med Mock i stedet for MagicMock
11     pg.cursor = Mock()
12     pg.conn = Mock()
13     return pg
14
15  def test_add_value(mock_postgres):
16      # Kald add_value
17      mock_postgres.add_value("sensor1", 42)
18
19      # Tjek at execute blev kaldt med noget der indeholder 'INSERT'
20      assert "INSERT" in mock_postgres.cursor.execute.call_args[0][0]
21
22      # Tjek at commit blev kaldt
23      mock_postgres.conn.commit.assert_called_once()
```

På billedet kan ses et eksempel på, hvordan vi har testet en af de mest anvendte kodeblokke til PostgreSQL, det er essentielt at den agerer som vi forventer i praksis. For at teste dette har vi gjort brug af Pytest og Mock. Med Pytest har vi anvendt bibliotekets decorator kaldet fixture, for at opsætte en ny simuleret databaseforbindelse, som ved hver test funktion kaldes på ny. Derved kan testfunktionerne ikke påvirke hinanden, som der kunne være risiko for, hvis de havde den samme simulerede forbindelse (pytest.org, u.d.). For at teste funktionerne er de, som beskrevet tidligere, opsat med Mock som er et modul. Med Mock kan du oprette simulerede objekter, og dermed skabe et testmiljø, hvor kodelogik kan kontrolleres og testes. Dette kan ses på linje 11 og 12, her opsættes en simuleret cursor og forbindelse til den tilhørende Mock-klasse som tager udgangspunkt i vores egen

PostgreSQL-klasse, herefter kan vi køre prøvelser af de tilhørende metoder, og det kan ses på billedet (realpython.com, 2025).

7.2. Kriterier for programkvalitet for udvalgt kode inden for de udvalgte teorielementer

Figur 7.4: Initialiseringen af et PostgreSQL-forbindelse

```
class PostgresClass:
    """
    Initialiser informationer om den PostgreSQL man vil forbinde til.
    """
    def __init__(self, user:str, pswd:str, host:str, port:str, db:str):
        """
        Args:
            user, pswd, host, port, db: Information for at forbinde til databasen
        """
        self.user = user
        self.pswd = pswd
        self.host = host
        self.port = port
        self.db = db
```

Funktionalitet: Hvis klassen bliver initialiseret på den tiltænkte måde, virker den fejlfrit.

Læsbarhed: Med kendskab til klasser indenfor Python, er processen forståelig.

Dokumentation: Beskrivelsen er enkel, og kunne godt uddybes.

Overholdelse af standarder: Overordnet overholder det PEP 8, kun navngivningen af klassen er problematisk. Klassen er korrekt skrevet i CamelCase, men klassen burde beskrive, hvad dets samarbejde med PostgreSQL er. Så f.eks. PostgresConn ville være et mere passende navn.

Genanvendelighed: Dette opfylder klassen ikke. Klassen er skrevet specifikt til vores projekt, så man skal kende dokumentation for at anvende den optimalt. Dette kan f.eks. ses i metoden `create_sensor_table`, hvor man skal kende dokumentationen.

Vedligeholdelse: Der er god mulighed for at udvide klassen. I dette tilfælde ville man bare tilføje en metode med den ønskede funktionalitet.

Robusthed: Klassen i sig selv er ikke programmeret til at håndtere fejl, dette ansvar tilhører personen, som laver et objekt. Dette kræver igen at man kender koden på forhånd, hvilket kan føre til komplikationer.

Testbarhed: Er blevet testet med Pytest og Mock.

Effektivitet: Profiling viste at koden godt kunne optimeres.

Skalerbarhed: I tilfælde af opskalering, burde man tilføje de trin, som bliver diskuteret i profiling-afsnittet.

Sikkerhed: Sikkerheden er ikke optimal, koden er udelukkende til brug for ansatte med kendskab til funktionaliteten. Dette kan dog godt medføre fejl og problemer, hvis en ondsindet aktør får adgang eller en ansat får forkert input.

Figur 7.5: WebSocket server

```
20 postgres = PostgresClass(**postgres_params)
21 auto_postgres = AutoClosePostgres(postgres, timeout_seconds=30)
22
23 async def handler(websocket):
24     print("Client connected")
25     try:
26         async for json_message in websocket:
27             message = json.loads(json_message)
28             if not isinstance(message, dict):
29                 continue
30             db_value = message["current_dBSPL"]
31             print("Sensor data received:", db_value)
32             db = auto_postgres.use()
33             db.add_value(table_name="sensor1", db_value=db_value)
34     except websockets.exceptions.ConnectionClosed:
35         print("Client disconnected")
36
37 async def main():
38     # Start WebSocket server for sensorer
39     async with websockets.serve(handler, "192.168.10.3", 8765):
40         print("WebSocket server running on ip 192.168.10.3 port 8765")
41         await asyncio.Future()
42
43 asyncio.run(main())
```

Funktionalitet: Funktionen agerer som forventet, og producerer det tilsigtede resultat.

Læsbarhed: Koden opfylder ikke dette krav, der mangler forklaringer som doc-strings og parameter-input forklaring.

Dokumentation: I forlængelse af forklaring i læsbarhed, er dette krav heller ikke opfyldt.

Overholdelse af standarder: Hvis man ser bort fra dokumentationskravet i PEP 8, er kravet opfyldt. Funktioner er i snake case, og variabernes navn forklarer deres funktionalitet.

Genanvendelighed: Genanvendeligheden af koden er også minimal, det er sat op til at fungere med vores IP-adresser og forventede dataformat.

Vedligeholdelse: Koden kan godt ændres i tilfælde af opskalering el.

Robusthed: Koden fejlhåndterer ikke godt, den fanger kun WebSocket-forbindelsesfejl, og ville i tilfælde af andre fejl, crashe.

Testbarhed: Er blevet testet med Pytest og Mock.

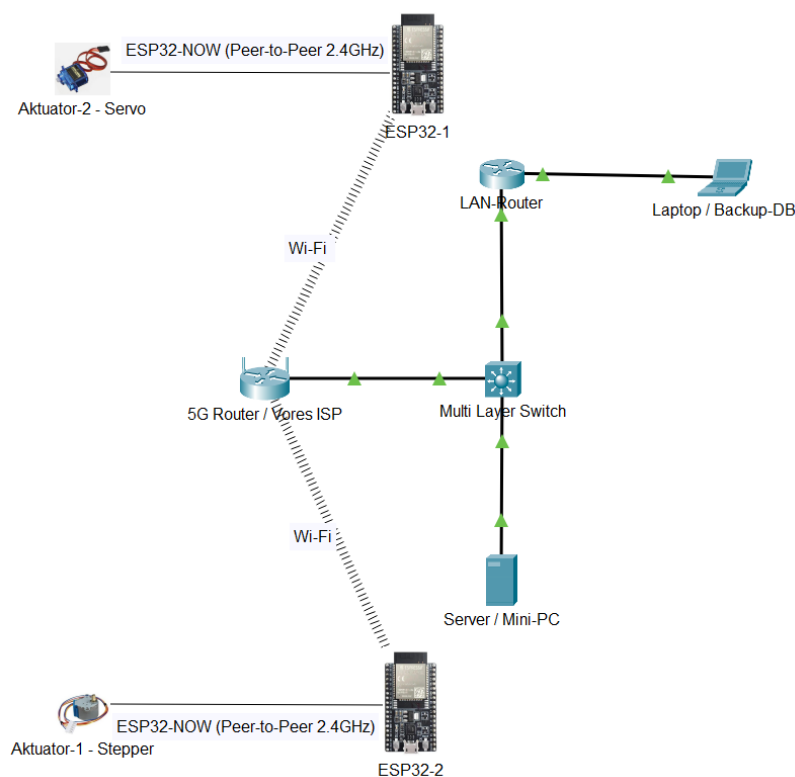
Effektivitet: Profiling viste at koden godt kunne optimeres.

Skalerbarhed: I tilfælde af opskalering, burde man tilføje de trin som bliver diskuteret i profiling-afsnittet.

Sikkerhed: Sikkerheden er dårlig. WebSocket mangler en API-nøgle så den kan autentificeres, og JSON-inputtet valideres ikke tilstrækkeligt. Derudover mangler server beskyttelse mod DDoS-angreb ved at begrænse data rate limit.

8. Netværk

Figur 8.1: Topologidiagram af vores netværksopsætning



8.1. Router/switch-opsætning og interfaceforbindelser

Dette afsnit beskriver opsætningen af interfaces på både MLS'en og Cisco1941-routeren. Formålet er at dokumentere, hvilke porte der anvendes, hvilket VLANs de tilhører, og hvilke IP-adresser og routing-funktioner der er konfigureret. Tabellen giver et samlet overblik over alle aktive forbindelser i netværket, herunder trunk linket mellem MLS og routeren, server- og uplink-portene VLAN 10 samt routerens subinterfaces, der indgår i OSPF-routingen.

Figur 8.2: Tabeloversigt af interface-forbindelser

Device	Interface	Destination	IP / Switchport / VLAN	ACL & Config
MLS (MultiLayer Switch)	VLAN 10 SVI	LAN 1 (Produktionsenheder)	192.168.10.1/24	L3 routing enabled
MLS	VLAN 30 SVI	Router 1941 (WAN-segment)	192.168.30.1/24	OSPF area 0

MLS	Gi1/0/1 (Trunk)	Router 1941 Gi0/0	Trunk: VLAN 10, 20, 30	Allowed VLANs: 10, 20, 30
MLS	Gi1/i0/2	Server	Access VLAN 10	
MLS	Gi1/0/11	5G Router (ISP)	Access VLAN 10	Default route peger mod 5G-router
MLS	WiFi via 5G-router	ESP32-enheder	Statiske LAN-IP 192.168.10.4/5	Default route peger mod 5G-router
Cisco 1941 Router	Gi0/0.30 (sub-interface)	VLAN 30 (Trunk til MLS)	192.168.30.2/24	OSPF area 0
Cisco 1941 Router	Gi0/0 (Fysisk)	Trunk til MLS	Encapsulation dot1q 30	Router-on-a-stick (kun VLAN 30)
Cisco 1941 Router	Gi0/1 (access)	Backup Laptop (VLAN 20)	192.168.20.1/24	OSPF area 0

Figur 8.3: MLS'ens interface-oversigt og routing-tabel

```
Switch#show ip interface brief | include Vlan10
Vlan10          192.168.10.1      YES manual up      up
Switch#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is 192.168.10.254 to network 0.0.0.0

C    192.168.30.0/24 is directly connected, Vlan30
C    192.168.10.0/24 is directly connected, Vlan10
S*   0.0.0.0/0 [1/0] via 192.168.10.254
Switch#
```

Billedet viser, at VLAN 10 SVI er konfigureret med IP-adressen 192.168.10.1 og er aktiv (up/up).

Samtidig ser vi routing-tabellen, hvor MLS'en har direkte forbindelser til VLAN 10 og 30 samt en default route mod 5G-routeren (192.168.10.254).

Default route mod 5G-router, som ses nederst på billedet (S* 0.0.0.0/0 [1/0] via 192.168.10.254) er konfigureret på MLS'en med følgende kommando:

ip route 0.0.0.0 0.0.0.0 192.168.10.254.

VLAN 10 og 30 er oprettet via MLS'en med følgende kommandoer:

```
conf t
vlan 10
name VLAN10
exit
vlan 30
name VLAN 30
exit
```

For at oprette default gateway for enheder i VLAN 10 har vi brugt følgende kommandoer:

```
interface Vlan10
ip address 192.168.10.1 255.255.255.0
no shutdown
exit
```

Da MLS'en fungerer som netværkets L3-kerne, er IP-routing aktiveret med følgende kommando:

```
ip routing
```

Figur 8.4: MLS – VLAN 30 SVI, trunk-konfiguration og OSPF-status

```
Switch#show ip interface brief | include Vlan30
Vlan30          192.168.30.1    YES manual up
Switch#show ip route | include 192.168.30.0
C    192.168.30.0/24 is directly connected, Vlan30
Switch#show interfaces trunk

Port      Mode          Encapsulation  Status        Native vlan
Gi1/0/1   on            802.1q         trunking      1

Port      Vlans allowed on trunk
Gi1/0/1   10,20,30

Port      Vlans allowed and active in management domain
Gi1/0/1   10,30

Port      Vlans in spanning tree forwarding state and not pruned
Gi1/0/1   10,30
Switch#show ip ospf interface Vlan30
Vlan30 is up, line protocol is up
  Internet Address 192.168.30.1/24, Area 0
  Process ID 1, Router ID 1.1.1.1, Network Type BROADCAST, Cost: 1
  Transmit Delay is 1 sec, State BDR, Priority 1
  Designated Router (ID) 2.2.2.2, Interface address 192.168.30.2
  Backup Designated router (ID) 1.1.1.1, Interface address 192.168.30.1
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:02
  Supports Link-local Signaling (LLS)
  Cisco NSF helper support enabled
  IETF NSF helper support enabled
  Index 3/3, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 1, maximum is 2
  Last flood scan time is 0 msec, maximum is 9 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 2.2.2.2 (Designated Router)
  Suppress hello for 0 neighbor(s)
Switch#
```

Billedet viser, at MLS'en har et aktivt SVI for VLAN 30 med IP-adressen 192.168.30.1/24, som fungerer som WAN-segmentet mellem switchen og Cisco 1941-routeren. Trunk-porten på Gi1/0/1 er konfigureret med 802.1Q og tillader VLAN 10, 20 og 30. Nederst ses, at OSPF kører korrekt på VLAN 30 i area 0, og at MLS'en har etableret naboskab med routeren (neighbor 2.2.2.2) via denne forbindelse.

Trunk-porten på Gi1/0/1 er konfigureret med følgende kommandoer:

```
interface GigabitEthernet1/0/1
switchport mode trunk
switchport trunk encapsulation dot1q
switchport trunk allowed vlan 10,20,30
```

SVI for VLAN 30 er konfigureret med følgende kommandoer:

```
interface Vlan30
ip address 192.168.30.1 255.255.255.0
no shutdown
```

OSPF i area 0 på MLS'en er sat op med:

```
router ospf 1
```

router-id 1.1.1.1

network 192.168.30.0 0.0.0.255 area 0

MLS'en danner herefter OSPF-naboskab med routeren, som har router-ID 2.2.2.2

Figur 8.5: MLS-Server-port (Gi1/0/2)

```
Switch#show interfaces GigabitEthernet1/0/2 switchport
Name: Gi1/0/2
Switchport: Enabled
Administrative Mode: static access
Operational Mode: static access
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: native
Negotiation of Trunking: Off
Access Mode VLAN: 10 (SERVER)
```

Billedet viser, at port Gi1/0/2 er konfigureret som statisk access-port i VLAN 10, hvilket bekræfter korrekt tilslutning af serveren til produktionsnetværket.

Konfigurationen af server-porten er udført med:

interface GigabitEthernet1/0/2

switchport mode access

switchport access vlan 10

no shutdown

Figur 8.6: MLS – 5G-Routerport (Gi1/0/11)

```
Switch#show interfaces GigabitEthernet1/0/11 switchport
Name: Gi1/0/11
Switchport: Enabled
Administrative Mode: static access
Operational Mode: static access
Administrative Trunking Encapsulation: negotiate
Operational Trunking Encapsulation: native
Negotiation of Trunking: Off
Access Mode VLAN: 10 (SERVER)
```

Billedet viser, at port Gi1/0/11 er konfigureret som statisk access-port i VLAN 10 og anvendes til tilslutning af 5G-routeren, som fungerer som netværkets default gateway mod internettet.

Konfigurationen af porten til 5G-routeren er udført med:

interface GigabitEthernet1/0/11

switchport mode access

switchport access vlan 10

no shutdown

Figur 8.7: Cisco 1941 – Trunk-port (Gi0/0) og VLAN 30 subinterface (Gi0/0.30)

```
Router#show running-config interface GigabitEthernet0/0.30
Building configuration...

Current configuration : 102 bytes
!
interface GigabitEthernet0/0.30
 encapsulation dot1Q 30
 ip address 192.168.30.2 255.255.255.0
end

Router#show running-config interface GigabitEthernet0/0
Building configuration...

Current configuration : 76 bytes
!
interface GigabitEthernet0/0
 no ip address
 duplex auto
 speed auto
end

Router#
```

Billedet viser, at GigabitEthernet0/0 fungerer som fysisk trunk-port mod MLS'en, og derfor ikke har en IP-adresse. VLAN 30 håndteres via subinterfacet Gi0/0.30, som er konfigureret med 802.1Q-tagging og IP-adressen 192.168.30.2/24. Dette interface udgør routerens WAN-segment mod MLS'en og bruges til OSPF-routing mellem enhederne.

Konfiguration er udført med følgende kommandoer:

```
interface GigabitEthernet0/0
no ip address
no shutdown
exit
interface GigabitEthernet0/0.30
 encapsulation dot1q 30
 ip address 192.168.30.2 255.255.255.0
no shutdown
exit
```

Figur 8.8: Cisco 1941 – Backup LAN interface (Gi0/1)

```
Router#show running-config interface GigabitEthernet0/1
Building configuration...

Current configuration : 100 bytes
!
interface GigabitEthernet0/1
 ip address 192.168.20.1 255.255.255.0
 duplex auto
 speed auto
end
Router#
```

Billedet viser, at GigabitEthernet0/1 fungerer som Layer 3-interface til backup-netværket, konfigureret med IP-adressen 192.168.20.1/24. Interfacet anvendes som gateway for backup-laptoppen og indgår i routerens OSPF-konfiguration.

Konfigurationen er udført med:

```
interface GigabitEthernet0/1
ip address 192.168.20.1 255.255.255.0
no shutdown
exit
```

8.2. IP – plan

Vores netværk er en VLAN-baseret opbygning, som giver en klar opdeling mellem funktionerne i vores system. Det giver en god mulighed for fleksibilitet og skalering. Vi bruger ét samlet netværk som består af tre logiske VLANs. De har hver deres egen funktion.

VLAN 10 (LAN 1)

VLAN 10 består af de enheder, som indsamler og behandler støjdata. Vores Multilayer switch (MLS) fungerer som kernen. VLAN 10 omfatter:

- MLS'ens VLAN 10 SVI (default gateway)
- Server – modtager og lagrer data
- ESP32'erne – kommunikerer via 5G-routeren
- Forbindelsen videre mod WAN-routeren

5G-routeren er kablet til MLS'en og fungerer som ISP. ESP32'erne får adgang til netværket via Wi-Fi, og sender efterfølgende data videre til vores server gennem MLS'en.

VLAN 20 (LAN 2)

På GigabitEthernet0/1-porten tilhørende vores 1941 Cisco router har vi vores backup-laptop. 1941-routeren klarer VLAN-skiftet fra trunk-linjen mellem MLS og routeren. Den giver backup-PC'en sit eget isolerede netværk, som er VLAN 20.

Backup-laptoppen bruger routing til at kommunikere med serveren i VLAN 10. Alle VLAN er isoleret fra hinanden på Layer 2 og al trafikken går gennem routeren. Ved hjælp af den segmentering opnår vi øget sikkerhed.

WAN – VLAN-baseret routing mellem MLS og router

1941-routeren er forbundet til MLS'en via en trunk-port. De bruger VLAN 30 til deres interne routing. VLAN 30 virker som WAN-segmentet. Det er her at MLS'en og routeren (1941) udveksler OSPF-routing-informationer. På den måde opnår vi en dynamisk og skalerbar routing-struktur og undgår manuelt konfigurerede static routes.

DHCP

Det eneste tidspunkt, hvor der bliver anvendt DHCP i dette projekt er, når vores egne bærbare forbindes til 5G routeren, dette sker kun, hvis vi skal SSH ind i serveren. Alt andet, som er forbundet på netværket, vi tænker her på vores server og ESP'er, modtager en fast IP-adresse, som følger vores IP-plan.

IP-PLAN

VLAN 10

Net ID: 192.168.10.0/24

Subnet Mask: 255.255.255.0

Broadcast: 192.168.10.255

Antal host: 255

Figur 8.9: Tabeloversigt over IP-Plan

Enhedstype	Antal enheder	IP-adresse område	Udvidelsesmuligheder	MAC-adresser
5G Router / Gateway	1	192.168.10.254		18:EE:86:49:4A:87
MLS	1	192.168.10.1		503d.e519.3800
Server	1	192.168.10.3		4B:EA::62:E8:92:FE
ESP32	2	192.168.10.4-5	6-14 kan anvendes hvis der vil tilføjes flere mikrofoner	stepmotor mac:\xc8.\x18\x16\xac\xc0 mikrofon 1: \xd4\x8a\xfc\x94\x88 servomotor mac:\xb4.\x12\x58\xaf\x44

VLAN 20 (Backup Lan)

Net ID: 192.168.20.0/24

Subnet Mask: 255.255.255.0

Broadcast: 192.168.20.255

Antal host: 255

Enhedstype	Antal enheder	IP-adresse område	Udvidelsesmuligheder	MAC-adresse
Router 1941 / Gateway	1	192.168.20.1		
Backup Laptop	1	192.168.20.50		A0:CE:C8:DC:D7:86

VLAN 30

Net ID: 192.168.30.0/24

Subnet Mask: 255.255.255.0

Broadcast: 192.168.30.255

Antal host: 255

Enhedstype	Antal enheder	IP-adresse område	Udvidelsesmuligheder	MAC-adresse
Lan 1 MLS	1	192.168.30.1		
Lan 2 Router	1	192.168.30.2		

Adressetabel:

Figur 8.10: Adressetabel

Device	Interface	IP-adresse
MLS (MultiLayer Switch)	VLAN 10 SVI	192.168.10.1
	VLAN 30 SVI	192.168.30.1

	Default route -> 5G	192.168.10.254 (next-hop)
Server	Gi1/0/2 (VLAN 10)	192.168.10.3
ESP32 Mikrofon 1	Wi-Fi (VLAN 10)	192.168.10.4
ESP32 Mikrofon 2	Wi-Fi (VLAN 10)	192.168.10.5
5G-router (ISP)	Gi1/0/11	192.168.10.254
Cisco 1941 Router	Gi0/0.30 (VLAN 30)	192.168.30.2
	Gi0/1 (Backup LAN)	192.168.20.1
Backup Laptop	Ethernet (VLAN 20)	192.168.20.3

TCPDump-Dokumentation

For at dokumentere TCP-trafik i sensornetværket kørte vi følgende kommando på serveren:

```
sudo tcpdump -i eno1 -nn tcp port 22
```

Det filtrerer trafikken, så vi kun får vist TCP-pakker på port 22. Det vil sige SSH-trafik. Vi forventer at se en aktiv TCP-session mellem serveren og en klient i VLAN 10.

Figur 8.11: Billede af TCPdump af SSH-forbindelse

```
server@agruppe:~$ sudo tcpdump -i eno1 -nn tcp port 22
[sudo] password for server:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eno1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
10:26:36.072104 IP 192.168.10.3.22 > 192.168.10.53.59944: Flags [P.], seq 3502800972:3502801160, ack 314884974, win 54
0, length 188
10:26:36.075138 IP 192.168.10.53.59944 > 192.168.10.3.22: Flags [P.], seq 1:37, ack 0, win 252, length 36
10:26:36.075243 IP 192.168.10.3.22 > 192.168.10.53.59944: Flags [P.], seq 188:224, ack 37, win 548, length 36
10:26:36.078148 IP 192.168.10.53.59944 > 192.168.10.3.22: Flags [.], ack 224, win 251, length 0
10:26:36.118851 IP 192.168.10.53.59944 > 192.168.10.3.22: Flags [P.], seq 37:73, ack 224, win 251, length 36
10:26:36.118994 IP 192.168.10.3.22 > 192.168.10.53.59944: Flags [P.], seq 224:260, ack 73, win 548, length 36
10:26:36.136943 IP 192.168.10.53.59944 > 192.168.10.3.22: Flags [P.], seq 73:109, ack 260, win 251, length 36
10:26:36.137241 IP 192.168.10.3.22 > 192.168.10.53.59944: Flags [P.], seq 260:296, ack 109, win 548, length 36
10:26:36.164875 IP 192.168.10.3.22 > 192.168.10.53.59944: Flags [P.], seq 296:1228, ack 109, win 548, length 932
10:26:36.167286 IP 192.168.10.53.59944 > 192.168.10.3.22: Flags [P.], seq 109:145, ack 296, win 250, length 36
10:26:36.167570 IP 192.168.10.3.22 > 192.168.10.53.59944: Flags [P.], seq 1228:1264, ack 145, win 548, length 36
10:26:36.167948 IP 192.168.10.53.59944 > 192.168.10.3.22: Flags [.], ack 1228, win 255, length 0
10:26:36.170669 IP 192.168.10.53.59944 > 192.168.10.3.22: Flags [.], ack 1264, win 255, length 0
10:26:36.186366 IP 192.168.10.53.59944 > 192.168.10.3.22: Flags [P.], seq 145:181, ack 1264, win 255, length 36
10:26:36.186642 IP 192.168.10.3.22 > 192.168.10.53.59944: Flags [P.], seq 1264:1300, ack 181, win 548, length 36
10:26:36.239934 IP 192.168.10.53.59944 > 192.168.10.3.22: Flags [P.], seq 181:217, ack 1300, win 255, length 36
10:26:36.240225 IP 192.168.10.3.22 > 192.168.10.53.59944: Flags [P.], seq 1300:1336, ack 217, win 548, length 36
10:26:36.268782 IP 192.168.10.53.59944 > 192.168.10.53.59944: Flags [P.], seq 1336:2364, ack 217, win 548, length 1028
10:26:36.272499 IP 192.168.10.53.59944 > 192.168.10.3.22: Flags [P.], seq 217:253, ack 1336, win 255, length 36
10:26:36.272795 IP 192.168.10.3.22 > 192.168.10.53.59944: Flags [P.], seq 2364:2400, ack 253, win 548, length 36
10:26:36.276078 IP 192.168.10.53.59944 > 192.168.10.3.22: Flags [.], ack 2400, win 251, length 0
10:26:36.276079 IP 192.168.10.53.59944 > 192.168.10.3.22: Flags [P.], seq 253:289, ack 2400, win 251, length 36
10:26:36.276389 IP 192.168.10.3.22 > 192.168.10.53.59944: Flags [P.], seq 2400:2436, ack 289, win 548, length 36
```

Vi kan se at der foregår en TCP-session mellem 192.168.10.3.22 (serverens SSH-port) og 192.168.10.53.59944 (en klient i VLAN 10, som har startet en SSH-forbindelse mod serveren). Klienten benytter en såkaldt ephemeral port (59944), som er normale, tilfældige porte for udgående TCP-forbindelser fra Windows- eller Linux-klienter.

Sekvens- og ack-numrene (seq & ack) stiger synkront på begge sider, hvilket viser en fuldt etableret og stabil TCP-forbindelse. Det samlede dump viser et godt samarbejde mellem server og klient og viser at der er fuld funktionalitet i netværkets primære segment.

For at dokumentere kommunikationen mellem VLAN 20 (Backup Lan) og VLAN 10 (Produktions-LAN) kørte vi TCPDump på serveren i VLAN 10 med følgende kommando:

```
sudo tcpdump -i eno1 -nn -icmp
```

Figur 8.12: Billede af TCPdump m. ICMP

```
server@5agruppe:~$ sudo tcpdump -i eno1 -nn icmp
[sudo] password for server:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eno1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
08:52:32.672175 IP 192.168.20.3 > 192.168.10.3: ICMP echo request, id 1, seq 386, length 40
08:52:32.672270 IP 192.168.10.3 > 192.168.20.3: ICMP echo reply, id 1, seq 386, length 40
08:52:32.674384 IP 192.168.10.1 > 192.168.10.3: ICMP redirect 192.168.20.3 to net 192.168.10.2, length 36
08:52:33.693744 IP 192.168.20.3 > 192.168.10.3: ICMP echo request, id 1, seq 387, length 40
08:52:33.693799 IP 192.168.10.3 > 192.168.20.3: ICMP echo reply, id 1, seq 387, length 40
08:52:33.695279 IP 192.168.10.1 > 192.168.10.3: ICMP redirect 192.168.20.3 to net 192.168.10.2, length 36
08:52:34.707200 IP 192.168.20.3 > 192.168.10.3: ICMP echo request, id 1, seq 388, length 40
08:52:34.707256 IP 192.168.10.3 > 192.168.20.3: ICMP echo reply, id 1, seq 388, length 40
08:52:34.708255 IP 192.168.10.1 > 192.168.10.3: ICMP redirect 192.168.20.3 to net 192.168.10.2, length 36
08:52:35.725832 IP 192.168.20.3 > 192.168.10.3: ICMP echo request, id 1, seq 389, length 40
08:52:35.725893 IP 192.168.10.3 > 192.168.20.3: ICMP echo reply, id 1, seq 389, length 40
08:52:35.727511 IP 192.168.10.1 > 192.168.10.3: ICMP redirect 192.168.20.3 to net 192.168.10.2, length 36
```

Outputtet viser både ICMP echo request fra backup-laptoppen (192.168.20.3 → 192.168.10.3) og echo reply retur. Det bekræfter, at trafikken sendes korrekt gennem router 1941 og MLS'en i begge retninger. Der ses også ICMP Redirect-beskeder fra MLS'en. De opstår fordi MLS forsøger at optimere routing ved at foreslå et mere direkte next-hop. I det her tilfælde foreslår MLS'en et redirect, selvom serveren reelt ikke kan nå routerens adresse direkte. Det skyldes, at MLS baserer redirect på sin routing-logik og ikke på den reelle VLAN-adskillelse. TCPDump-screenshottet viser hermed, at routing fungerer som forventet mellem VLAN 10 og VLAN 20, og at trafikken bevæger sig korrekt gennem netværkets Layer 3-infrastruktur.

8.3. Teori af relevante netværk og server-blokke

Vores arkitektur består af en VLAN-segmenteret netværksstruktur, hvor vores Multilayer Switch (MLS) fungerer som routing-kerne, og vores Cisco 1941-router bliver brugt som et særskilt backup-netværk samt til OSPF-kommunikationen med MLS'en. For at skabe et stabilt og sikkert netværk har vi opdelt det i tre VLANs. Vi har VLAN 10 til vores produktionsdel, VLAN 20 til backup, og VLAN 30 som WAN-segmentet mellem MLS og routeren. Hvert VLAN er forsynet med en SVI eller et router subinterface, hvilket gør det muligt at håndtere trafikken på Layer 3 og samtidig bevare en klar logisk opdeling.

VLAN-teknologien giver en struktureret opdeling af broadcast-domæner, hvilket reducerer mængden af potentiel forstyrrende trafik og så øger det sikkerheden. Routing på MLS'en udføres gennem Switch Virtual Interfaces (SVI'er), som virker som gateways for VLAN 10 og VLAN 30. Routeren benytter router-on-a-stick via subinterfaces (Gi0/0.10 og Gi0/0.30) til at håndtere VLAN 10 og VLAN 30, mens VLAN 20 håndteres som et separat Layer-3-interface på Gi0/1. Trunk-forbindelsen mellem MLS og router (1941) er lavet på 802.1Q, som gør det muligt at transportere flere VLANs over ét fysisk link.

Til routing mellem MLS og router anvendes OSPF i area 0. OSPF giver en dynamisk ruteudveksling og eliminerer behovet for statiske ruter. Det gør det mere robust og gør arkitekturen nemmere at skalere op. OSPF udveksler ruter mellem MLS'ens SVI for VLAN 30 og routerens subinterface Gi0/0.30. Det danner det centrale kommunikationslag mellem netværkets produktions- og backupsegmenter. For internetadgang bruger vi en 5G-router, der fungerer som gateway for VLAN 10 og håndterer ESP32-enhederne WiFi-forbindelser. Default route på MLS'en peger mod 5G-routeren, så al trafik til internettet sendes den vej.

Vores arkitektur er lavet ud fra en række bestemte valg. Vi bruger statiske IP-adresser for at sikre forudsigelighed, let fejlfinding og stabilitet. VLAN-segmenteringen opdeler produktions-LAN'et fra backup-LAN'et og mindsker risikoen for uønsket påvirkning LAN'ene imellem. Vi har valgt OSPF frem for statiske ruter for at få bedre fleksibilitet og for at minimere mængden af manuel konfiguration. Vi bruger Router-on-a-stick, da det giver mulighed for håndtering af flere VLANs uden ekstra hardware.

Der er dog også begrænsninger ved vores opbygning. Routerens subinterfaces kan skabe flaskehalse, hvis vi i fremtiden vil opskalere og mere trafik skal sendes igennem samme fysiske port. WiFi-forbindelsen til ESP32-enhederne er mere ustabil end kablede forbindelser og kan blive påvirket af interferens. 5G-routeren er et single point of failure for både WAN og WiFi. Statiske IP'er kræver disciplin i vores dokumentation. OSPF kan skabe utilsigtede forbindelser, hvis interfaces aktiveres forkert, og VLAN 20 er afhængig af routerens stabilitet.

Støjdata bevæger sig fra ESP32-enhederne via 5G-routerens WiFi, gennem VLAN 10 på MLS'en og videre til serveren. Trafikken fra backup-laptoppen går fra VLAN 20 gennem routerens OSPF-routing til VLAN 30 og derfra videre til MLS og serveren. Routing-protokoltrafikken går mellem MLS og routeren via VLAN 30, mens brugerdata fra produktionens-LAN'et primært bevæger sig i VLAN 10.

Arkitekturs opbygning og valg af teknologier er lavet for at sikre et stabilt og overskueligt netværk. Løsningen giver en tydelig opdeling mellem funktionerne og gør driften enkel. Det giver ligeledes mulighed for nemt at udvide systemet senere.

8.4. GDPR.

I dette projekt har GDPR-reglerne haft stor indflydelse på, hvordan vi har valgt at anvende de valgte mikrofoner. Det er vigtigt, at mikrofonerne ikke optager tale eller lagre samtaler, men udelukkende måler DB værdier. Disse typer data kan ikke bruges til at identificere personer eller til stemmegenkendelse og betragtes derfor ikke som personoplysninger i GDPR-forstand (Datatilsynet, u.d.).

Da GDPR kun gælder, når der behandles personoplysninger, er der ikke krav om et lovligt behandlingsgrundlag i dette projekt. Alligevel skal formålet med målingerne begrundes, og her anvendes dataene til støjmonitorering, hvilket er et klart og sagligt formål. Selvom projektet ikke behandler personoplysninger, kan det alligevel være en fordel at gennemføre en risikovurdering (DPIA) eller en mindre vurdering af databeskyttelsesrisici for at dokumentere, at systemet ikke indebærer risiko for privatlivskrænkelser. Derudover vil der blive udarbejdet en teknisk beskrivelse af databehandlingen, samt en kort information/privatlivs meddelelse til de personer, der opholder sig i områder, hvor der foretages målinger. Dette sikrer transparens og understøtter god praksis for databeskyttelse.

9. Linux

Til vores Linux server faldt valget af OS-system på Ubuntu 24.04, dog til en start havde vi valgt at prøve 25.10 "Questing Quokka", men da vi begyndte at sætte vores server op, oplevede vi mange udfordringer med versionen. Dette skyldtes at versionen er ny, og derfor ikke understøttes af mange programmer, som f.eks. pgadmin. Derfor valgte vi Ubuntu 24.04.3 LTS, da vi hermed arbejder med et kendt OS fra undervisningen, samt findes der rigeligt med dokumentation omkring serveren. Derudover stødte vi ind i problemer ved brug af iptables, da det skabte komplikationer, når vi forsøgte at oprette en fast forbindelse igennem Visual Studio Code. Derfor har vi anvendt os af UFW under udviklingen af vores løsning, da vi hermed problemfrit kunne forbinde igennem Visual Studio Code. Når løsningen skulle ud i produktion hos en kunde, ville vi anvende iptables, og dermed havde mere kontrol over hvilke porte er åbne, og hvilke ip/mac-adresser som kan kommunikere med serveren.

I dette projekt bliver Ubuntu-serveren brugt til at hoste vores database samt vores hjemmeside. Serveren bliver der, hvor alt bliver samlet. Yderligere bliver der lavet forskellige Bash-scripts for at automatisere gentagne opgaver. En opgave kunne være at starte et python-script fra vores server hver morgen og afslutte det om eftermiddagen.

Der er lavet et netværks-script, som tjekker, at forskellige dele af netværket er aktive, herunder 1941-routeren, MLS, serveren og backup-laptoppen. Et andet script logger, når serveren tilgås via SSH, så man kan få et overblik over, hvem og hvornår folk har været inde på serveren. Loggen sendes også via SMTP til en mail, så man har den som backup, samme sker med status fra netværks-scriptet.

Til en start vil vi gennemgå hvordan vi via vores bash script, starter vores python-script fra serveren, der tænder for vores hjemmeside, dette sker i bash scriptet som hedder `start_stop_app.sh`

Figur 9.1: Screenshot af Bash script - automatisering af server-start og stop

```
#!/bin/bash

### === INDSTILLINGER === ###
START_TIME="9:00" # Tidsstyring start
STOP_TIME="17:00" # Tidsstyring stop
APP_DIR="/home/server/backend"
VENV_DIR="$APP_DIR/venv"
APP_MODULE="app:app" # Flask-app objekt i app.py
WS_FILE="$APP_DIR/ws_server.py"
RETRY_DELAY=5 # sekunder mellem genstarts-forsøg

### === HJÆLPE-FUNKTIONER === ###

# --- Tjek og ryd PID hvis processen ikke lever ---
fix_pid() {
    local pidfile="$1"
    local pname="$2"

    if [ -f "$pidfile" ]; then
        PID=$(cat "$pidfile")
        if ! kill -0 "$PID" 2>/dev/null; then
            echo "$(date +%F %T) - $pname PID-fil fandtes men processen kørte ikke. Rydder op." >> "$APP_DIR/startup.log"
            rm "$pidfile"
        else
            return 1 # processen kører → ingen opstart
        fi
    fi

    # Process kører uden PID-fil
    RUNNING_PID=$(pgrep -f "$pname" | head -n 1)
    if [ -n "$RUNNING_PID" ]; then
        echo "$RUNNING_PID" "$pidfile"
        echo "$(date +%F %T) - $pname kører uden PID-fil - genskab PID: $RUNNING_PID" >> "$APP_DIR/startup.log"
        return 1
    fi

    return 0
}

# --- Start Gunicorn ---
start_app() {
    fix_pid "$APP_DIR/gunicorn.pid" "gunicorn"
    if [ $? -eq 1 ]; then return; fi

    echo "$(date +%F %T) - Starter Gunicorn..." >> "$APP_DIR/startup.log"
    cd "$APP_DIR" || return

    # Start Gunicorn i baggrunden med virtualenv
    nohup bash -c "source $VENV_DIR/bin/activate && gunicorn --workers 3 --bind 127.0.0.1:8000 $APP_MODULE" \
    & "$APP_DIR/gunicorn.log" 2>&1 &

    sleep 2
    GUNI_PID=$!

    if ! kill -0 "$GUNI_PID" 2>/dev/null; then
        echo "$(date +%F %T) - FEJL: Gunicorn startede ikke korrekt. Tjek gunicorn.log" >> "$APP_DIR/startup.log"
        return
    fi

    echo "$GUNI_PID" > "$APP_DIR/gunicorn.pid"
    echo "$(date +%F %T) - Gunicorn startet med PID: $GUNI_PID" >> "$APP_DIR/startup.log"
}

# --- Start ws_server.py ---
start_ws() {
    fix_pid "$APP_DIR/ws_server.pid" "ws_server.py"
    if [ $? -eq 1 ]; then return; fi

    echo "$(date +%F %T) - Starter ws_server.py..." >> "$APP_DIR/startup.log"

    nohup $VENV_DIR/bin/python "$WS_FILE" > "$APP_DIR/ws_server.log" 2>&1 &
    sleep 1
    WS_PID=$!

    if ! kill -0 "$WS_PID" 2>/dev/null; then
        echo "$(date +%F %T) - FEJL: ws_server.py startede ikke korrekt. Tjek ws_server.log" >> "$APP_DIR/startup.log"
        return
    fi

    echo "$WS_PID" > "$APP_DIR/ws_server.pid"
    echo "$(date +%F %T) - WS-server startet med PID: $WS_PID" >> "$APP_DIR/startup.log"
}

# --- Stop Gunicorn ---
stop_app() {
    if [ -f "$APP_DIR/gunicorn.pid" ]; then
        PID=$(cat "$APP_DIR/gunicorn.pid")
        echo "$(date +%F %T) - Stopper Gunicorn (PID: $PID)..." >> "$APP_DIR/startup.log"
        kill $PID 2>/dev/null
        rm "$APP_DIR/gunicorn.pid"
        echo "$(date +%F %T) - Gunicorn stoppet." >> "$APP_DIR/startup.log"
    else
        echo "$(date +%F %T) - Gunicorn kører ikke." >> "$APP_DIR/startup.log"
    fi
}

# --- Stop ws_server.py ---
stop_ws() {
    if [ -f "$APP_DIR/ws_server.pid" ]; then
        PID=$(cat "$APP_DIR/ws_server.pid")
        echo "$(date +%F %T) - Stopper ws_server.py (PID: $PID)..." >> "$APP_DIR/startup.log"
        kill $PID 2>/dev/null
        rm "$APP_DIR/ws_server.pid"
        echo "$(date +%F %T) - WS-server stoppet." >> "$APP_DIR/startup.log"
    else
        echo "$(date +%F %T) - ws_server.py kører ikke." >> "$APP_DIR/startup.log"
    fi
}

### === MAIN LOOP (TIDSSYRET) === ###
echo "$(date +%F %T) - Starter tidsstyret app-kontrol..." >> "$APP_DIR/startup.log"

while true; do
    CURRENT=$(date +%H:%M)
    echo "DEBUG: CURRENT=$CURRENT, START_TIME=$START_TIME, STOP_TIME=$STOP_TIME" >> "$APP_DIR/startup.log"

    if [ [ "$CURRENT" > "$START_TIME" && "$CURRENT" < "$STOP_TIME" ] ]; then
        start_app
        start_ws
    fi

    if [ "$CURRENT" == "$STOP_TIME" ]; then
        stop_app
        stop_ws
    fi

    sleep 30
done
```

Til en start kalder man en shebang, hvilket angiver at scriptet skal køres med bash-shell.

Næst indstiller vi start- og slutid for hvornår scriptet skal starte og stoppe. I dette script er det kl. 9 om morgen og kl. 17 om eftermiddagen, da dette er indenfor normal arbejdsdag.

APP_DIR er den mappe som de python programmer der skal startet ligger i. VENV_DIR viser hvor det virtuelle environment ligger, og til sidst er vist vej til de to Python scripts som skal køres via \$APP_DIR og navnet på dem, yderligere en restart efter fem sekunder, hvis det ikke virkede.

Efterfølgende har vi en funktion, som hedder fix_pid. Dens formål er at den tjekke om der allerede er en proces, som kører også selvom at PID-filen kører eller mangler. Den tjekker efterfølgende om PID-filen findes, men hvis processen ikke kører så bliver den slettet, og hvis processen kører uden en PID-fil så bliver den genskabt.

Næst har vi en funktion, som hedder start_app. Her starter de to scripts. Først bliver der tjekket om der allerede findes et gunicorn .pid. Denne bliver kørt i virtualenv, da det er her Flask, og andre pakker ligger, det hele bliver logget i gunicorn.log, og PID bliver gemt i gunicorn.pid til senere stop.

Næste funktion er start_ws, og her foregår næsten det samme som i ovenstående funktion, men det er til vores WebSocket server.

I dette bash script er der også en automatisk stop til begge start funktioner som blev gennemgået før, stopfunktionen sker ved at der i starten bliver tjekket om der findes en app.pid, hvis ja læses PID og logger at appen stoppes. Dette sker ved at sende en kill command, hvilket sletter PID-filen, samt bliver det logget hvis PID-filen ikke findes, fordi den allerede var stoppet før scriptet kørte.

Til sidst har vi lavet en manuel start af test. Disse er dog kommenteret ud, da vi ville bruge fastlagte tidspunkter. Tidspunkterne bruges sammen med Crontab, så vi har en form for "failsafe" hvis scriptet skulle fejle så vil Crontab fortsat aktivere det. Næste script gennemgår vores automatiske logging af SSH-forbindelser til vores server, dette bliver gjort i følgende script som hedder log_ssh.sh.

Figur: 9.2: Screenshot af Bash script - Logging af SSH-forbindelser

```
#!/bin/bash

LOGFILE="/home/server/scripts/ssh_logins.log"

TIMESTAMP=$(date "+%Y-%m-%d %H:%M:%S")
USER=$USER
FROM=$(echo $SSH_CLIENT | awk '{print $1}')

INFO="$TIMESTAMP - USER: $USER FROM: $FROM"
echo "$INFO" >> "$LOGFILE"

# Send mail
MAIL_TO="alerts@ligegyldigmail.dk"
MAIL_FROM="noor@ligegyldigmail.dk"
SUBJECT="SSH login registreret på serveren"
BODY="Der er registreret et SSH-login:\n$INFO"

printf "Subject: %s\nFrom: %s\nTo: %s\n\n%s\n" \
"$SUBJECT" "$MAIL_FROM" "$MAIL_TO" "$BODY" \
| msmtp -a brevo "$MAIL_TO"
```

I dette script starter vi igen en shebang som fortæller systemet at dette script skal køres med bash.

I scriptet fortæller vi hvor at alle ssh login skal blive logget, hvilket sker i ssh_logins.log. Næst henter vi dato og tid til logging, dette bliver gemt i variabelen TIMESTAMP.

Næst bliver der logget hvilken bruger det er som har logget ind, dette bliver gemt i variabelen USER.

I FROM variabelen bliver det via \$SSH_CLIENT gemt hvilken ip adresse, port og local port det er som er logget ind, og vælger vi kun at `| awk '{print $1}'` hvilket betyder at vi kun henter det første felt, hvilket er IP-adressen som logger ind.

Disse informationer bliver til en loglinje, som man ser i INFO, og dette bliver sendt til logfilen.

Udover dette bliver gemt lokalt, har vi valgt at der skal sendes en mail udenfor serveren. Dette sker via en mail som modtager alarmen, dette bliver sendt via msmtp, som er en SMTP-klient til Linux.

Figur 9.3: Screenshot af e-mail fra SSH-forbindelse

SSH login på serveren fra 192.168.10.52



From noor@ligegyldigmail.dk on
15.12.2025 09:59

[Details](#) [Headers](#)

Der er registreret SSH-login:

Bruger: server
Fra IP: 192.168.10.52
Tidspunkt: 2025-12-15 09:59:00

Næste bash script vi har er vores automatiske backup af vores database, dette script hedder

pg_backup.sh

Figur 9.4: Screenshot af Bash script - automatisering af backup af database

```
#!/bin/bash
set -euo pipefail

# ----- KONFIGURATION -----
DB_NAME="postgres"
DB_USER="postgres"
PGHOST="127.0.0.1"
PGPORT="5432"

BACKUP_USER="nico7634"
BACK_HOST="192.168.20.50"
BACKUP_DIR="/home/nico7634/db_backups"

SSH_KEY="${HOME}/.ssh/id_backup"
RETENTION_DAYS=30

# ----- FILNAVNE & TMP -----
TIMESTAMP=$(date +%Y%m%d-%H%M%S)
FILENAME="${DB_NAME}-${TIMESTAMP}.sql.gz"
LOCAL_TMP=$(mktemp --tmpdir="${TMPDIR:-/tmp}" "dbbackup-XXXXXX-${FILENAME}")

# ----- CLEANUP -----
cleanup() {
    rc=$?
    rm -f "$LOCAL_TMP" || true
    exit $rc
}
trap cleanup EXIT INT TERM

# ----- ERROR FUNCTION -----
err() { echo "FEJL: $*" >&2; exit 1; }

# ----- KOMMANDO-TJEK -----
for cmd in pg_dump gzip scp ssh mktemp date find; do
    if ! command -v "$cmd" >/dev/null 2>&1; then
        err "Påkrævet kommando mangler: $cmd"
    fi
done

# ----- PGPASS ADVARSEL -----
if [[ ! -f "${HOME}/.pgpass" ]]; then
    echo "Advarsel: ~/.pgpass findes ikke. Hvis Postgres kræver password, vil pg_dump bede om det." >&2
fi

# ----- SIKR FJERNMAPPE -----
echo "Sikrer fjern-mappe på ${BACK_HOST}..."
ssh -i "$SSH_KEY" -o BatchMode=yes -o ConnectTimeout=10 \
    "${BACKUP_USER}@${BACK_HOST}" "mkdir -p '${BACKUP_DIR}'" \
    || err "Kunne ikke oprette/få adgang til ${BACK_HOST}:${BACKUP_DIR} (ssh fejl)"

# ----- LAV BACKUP -----
echo "Starter pg_dump for database '${DB_NAME}' (host: ${PGHOST}:${PGPORT})..."
PGPASSWORD="${PGPASSWORD:-}" \
pg_dump -h "${PGHOST}" -p "${PGPORT}" -U "${DB_USER}" "${DB_NAME}" \
2>/tmp/pg_dump.err | gzip > "$LOCAL_TMP" || {
    echo "pg_dump fejlede. Sidste fejl (kort):" >&2
    tail -n 50 /tmp/pg_dump.err || true
    err "pg_dump mislykkedes"
}

# ----- SEND BACKUP TIL VM -----
echo "Overfører backup til ${BACK_HOST}:${BACKUP_DIR}/${FILENAME} ..."
scp -i "$SSH_KEY" -o BatchMode=yes -o ConnectTimeout=20 \
    "$LOCAL_TMP" "${BACKUP_USER}@${BACK_HOST}:${BACKUP_DIR}/${FILENAME}" \
    || err "scp fejlede"

# ----- FJERN GAMLE BACKUPS -----
echo "Fjerner backups ældre end ${RETENTION_DAYS} dage på backup-hosten..."
ssh -i "$SSH_KEY" -o BatchMode=yes "${BACKUP_USER}@${BACK_HOST}" \
    "find '${BACKUP_DIR}' -maxdepth 1 -type f -name '*.sql.gz' -mtime +${RETENTION_DAYS} -print -delete || true"

# ----- FÆRDIG -----
echo "Backup færdig: ${BACK_HOST}:${BACKUP_DIR}/${FILENAME}"
```

Til en start bliver der fortalt at scriptet skal køres i bash, dette kan ses med et shebang. Som det næste bliver der sat -euo pipefail, hvilket betyder at scriptet skal stoppe ved første fejl. Fejlen kan være hvis en variable ikke er defineret, eller hvis en af pipe fejler, dette bliver gjort så scriptet er mere robust over for eventuelle fejl.

Derudover bliver der konfigureret, hvilken database det er, som skal dumpes, hvilken bruger der skal bruges til dette og hvilken IP-adresse databasen er på, samt dens port.

Det er også her vores backup bruger bliver defineret, dette er både brugeren, og dens IP-adresse, samt hvor hos backup brugeren databasen skal gemmes, nemt bliver der sat en privat SSH-nøgle så man kan forbinde uden brug for adgangskode, og til sidst bliver backups ældre end 30 dage slettet.

Den næste del af scriptet bliver der oprettet et unikt timestamp, som bliver navnet på filen som bliver sendt. Mktmp laver en midlertidig fil i /tmp, samt en fallback, hvis \$TMPDIR ikke findes.

Efterfølgende bliver der lavet en automatisk cleanup. Dette bliver gjort så den sletter midlertidig backup-fil, og returnerer med en exit kode hvis dette fejler, samt køres der en trap som sikrer cleanup, ved at der bliver exit, kørt CTRL+c og kill. Dette bliver gjort så der ikke bliver efterladt halvfærdige backup-filer.

Næste funktion er en error-funktion. Hvis det skulle være en fejl, ville dette blive udskrevet og afsluttet. I kommandotjek bliver der kontrolleret at de nødvendige programmer findes, så hvis pg_dump mangler bliver scriptet stoppet inden man forsøger at køre scriptet færdigt.

I næste del af koden bliver der kontrolleret om PostgreSQL kan autentificeres uden en adgangskode. Der bliver også kontrolleret om den mappe som backuppen skal sendes til på backup-laptoppen findes. Dette sker via brug af SSH-nøglen, samt køres det mkdir -p til oprettelse, og sender muligt en fejl hvis scriptet ikke kan oprettes eller har adgang.

Efterfølgende bliver selve backuppen lavet dette sker ved at pg_dump laver et SQL-dump af databasen, og outputtet bliver gemt som en gzip-fil, samt hvis noget fejler udskrives de sidste 50 linjer af fejlloggen og stopper scriptet, så man kan eftersøge hvor fejlen er.

Så bliver backup sendt til backupserveren, dette sker via SSH-forbindelsen, og der bliver som beskrevet før, slettet backup ældre end 30 dage.

Følgende script er lavet i Bash. Vi bruger det til at overvåge vores netværk. Vi tester om der er gennemgang i hele netværket ved hjælp af ICMP-ping. Resultaterne af hver test bliver logget og hvis der opdages at dele af netværket ikke kan pinges, registreres det som en fejl og der vil blive sendt en mail via SMTP med en alarm om at dele af netværket ikke er tilgængeligt.

Figur 9.5: Screenshot af Bash script - Overvågning af netværk og alarm-e-mail

```
#!/bin/bash

FAILED=0
ERRORS=""

GREEN='\e[32m'
RED='\e[31m'
NC='\e[0m'

LOGFILE="$HOME/scripts/logs/net_healthcheck.log"
TIMESTAMP=$(date '+%Y-%m-%d %H:%M:%S')

echo "---- Netværks-healthcheck ($TIMESTAMP) ----"
echo "---- Netværks-healthcheck ($TIMESTAMP) ----" >> "$LOGFILE"

declare -A HOSTS=(
    ["MLS"]="192.168.10.1"
    ["Router 1941 VLAN10"]="192.168.10.2"
    ["Server (localhost test)"]="192.168.10.3"
    ["Backup Laptop VLAN20"]="192.168.20.3"
)

for NAME in "${!HOSTS[@]}; do
    IP=${HOSTS[$NAME]}
    echo "Tester $NAME ($IP)..."
    echo "Tester $NAME ($IP)..." >> "$LOGFILE"

    if ping -c 1 $IP > /dev/null 2>&1; then
        MSG="[OK] $NAME svarer på ping!"
        echo -e "${GREEN}$MSG${NC}"
    else
        MSG="[FEJL] $NAME svarer ikke! Tjek netværket."
        echo -e "${RED}$MSG${NC}"
        FAILED=1
        ERRORS+="$MSG"$'\n'
    fi

    echo "$MSG" >> "$LOGFILE"
    echo "-----"
    echo "-----" >> "$LOGFILE"
done

if [ "$FAILED" -ne 0 ]; then
    MAIL_TO="alerte@ligegyldigmail.dk"
    MAIL_FROM="noor@ligegyldigmail.dk"
    SUBJECT="Netværksfejl på serveren $(hostname)"

    BODY="Der er registreret en fejl i netværks-healthcheck scriptet $(date '+%Y-%m-%d %H:%M:%S')
    Folgende tests fejlede:
    $ERRORS

    Se ogsaa logfilen:
    $LOGFILE"

    printf "Subject: %s\nFrom: %s\nTo: %s\n\n%s\n" "$SUBJECT" "$MAIL_FROM" "$MAIL_TO" "$BODY" | msmtp -a breve "$MAIL_TO"
fi
```

#!/bin/bash i starten af scriptet betyder at scriptet bliver kørt af Bash-fortolkeren. De efterfølgende linjer:

FAILED=0

ERRORS=

er hvor to vigtige variabler bliver initialiseret. FAILED virker som en indikator på om der er fejl i systemet eller ej, og med ERRORS opsamler vi de fejl der evt. måtte hænde.

For at forbedre læsbarheden, hvis vi kører scriptet manuelt i terminalen, har vi farvekodet outputtet, så man hurtigt har et overblik over, om der er fejl eller ej. Grøn for "ALT OK" og rød for "FEJL". Det er ANSI-escape-koder og de bruges kun til at gøre det visuelt nemmere at overskue. Efterfølgende definerer vi, hvor vi vil placere logfilen som bliver logged, hver gang scriptet kører:

LOGFILE="\$HOME/scripts/logs/net_healthcheck.log"

Timestampet: TIMESTAMP=\$(date '+%Y-%m-%d %H:%M:%S') bliver brugt i både output i terminalen og i logfilen for at gøre det nemmere at finde ud af, hvornår der er sket fejl på netværket.

Der bliver lavet en overskrift i terminal og logfilen inden testen køres:

```
echo "- - - Netværks-healthcheck ($TIMESTAMP) - - -"
```

```
echo "- - - Netværks-healthcheck ($TIMESTAMP) - - -" >> "$LOGFILE"
```

Når testen sættes i gang, definerer vi først, hvilke netværksenheder, som skal testes. Det gør vi i et associativt array. Alle enheder består af både et navn og en IP-adresse som f.eks.: `["MLS"]="192.168.10.1"`.

Det er en god måde at bygge det op på, da vi kan tilføje eller fjerne enheder som det passer os uden at ændre i selve scriptet.

Bagefter itererer vi over alle de definerede hosts i et for-loop og på den måde tjekker for om der kan pinges fra og til alle enheder:

```
for NAME in "${!HOSTS[@]}"; do
    IP=${HOSTS[$NAME]}
    if ping -c 1 $IP > /dev/null 2>&1; then
```

Hvis testen er positiv får vi i grøn tekst vist meddelelsen på følgende måde:

```
MSG="[OK] $NAME svarer på ping!"
```

Hvis betingelsen for en fejl er opfyldt:

```
if [ "$FAILED" -ne 0 ]; then
```

Vil der blive skrevet og sendt en e-mail med de forskellige informationer om, hvilken enhed der fejler og hvornår fejlen er sket. Korrekt formatering sker med `printf`, hvorefter der sendes via `msmtp`:

```
printf "Subject: %s\nFrom: %s\nTo: %s\n\n" . . . | msmtp -a brevo "$MAIL_TO"
```

På den måde sikrer vi, at netværksfejl bliver opdaget og vi kan dermed handle med det samme. I øvrigt køres denne netværksscanning med `Crontab` én gang i timen. Hvor ofte netværksscanningen skal køres, kan indstilles efter behov.

9.1. Hardeningliste

Hardening-processen er en grundlæggende del af Linux server-konfiguration. Det er essentielt at de opsatte enheder, som kører med et Linux OS, opholder basale sikkerhedskrav; server miljøet er sårbart over for ondsindede aktører, hvilket kan medføre datalækage og cyberangreb. Derfor er det normal praksis at opsætte en hardening-liste, hvori man opsætter krav for serverkonfigurationen (serverion.com, 8).

De opsatte brugere, må i konfigurationsfasen inden den bliver sat i drift, godt have et monotont navn som `admin` el., men skal efter have ændret navn til et mere unikt ID for at minimere faren for brute-

force angreb. Yderligere skal der være opsat filrettigheder. Derved kan vi kontrollere, hvilke brugere, der har skrive-, læse- og eksekverer-rettighe

Vores filosofi for vores bruger- og filrettigheder tager udgangspunkt i Zero-Trust modellen. Den går helt essentielt ud på, at brugere af et system kun har rettigheder til de software og mapper, som er absolut nødvendige for netop deres arbejde og virke. Cybersikkerhed er det vilde vesten, så hvis en bruger har adgang til at eksekvere en filtype, kan det være kritisk for datasikkerheden i netværksinfrastrukturen.

Hardening-listen er lavet ud fra de tiltag, vi allerede har implementeret i projektets udarbejdelse, samt hvilke tiltag som vi mener at der skal implementeres, hvis systemet skal sættes i drift.

1. Systemopdateringer ved nye sikkerheds features
2. Linux UFW Firewall
3. Brugerrettigheder
4. Filrettigheder.
5. IP-Tables (f.eks. drop alle ICMP protokol request)
6. tcpdump
7. System login log over når SSH port bruges
8. MAC-Adresse whitelist
9. Playbooks
10. Port ændringer væk fra mest brugte porte
11. Email-notifikation ved login eller nedbrud
12. Mounting permission

10. Krav, prioritet, accepttestprocedure og resultater

I dette projekt skulle gruppen udvikle egne krav og prioriteter, dette er blevet valgt ud fra, hvor vigtig denne test er for den generelle brug af løsningen. Følgende farver er brugt til opdeling af krav til de forskellige fag, det skal nævnes at flere af kravene passer til flere fag.

ILS LINUX PYTHON NETVÆRK

ILS:

ID 1	Krav Mikrofonerne skal sende data om støjniveauet igennem deres tilsluttede ESP32 til ubuntu-serveren og aktuatorerne	Prioritet 1
Kategori ILS	Accepttest Dataen fra mikrofonerne skal kunne behandles og fremvises på serveren. Derudover skal aktuatorerne kunne handle ud fra dataen.	Godkendt
Billede og link til video af testen <i>Udklip af terminal på ESP32 forbundet til INMP441-mikrofoner.</i> <i>Data bliver sendt til WebSocket som fremviser data på webserver, samt over ESP-NOW til aktuator</i> <pre>JSON: {"current_dBSPL": 55.5, "top_punkt": 65.7} Sent WebSocket: {"current_dBSPL": 55.5, "top_punkt": 65.7} JSON: {"current_dBSPL": 54.7, "top_punkt": 65.7} Sent ESP-NOW: {"current_dBSPL": 54.7, "top_punkt": 65.7} JSON: {"current_dBSPL": 45.3, "top_punkt": 65.7} Sent WebSocket: {"current_dBSPL": 45.3, "top_punkt": 65.7} JSON: {"current_dBSPL": 56.8, "top_punkt": 65.7} Sent ESP-NOW: {"current_dBSPL": 56.8, "top_punkt": 65.7}</pre>		

ID 2	Krav: Design af sensor-kredsløb Design et kredsløb med en INMP441 sensor, som opfanger støjniveau fra et givent lokale.	Prioritet 1
--------------------	---------------------------------------------------------------------------------------------------------------------------------------	---------------------------

Kategori	Accepttest	Godkendt
ILS	Hvis vi kan se digital output fra kredsløbet i vores terminal, er kravet godkendt.	
<p>Billede og link til video af testen</p> <p>Udklip af terminalen som viser den "rå" data i terminalen. Dette script viser data, før det bliver lavet til JSON format.</p> <pre> Warming up microphone... Live SPL Meter - Ctrl+C to stop dB SPL: 63.7 {"current_dBSPL": 63.7, "top_dBSPL": 63.7} dB SPL: 67.1 {"current_dBSPL": 67.1, "top_dBSPL": 67.1} dB SPL: 59.4 {"current_dBSPL": 59.4, "top_dBSPL": 67.1} dB SPL: 52.9 {"current_dBSPL": 52.9, "top_dBSPL": 67.1} dB SPL: 54.0 {"current_dBSPL": 54.0, "top_dBSPL": 67.1} dB SPL: 65.0 {"current_dBSPL": 65.0, "top_dBSPL": 67.1} dB SPL: 53.6 </pre>		

ID	Krav: Design af sensor-kredsløb	Prioritet
3	Design kredsløb med en servomotor, som hejser et "flag" ved høj støj.	1
Kategori	Accepttest	Godkendt
ILS	Hvis der bliver larmet i lokalet og dette overskrider den opsatte grænseværdi i kredsløbet, skal servomotoren aktiveres og hejse et "flag".	
<p>Billede og link til video af testen</p> <p>Link til video:</p> <p>https://drive.google.com/file/d/1TjZMgIWfVmnwI8lELa04-E5fGVHAN19z/view?usp=drive_link</p>		

ID 4	Krav: Design af sensor-kredsløb Design kredsløb med en steppermotor, som sænker lyddæmpende materiale.	Prioritet 1
Kategori ILS	Accepttest Hvis der bliver larmet i lokalet og dette bliver overskrider den opsatte grænseværdi i kredsløbet, skal steppemotoren aktiveres og sænke lyddæmpende materiale.	Godkendt
Billede og link til video af testen Indsæt link til video her https://drive.google.com/file/d/1n2dkhW7VZ25ZeFGzTcl0XIWT7K7KCdSb/view?usp=drive_link		

ID 5	Krav Stabil forbindelse mellem ILS enhederne.	Prioritet 2
Kategori ILS	Accepttest De motorstyrede elementer agerer som forventet i forhold til variable ændringer fra sensorer og definerede grænseværdier.	Godkendt
Billede og link til video af testen som vist i krav 4 videoen er der stabil forbindelse mellem vores LIS enheder, dog oplevede vi i starte problemer med ESP-NOW dog var disse fejl mindre til sidst i projektet.		

Udklip af at der blive sendt dB data fra INMP441 kredsløb, videre til aktuator kredsløb over ESP-NOW



```
JSON: {"current_dBSPL": 53.2}
Sent ESP-NOW: {"current_dBSPL": 53.2}
JSON: {"current_dBSPL": 44.7}
Sent ESP-NOW: {"current_dBSPL": 44.7}
JSON: {"current_dBSPL": 49.3}
Sent ESP-NOW: {"current_dBSPL": 49.3}
JSON: {"current_dBSPL": 43.0}
Sent ESP-NOW: {"current_dBSPL": 43.0}
JSON: {"current_dBSPL": 57.7}
Sent ESP-NOW: {"current_dBSPL": 57.7}
JSON: {"current_dBSPL": 46.5}
Sent ESP-NOW: {"current_dBSPL": 46.5}
JSON: {"current_dBSPL": 52.8}
Sent ESP-NOW: {"current_dBSPL": 52.8}
```




ID 6	Krav Undersøgelse af elektronisk støj	Prioritet 1
Kategori ILS	Accepttest Via undersøgelsen af vores kredsløb, ser vi hvor den elektriske støj kommer fra og hvad vi kan mindske den med.	Godkendt
	<p><i>Udfra vores støjmålinger i punkt 6.#, har vi vurderet at den elektriske støj vi oplever på kredsløbet, må komme fra blandede forbindelser på fumlebrættet. Dette kan rettes ved at sætte kredsløbet på et PCB.</i></p> <p><i>Dog ser vi langt mere EMC forurening. Det er især ESP-NOW som operere på 2.4GHz båndet, som oplever problemer med interferens støj.</i></p>	

ID 7	Krav Lokalisering af støjforurening i lokalet med de to mikrofoner.	Prioritet 3
Kategori ILS	Accepttest Undersøg hvilken af mikrofonerne der opfanger mest støj.	Ikke testet
Billede og link til video af testen Vi har ikke testet for dette krav.		

Linux:

ID 9'8	Krav: Bash Script Lav et script som starter løsningen op og lukker løsningen ned igen på to prædefinerede tidspunkter - Når første medarbejder møder ind og når sidste medarbejder går hjem.	Prioritet 1
Kategori Linux	Accepttest Overvåg om løsningen starter op og lukker ned på de to definerede tidspunkter.	Godkendt
Billede og link til video af testen Udklip viser system log, over hvornår serveren er startet og stoppet.		
<pre>DEBUG: CURRENT=15:56, START_TIME=9:00, STOP_TIME=17:00 2025-12-16 09:06:55 - Starter tidsstyret app-kontrol... DEBUG: CURRENT=09:06, START_TIME=9:00, STOP_TIME=17:00</pre>		

ID 10	Krav Lav et bash script som overvåger netværksetup og sender mail, hvis fejl opstår i forbindelse.	Prioritet 1
Kategori Linux	Accepttest Der skal oprettes en logfil med dagens dato. Log-filen skal indeholde status på netværk, og mail bliver sendt hvis fejl opstår.	Godkendt
	<p>Billede og link til video af testen</p> <p>Her ser billede fra hvori log filen på serveren dette bliver logget.</p> <pre>----- Netværks-healthcheck (2025-12-15 11:00:01) ----- Tester Server (localhost test) (192.168.10.3)... [OK] Server (localhost test) svarer på ping! ----- Tester Router 1941 VLAN10 (192.168.10.2)... [OK] Router 1941 VLAN10 svarer på ping! ----- Tester MLS (192.168.10.1)... [OK] MLS svarer på ping! ----- Tester Backup Laptop VLAN20 (192.168.20.3)... [FEJL] Backup Laptop VLAN20 svarer ikke! Tjek netværket. -----</pre> <p>Samt bliver dette sendt til email via smtp som ekstern logging</p> <div><p>Netværksfejl på serveren 5agruppe  </p><div> From noor@ligegyldigmail.dk on 15.12.2025 11:00</div><div> Details  Headers</div><p>Der er registreret en fejl i netværks-healthcheck scriptet 2025-12-15 11:00:11</p><p>Følgende tests fejlede: [FEJL] Backup Laptop VLAN20 svarer ikke! Tjek netv=C3=A6rket.</p><p>Se ogsaa logfilen: /home/server/scripts/logs/net_healthcheck.log</p></div>	

ID 11	Krav Ubuntu-serveren skal oprette log af hvem der tilgår serveren. Eventuelt hvilken host-ip som tilgår, samt host-brugernavn.	Prioritet 1
Kategori Linux	Accepttest Logfilen skal kunne ses i det directory vi ønsker, samt vi skal kunne se relevant information i log-filen.	Godkendt
<p>Billede og link til video af testen</p> <div> <p>SSH login på serveren fra 192.168.10.52  </p> <p>From noor@ligegyldigmail.dk on 15.12.2025 09:59</p> <p> Details Headers</p> <p>Der er registreret SSH-login:</p> <p>Bruger: server Fra IP: 192.168.10.52 Tidspunkt: 2025-12-15 09:59:00</p> <p>Her ser man at via vores script kan vi se at der er blevet logget ind via ssh på serveren via ip adressen 192.168.10.52 hvilket i dette tilfælde er Nicolai's bærbar. dette login kan også se i vores ssh_logins.log</p> <pre>2025-12-15 09:59:00 server 192.168.10.52</pre> </div>		

ID 12	Krav: Pentest af ubuntu-server <ul style="list-style-type: none"> - Brug af Bruteforce med de mest kendte wordlists - Portscanning med relevante script værktøjer 	Prioritet 3
Kategori Linux	Accepttest Serveren skal kunne modstå vores forsøg på at få uautoriseret adgang i henhold til vores hardening-liste.	Ikke testet
<p>Billede og link til video af testen</p> <p><i>Vi har ikke testet for dette krav.</i></p>		

Netværk:

ID 13	Krav Opsætning af MLS og 1941 router.	Prioritet 1
Kategori Netværk	Accepttest Vi skal kunne se routeren som neighbors i deres respektive CLI's.	Godkendt
	<p>I dette krav skulle vores MLS og 1941 router forbindes, dette gør vi ved at sætte dem som Neighbors via OSPF i det VLAN som går mellem dem det som vi kalder for vores WAN, dette bliver gjort som kan ses i billedet der viser at man på vores MLS kan se routeren 2.2.2.2 som er 1941 router og MLS som er 1.1.1.1.</p> <pre> Switchshow ip interface brief include Vlan30 Vlan30 192.168.30.1 YES manual up Switchshow ip route include 192.168.30.0 C 192.168.30.0/24 is directly connected, Vlan30 Switchshow interfaces trunk Port Mode Encapsulation Status Native vlan Gi1/0/1 on 802.1q trunking 1 Port Vlans allowed on trunk Gi1/0/1 10,20,30 Port Vlans allowed and active in management domain Gi1/0/1 10,30 Port Vlans in spanning tree forwarding state and not pruned Gi1/0/1 10,30 Switchshow ip ospf interface Vlan30 Vlan30 is up, line protocol is up Internet Address 192.168.30.1/24, Area 0 Process ID 1, Router ID 1.1.1.1, Network Type BROADCAST, Cost: 1 Transmit Delay is 1 sec, State DR, Priority 1 Designated Router (ID) 2.2.2.2, Interface address 192.168.30.2 Backup Designated router (ID) 1.1.1.1, Interface address 192.168.30.1 Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5 nob-resync timeout 40 Hello due in 00:00:02 Supports Link-Local Signaling (LLS) Cisco NSF helper support enabled ITF NSF helper support enabled Index 3/3, flood queue length 0 Next 0x0(0)/0x0(0) Last flood scan length is 1, maximum is 2 Last flood scan time is 0 msec, maximum is 9 msec Neighbor Count is 1, Adjacent neighbor count is 1 Adjacent with neighbor 2.2.2.2 (Designated Router) Suppress hello for 0 neighbor(s) Switch# </pre>	

ID 14	Krav: Sikkerhed <ul style="list-style-type: none"> - Netværkssikkerhed - Konfigurering af firewall - Protokolstyring - Portkontrol - Hardeningliste 	Prioritet 1
---------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------

Kategori	Acceptttest	Godkendt
Netværk	<p>Der gennemføres en test af netværkstrafik, hvor både godkendte og ikke-godkendte protokoller, porte og IP-adresser afprøves. Trafik, som ifølge den definerede firewall-konfiguration skal være tilladt, skal passere, mens al øvrig trafik skal afvises. Acceptkriteriet er, at systemet kun giver adgang via de eksplicit godkendte protokoller og porte, mens alle andre forsøg blokeres.</p>	
	<pre>PS C:\Users\s nema> ping 192.168.10.3 Pinging 192.168.10.3 with 32 bytes of data: Reply from 192.168.10.3: bytes=32 time=10ms TTL=64 Reply from 192.168.10.3: bytes=32 time=11ms TTL=64 Reply from 192.168.10.3: bytes=32 time=11ms TTL=64 Reply from 192.168.10.3: bytes=32 time=11ms TTL=64 Ping statistics for 192.168.10.3: Packets: Sent = 4, Received = 4, Lost = 0 (0% loss), Approximate round trip times in milli-seconds: Minimum = 10ms, Maximum = 11ms, Average = 10ms PS C:\Users\s nema> ping 192.168.10.3 Pinging 192.168.10.3 with 32 bytes of data: Request timed out. Request timed out. Request timed out. Request timed out. Ping statistics for 192.168.10.3: Packets: Sent = 4, Received = 0, Lost = 4 (100% loss), PS C:\Users\s nema> ssh 192.168.10.3 server@192.168.10.3's password: Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.8.0-88-generic x86_64)</pre> <p>i denne test vises der at man ved hjælp af oprettede sikkerhed kan ændre på hvad serveren acceptere.</p> <p>Ændringer blev foretaget, men følgende kommandoer</p> <p><i>/etc/ufw/before.rules</i></p> <pre># ok icmp codes for INPUT -A ufw-before-input -p icmp --icmp-type destination-unreachable -j DROP -A ufw-before-input -p icmp --icmp-type time-exceeded -j DROP -A ufw-before-input -p icmp --icmp-type parameter-problem -j DROP -A ufw-before-input -p icmp --icmp-type echo-request -j DROP</pre>	

ID 15	Krav Mikrocontrollerne forbindes til Ubuntu-serveren igennem vores 5G-router.	Prioritet 1
Kategori Netværk	Accepttest Vi skal kunne pinge fra vores Linux-server ud til alle ESP32'ere	Godkendt
	<p>Billede og link til video af testen</p> <p><i>Mikrokontrolleren forbinder til Wi-Fi netværk, med en IP.</i></p> <p><i>ping fra server til esp</i></p> <pre>server@Sagruppe:~\$ ping 192.168.10.4 PING 192.168.10.4 (192.168.10.4) 56(84) bytes of data. 64 bytes from 192.168.10.4: icmp_seq=1 ttl=64 time=87.3 ms 64 bytes from 192.168.10.4: icmp_seq=2 ttl=64 time=130 ms 64 bytes from 192.168.10.4: icmp_seq=3 ttl=64 time=24.2 ms 64 bytes from 192.168.10.4: icmp_seq=4 ttl=64 time=55.8 ms 64 bytes from 192.168.10.4: icmp_seq=5 ttl=64 time=65.5 ms 64 bytes from 192.168.10.4: icmp_seq=6 ttl=64 time=234 ms 64 bytes from 192.168.10.4: icmp_seq=7 ttl=64 time=129 ms ^C --- 192.168.10.4 ping statistics --- 7 packets transmitted, 7 received, 0% packet loss, time 6008ms rtt min/avg/max/mdev = 24.199/103.750/234.290/64.079 ms</pre> <pre>Connecting to Wi-Fi... Connecting to Wi-Fi... Connecting to Wi-Fi... Connected! Network config: ('192.168.10.4', '255.255.255.0', '192.168.10.1', '8.8.8.8') JSON: {"current_dBSPL": 84.5} Sent ESP-NOW: {"current_dBSPL": 84.5} JSON: {"current_dBSPL": 55.4} Sent ESP-NOW: {"current_dBSPL": 55.4}</pre>	

ID 16	Krav Pentest af vores router	Prioritet 3
Kategori Netværk	Accepttest Kan vi få adgang til router eller ej. skal uddybes	Ikke testet
	<p>Billede og link til video af testen</p> <p><i>Vi har ikke testet for dette krav.</i></p>	


Python:

ID 17	Krav Data fra ESP32-enhederne indsættes automatisk i PostgreSQL-databasen på Linux-serveren.	Prioritet 1
Kategori Python	Accepttest Nye data skal kunne verificeres ved, at de fremgår korrekt i de relevante tabeller i PostgreSQL-databasen..	Godkendt

Billede og link til video af testen

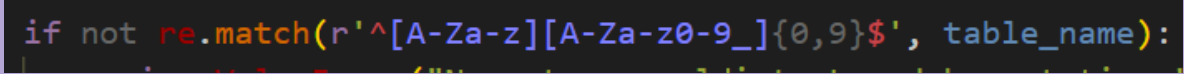
Udklipet viser at dB-data kommer ind i det rigtige table, samt viser timestamp at data'en er ny.

		id integer	* timestamp timestamp without time zone	* db smallint
<input type="checkbox"/>	>	1	2025-12-10 12:59:03.464526	51
<input type="checkbox"/>	>	2	2025-12-10 12:59:04.074788	55
<input type="checkbox"/>	>	3	2025-12-10 12:59:07.154286	53
<input type="checkbox"/>	>	4	2025-12-10 12:59:08.410621	54
<input type="checkbox"/>	>	5	2025-12-10 12:59:10.381661	60
<input type="checkbox"/>	>	6	2025-12-10 12:59:12.118601	56
<input type="checkbox"/>	>	7	2025-12-10 12:59:14.838472	53
<input type="checkbox"/>	>	8	2025-12-10 12:59:16.336952	55
<input type="checkbox"/>	>	9	2025-12-10 12:59:18.323031	55
<input type="checkbox"/>	>	10	2025-12-10 12:59:20.269463	55
<input type="checkbox"/>	>	11	2025-12-10 12:59:22.237644	52

ID 18	Krav Opsætning af Flask-applikation for at fremvise data fra PostgreSQL-databasen.	Prioritet 1
Kategori Python	Accepttest Databasens indhold skal vises visuelt på hjemmesiden.	Godkendt
<p>Billede og link til video af testen</p> 		

ID 19	Krav Pytest af kode	Prioritet 1
Kategori	Accepttest Vores kode skal bestå pytest med 100%	Godkendt

Python		
Billede og link til video af testen <i>Se afsnit ang. Pytest for uddybning af vores Pytesting</i>		

ID 20	Krav Regex anvendes i funktionel kode	Prioritet 1
Kategori Python	Accepttest database bliver oprettet med regex-reglerne og opfyldes	Godkendt
Billede og link til video af testen  <pre>if not re.match(r'^[A-Za-z][A-Za-z0-9_]{0,9}\$', table_name):</pre>		

10.1. Brugertest og resultater

Dato	Testbeskrivelse og -resultater
16.12.25	<p>Gruppen satte sig i et lokale hvor testen af vores løsning blev gennemført, denne test varede cirka 30 minutter, og kig på at vi ville tjekke at hele løsningen kunne fungere sammen.</p> <p>Ved opstart virkede det hele, dog oplevede vi at ikke alt dataen fra mikrofonerne blev leveret via ESP-NOW til aktuatorerne, dette var irriterende, men funktionelt virkede løsningen stadigvæk.</p> <p>Ved højt lydniveau blev vores aktuator aktiveret, hvilket i vores endelig version, kører materiale ned, samt hejse et flag til markering af høj larm.</p> <p>Ved lavt lydniveau kørte vores steppermotor materiale tilbage, samt blev flaget sænket igen.</p> <p>Dataen blev vist på vores hjemmeside, og alt blev gemt i databasen. Til sidst satte vi vores backup computer til, og fik gennemført en backup overførsel af serveren til backup-laptop via vores netværk.</p>

Anbefalede ændringer til næste version af løsningen.

Emne	Nuværende	Anbefales
Anvendelse af ESP-NOW	Ændre fra ESP-NOW	Anvend en mere stabil forbindelse.
Gør produktet mere stabilt	Gå fra breadboard	Til loddet løsning

Disse 2 punkter er noget som hvis vi havde længere tid til projektet godt kunne have ønsket ændringer på, første punkt ændring af protokol til sending mellem ESP32'er, og andet punkt til at lave løsninger på print.

11. Praktisk projektplanlægning og -ledelse

Projektet er planlagt og gennemført med en agil tilgang. Vores vurdering har været, at tidsrammen og arbejdsformen passer bedre til en agil tilgang end til de mere traditionelle metoder som WBS og Gantt, dog lavede vi en WBS lignende Brainstorm i starten af projektet, den ligger i bilag 11.1. Vores projektstyringsværktøj består af elementer fra Scrum. Det primære værktøj har været vores Scrum-board, som fungerede som det overordnede overblik, hvor alle opgaver blev placeret i de klassiske "To Do", "Doing" og "Done" felter mf.

Scrum-boardet gjorde det let at følge driften af projektet, koordinere opgaver og arbejde på tværs af alle teamets medlemmer. Som en fast del af vores proces har vi hver morgen afholdt et Scrum-møde. Det har været en metode til at få opsummeret projektet på individuelt plan i forhold til, hvad vores teammedlemmer hver især enten har opnået, kæmpede med eller haft spørgsmål til - så altså en måde til dagligt at optimere dagens kommende arbejde. Vi har ligeledes hver mandag afholdt et sprint planning meeting for at få klarlagt ugens opgaver, og for at få et overblik over projektet på en mere generel plan. Det har været en god metode til løbende at få evalueret projektet i forhold til projektets tidsramme.

I den indledende fase af vores projekt etablerede vi kontakt til speciallæge Jane Bjerg Groth, som har skrevet en PhD-afhandling omhandlende temaer som vores løsning tager udgangspunkt i. Vi interviewede hende i forhold til at få en bredere viden om problematikkerne, som vi med vores løsning prøver at afhjælpe.

Vores løsning har ikke en klassisk slutbruger, så i stedet for at vi lavede en klassisk interessentanalyse, fungerede interviewet med Jane som en bekræftelse af, at problemstillingen vi arbejder med både er reel og relevant. Derudover har vores fleksible tilgang virket efter hensigten og vi er ikke stødt på problemer med kommunikation, udfordringer med fordeling af arbejdsopgaver eller andre strukturelle problemer i forhold til vores arbejdsproces.

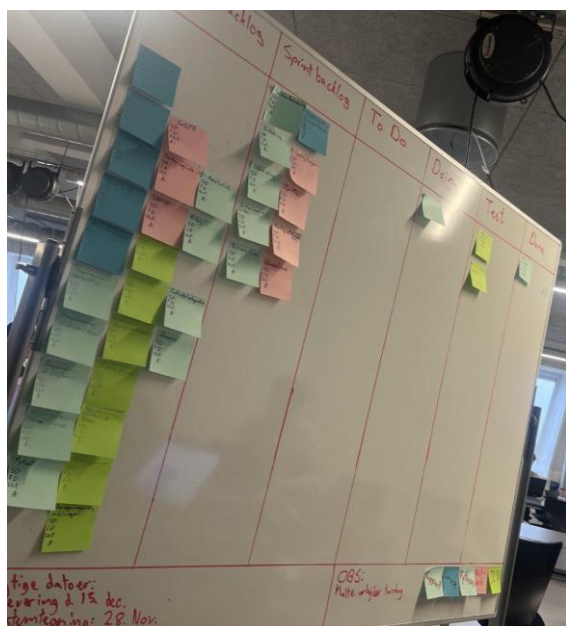
11.1. Scrum

I dette projekt har vi gjort brug af scrum som vores agil projektplanlægnings værktøj.

Denne blev valgt da alle i gruppen havde gode erfaringer med brugen af dette værktøj i tidligere projekter, samt da gruppen valgte at gøre brug af et fysisk scrum board gav det god mening at bruge dette værktøj.

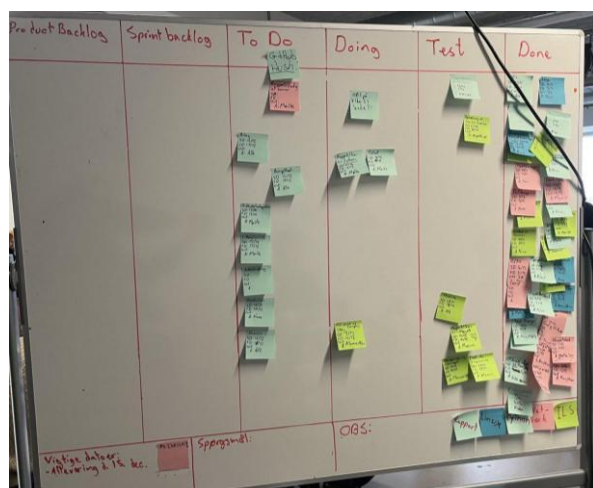
Her er et eksempel på hvordan vores scrumboard ser ud.

Figur 11.1: Billede af Scrum-board i den tidlige fase af projektet



Her ser man hvordan vi bruger vores sprint backlog, i starten af vores sprint til at ligge, de opgaver fra product backlog over som vi skal udføre i gennem ugen.

Figur 11.2: Billede af Scrum-board fra slutningen af projektet



Sådan ser vores scrumboard ud, den sidste uge inden afleveringen, som man kan se, har dette været brugt flittigt, og været en god hjælp til at huske opgaverne.

andet eksempel på vores brug af scrum board findes i bilag 11.2.

12. Konklusion

Projektet har haft til formål at undersøge og identificere støjgener i forskellige arbejdsmiljøer. Fokus har været på at finde ud af, hvordan vi ved hjælp af IoT-baserede tekniske løsninger kunne være med til at sætte fokus på- og løse problemet. Da vi via vores problemformulering konstaterede at selv små støjniveauer i åbne kontorlandskaber kan hæmme koncentrationsevnen og på lang sigt bl.a. give stress. Det giver negative omkostninger for arbejdspladsen og den individuelle medarbejder, i form af forværret og evt. nedsat arbejdsevne og produktivitet. Vi har været opsat på at finde en datadrevet løsning, som kan forbedre lydniveauet og dermed arbejdsglæden og produktiviteten i det enkelte arbejdsmiljø.

Vi har udviklet et system, som via en mikrofon måler, lydniveauet i et givent lokale. Den indsamlede data overføres via vores netværk til vores server. Her lagres dataen i en PostgreSQL-database. På en hjemmeside kan vi se dataene og få et visuelt overblik i realtid. På den måde kan vi over en periode få indblik i støjtendenserne og på et dokumenteret grundlag vurdere støjniveauet.

Løsningen tilbyder, at man i lokalet, hvor lydniveauet måles, kan tilkoble fysiske motor enheder. Aktuatorerne findes som enten et flag viser som støjniveauet, så opmærksomheden øges på den generelle støj. Eller man kan tilvælge et system, hvor lyddæmpende loftsplader bliver sænket ned fra loftet, hvis en predefineret lydgrænse overskrides - på den måde automatiserer man at forbedre lyd klimaet når støjniveauet bliver uhensigtsmæssigt højt.

Vores løsning har til hensigt at vise, hvordan man kan måle og indsamle støjdata og bruge den viden til at forbedre det indendørs lydklima - det er vi lykkedes med. Vores accept test er gennemført til et tilfredsstillende niveau. Vores nuværende produkt er et godt grundlag for, hvordan man kan tage første skridt mod et forbedret lydklima på en given arbejdsplads og det giver rig mulighed for at videreudvikle til endnu mere automatiserede og effektive løsninger.

13. Projektforløbet

Da vi er en helt ny gruppe, som er slået sammen af to tidligere eksisterende grupper, var det svært at vide, hvad man skulle forvente af nye dynamikker som evt. kunne opstå.

Vi var bevidst om, at det var vigtigt, at vi genbesøgte egne forventninger og den nye gruppes samlede forventninger, så vi på den måde kunne få lavet en ny ordentlig forventningsafstemning. Vi var alle enige om, hvilken måde vi skulle projektstyre vores forløb på. Vi har taget det seriøst at holde os til vores plan og stole på vores planlægning gennem hele forløbet. Det har medvirket til, at vi ingen kontroverser har haft, og at vi har haft et meget gnidningsfrit forløb i forhold til alt det kommunikative.

Alle teamets medlemmer har været gode til at give plads, lytte og til at tage sin plads på de rette tidspunkter. Det kommer tilbage til den gode plan og forventningsafstemning, at der er ro i gruppen til at alle føler sig så tilpas at man bidrager med egne evner så godt man kan og udover det hjælper sit team der, hvor man kan se at der er brug for det.

Alt i alt er det svært at se, hvor man kan optimere det kommunikative. Der er selvfølgelig altid noget man kan forbedre - pt. kan vi bare ikke få øje på det.

14. Perspektivering

En klar udvidelsesmulighed for vores nuværende løsning er, at udvide den til at kunne lokalisere bestemte støjfrekvenser og ikke bare som nu, målingen af selve lydniveauet. En mere præcis lokalisering af, hvor støj kommer fra, vil kunne være med til enten at sætte ind overfor personer eller enheder som støjer. Så altså en generel mulighed for at slå ned på de specifikke "støjskabere". Hvis man på baggrund af en mere præcis lokalisering af, hvilken støj der er til gene, vil man også kunne præcisere og udvikle specifikke løsninger til den enkelte situation. Man kan forestille sig, at man rykker ud med en lydlokaliseringsenhed og foretager en analyse og derefter en vurdering af bedst mulige afhjælpning af de støjgener, som kunden har. På den måde kan vi udvide vores forretningspotentiale og sælge specialiserede løsninger til mange forskellige typer af kunder. Et standardiseret og enkelt design kan hjælpe salgsmæssigt og æstetisk. Hvis vi vælger at lave et PCB af vores forskellige kredsløb, får vi muligheden for at lave en enkelt kasse som totalløsning, til hurtig og nem installation. Udvidelser i forhold til forelæsninger, speaker events osv., ville være en mulighed, hvis der forbindes fra den mikrofon som taleren bruger, og til en dB sensor bagerst i lokalet, for at sikre at alle i et givent lokale eller sal kan høre, hvad der bliver fortalt.

15. Litteraturliste

- Arbejdsmiljø, B. F. (11. 11 2024). *godtarbejdsmiljo.dk*. Hentet fra <https://www.godtarbejdsmiljo.dk/stoej-lys-luft/stoej-paa-arbejdspladsen/stoej-paa-hospitaler/case-stoejniveauet-saenkes-paa-opvaagningen>
- Arbejdsmiljø, B. F. (20. 2 2024). *godtarbejdsmiljo.dk*. Hentet fra <https://www.godtarbejdsmiljo.dk/stoej-lys-luft/stoej-paa-arbejdspladsen/stoej-paa-hospitaler/viden-om-stoej-paa-hospitaler>
- Arbejdsmiljø, B. F. (16. 5 2025). *godtarbejdsmiljo.dk*. Hentet fra <https://www.godtarbejdsmiljo.dk/stoej-lys-luft/stoej-paa-arbejdspladsen/stoej-og-forstyrrelser-i-det-store-kontor/stoej-i-storrumskontoret>
- Arbejdstilsynet. (u.d.). *at.dk*. Hentet fra <https://at.dk/arbejdsmiljoe-i-tal/national-overvaagning-af-arbejdsmiljoeet-blandt-loenmodtagere/arbejdsmiljoe-og-helbred-2012-2018/>
- Birdie. (u.d.). Hentet fra <https://www.birdie.design/da>
- Bjerg Groth, J. (6. 10 2020). *Rigshospitalet.dk*. Hentet fra <https://www.rigshospitalet.dk/afdelinger-og-klinikker/hovedorto/oere-naese-halskirurgi/forskning/forskning-i-oeresygdomme/afsluttede-ph.d.-projekter/Sider/stoej-induceret-hoeretab.aspx>
- Bolls. (u.d.). Hentet fra <https://bolls.dk/>
- Castellum. (2024). Fremtidens arbejdsliv– en årlig rapport om kontortendenser. *chrome-extension://efaidnbmninnbpcajpcglclefindmkaj/https://www.castellum.dk/siteassets/documents/find-ledige-lokaler/rad-og-trends/fremtidens-arbejdsliv/fremtidens-arbejdsliv-2024.pdf?utm_source=chatgpt.com*.
- Electronics, R. (u.d.). *chrome-extension://efaidnbmninnbpcajpcglclefindmkaj/https://www.rajguruelectronics.com/Product/1467/28BYJ-48%20-%205V%20Stepper%20Motor.pdf*. *Rajguru Electronics*.
- Erika Martínez Cantón, A., Gonzalez, T., & I. Ibarra-Zarate, D. (December 2019). Noise levels analysis based on sensorial perception as a strategy to boost critical thinking. *ResearchGate*.
- Instruments, T. (u.d.). *chrome-extension://efaidnbmninnbpcajpcglclefindmkaj/https://www.ti.com/lit/ds/symlink/uln2003a.pdf?ts=1765860102473*. *ULN200x, ULQ200x High-Voltage, High-Current Darlington Transistor Arrays*.
- IvenSense. (2014). *chrome-extension://efaidnbmninnbpcajpcglclefindmkaj/https://invensense.tdk.com/wp-content/uploads/2015/02/INMP441.pdf*. *INMP441: Omnidirectional Microphone with Bottom Port and I2S Digital Output*.
- mentech. (u.d.). Hentet fra <https://ek.itslearning.com/plans/courses/7233/plan/130365/element/1425536?BackDestination=0&BackData=%7B%22BackDestination%22%3A%221%22%7D&planner2-sb-collapsed=false>

MG90S. (u.d.). chrome-

extension://efaidnbmnnnibpcajpcgicfindmkaj/https://www.electronicoscaldas.com/datasheet/MG90S_Tower-Pro.pdf. *G90s: Metal Gear Servo*.

Miljøstyrelsen. (u.d.). *Faktaark: Elektrisk og elektronisk udstyr (RoHS-bekendtgørelsen)*. Hentet fra <https://mst.dk/erhverv/sikker-kemi/kemikalier/regler-og-handlingsplaner/faktaark-om-kemikalierreglerne/faktaark-elektrisk-og-elektronisk-udstyr-rohs-bekendtgørelsen>

Perrin Jegen, N., & Chevret, P. (6. 5 2018). Effect of noise on comfort in open-plan offices: application of an assessment questionnaire. *National Library of Medicine*.

Retsinformation. (u.d.). *retsinformation.dk*. Hentet fra <https://www.retsinformation.dk/eli/lt/2019/1107>

SMVdanmark. (1. 12 2022). . Hentet fra Ritzau: <https://via.ritzau.dk/pressemeddelelse/13665594/ny-analyse-antallet-af-akademikere-eksploderer-i-de-kommende-ar?publisherId=13559667>

Soundear. (u.d.). Hentet fra <https://soundear.dk/>