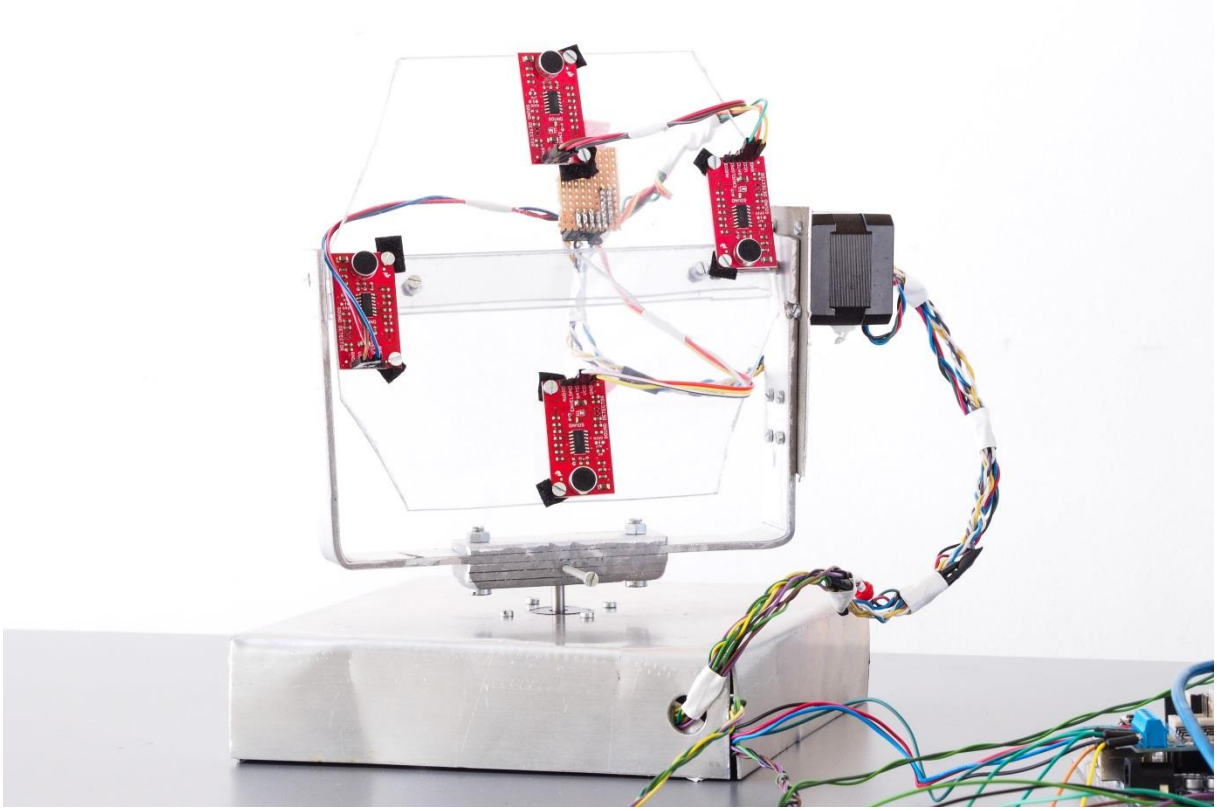
PROSJEKT I DIGITALE SYSTEMER II

Lydlokalisering



Av:

Jonas Skarstein Waaler

Daniel Gusland

INNHOLD

FORORD

I denne oppgavebesvarelsen vil vi beskrive prosessen hvor vi forsøkte å gjenskape "interaural time difference," som er en av metodene øret bruker for å lokalisere lyd. Vi skal også konstruere en enhet som kan rette seg mot lydkilden.

Takk til Halvard Fredly, John Sverre Dischington Hanssen og Hilde Hemmer. For god hjelp og støtte.

SAMMENDRAG

Hensikten med prosjektet var å implementere en gitt problemstilling i en konstruksjon som var basert på elektriske og digitale enheter. Vårt mål ble å få en enhet til å peke på en lydkilde. En av hovedprestasjonene har vært å klare å registrere tidsdifferansen mellom to punkter med hensyn til lyden, noe som ble gjort ved hjelp av et arduinokort og to mikrofoner. Videre har vi fått enheten til å rette seg mot lydkilden, både horisontalt og vertikalt, men da ved å sette sammen to enheter av samme type. Enheten viste seg å peke på lydkilden fire av fem ganger.

INNLEDNING

PROBLEMSTILLING OG FORMÅL

Lage en elektronisk enhet som kan lokalisere lyd og rette seg mot lydkilden.

OMFANG OG BEGRENSNINGER

Ut ifra prosjektets tidshorisont var det klart for oss helt fra starten at enheten ikke ville klare å lokalisere lyd med en skyhøy presisjon. Basert på dette ville en enhet som klarte å indikere lydens posisjon i to plan være prosjektets mål.

TEORI

Vi mennesker er relativt flinke til å lokalisere lyder, vi bruker i hovedsak tre forskjellige teknikker. To av disse teknikkene var for avanserte til å implementere i vårt prosjekt. Den siste er derimot relativt enkel i teorien, selv om praksis ofte viser seg å være en annen sak.

INERAURAL LEVEL DIFFERENCE (ILD)

Denne teknikken baserer seg på at lydintensiteten gradvis blir mindre etterhvert som lyden går gjennom luft. Forskjellene over store distanser er meget merkbare, men ved korte distanser vil de i beste fall være minimale. For vår applikasjon med bruk av arduino vil disse forskjellene ikke være målbare innenfor en avstand som er akseptable å ha mellom to eller flere mikrofoner.

Denne teknikken er også veldig avhengig av bølgelengde, da lange bølger rett og slett vil passere hodet uten noe merkbar forskjell i nivå. Menneske bruker likevel denne teknikken når det kommer til høye frekvenser fordi menneskehode lager veldig karakteristiske lydskygger(HRTF).

HEAD-RELATED TRANSFER FUNCTION (HRTF)

Formen på menneskehode er veldig karakteristisk, noe som gjør mye med hvordan lyden passerer. Dette gjør at lyden fra forskjellige vinkler vil nå øret på forskjellige måter, da mest med tanke på lydnivå. Lyder ovenfra vil blant annet høres annerledes ut enn lyder nedenfra. Dersom vi skulle brukt dette i vårt prosjekt måtte vi konstruert en avansert geometrisk form av et materiale som er relativt akustisk dødt. Deretter ville vi måtte foreta mange eksempelmålinger ved nøyaktige vinkler og lage en modell for vårt "hode". Dette ville vært svært tidkrevende og med arduino ville det vært svært vanskelig å gjennomføre.

INTERAURAL TIME DIFFERENCE (ITD)

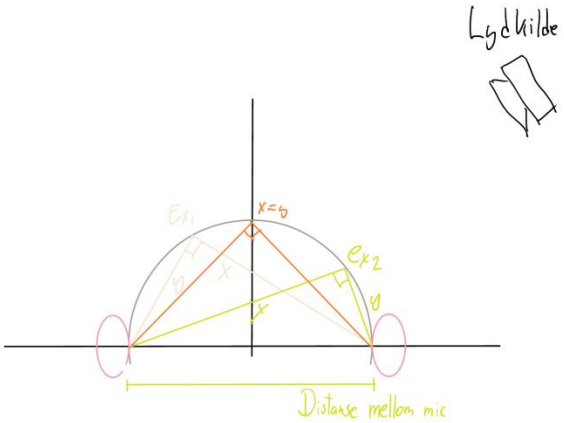
Dette er den metoden vi benytter oss av, da den er den klart letteste å gjennomføre. Denne metoden benytter seg av lydens relativt lave forplantningshastighet i luft. Det vil si at dersom lyden kommer fra en av sidene, vil det være en målbar tidsforskjell mellom to mikrofoner. Dersom disse mikrofonene er plassert like langt fra hverandre som et vanlig hode vil denne forskjellen være omtrent 600 millisekunder. Dette kommer vi mer tilbake til senere. Denne teknikken har også noen feilkilder, men dette er ikke noe stort problem for vår applikasjon.

BEREGNINGER OG FORMLER

Distansen vi har brukt mellom mikrofonene er 15cm. Dette er noe mindre enn et vanlig hode. Ved hjelp av lydhastigheten kan vi ganske enkelt regne ut hvor lang tid lyden vil bruke fra en mikrofon til en annen, dette vil være tilfelle dersom lyden kommer fra 90 grader på hver side:

$$Tidsforskjell = \frac{0.15\ m}{340\frac{m}{s}} \approx 440\ \mu s$$

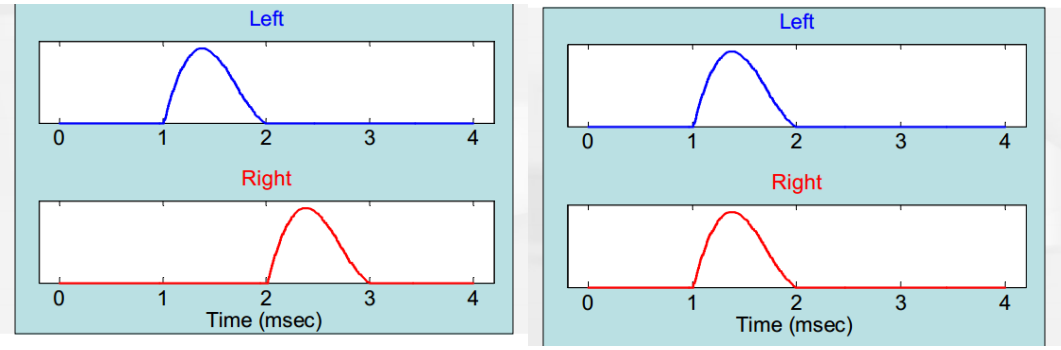
FIGUR 1: FIGUR SOM VISER DEN RELATIVT ENKLE TRIGONOMETRIEN



Som vist i figuren over vil dette danne en rettvinklet trekant, ut ifra dette kan vi utlede en formel for vinkelen lyden kommer fra ut ifra tidsforskjellen mellom de to mikrofonene. Formelen vi utledet ble:

$\phi = \arcsin\left(\frac{\Delta t \times c}{d}\right)$, *Hvor* Δt *er tidforskjellen,* c *lyshastigheten og* d *distansen.*

Under kan du se en figur som illustrerer hvordan dette kan se ut i praksis:

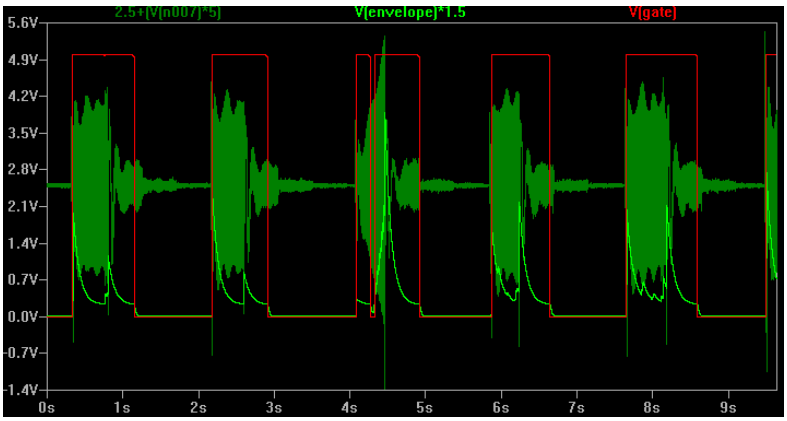


FIGUR 2: FIGUR SOM FORKLARER TIDSFORSKJELL (WASHINGTON UNIVERSITY)

HVORDAN GJENKJENNE LYD

Det er mange forskjellige måter man kan gjenkjenne lyd på. Dersom man skal kunne gjenkjenne lyd ved en spesiell frekvens eller takt – som for eksempel tale – vil det være en meget avansert oppgave. Derfor valgte vi å gjenkjenne en skarp og høy lyd, det være seg et klapp, et knips, dør som smeller eller liknende. På bakgrunn av dette vil det være relativt enkelt å gjenkjenne lyd. Det kan gjøres ved å sette en såkalt «gate», altså et lydnivå som lyden må være over. Når lyden går over dette vil en peaking indikator sette en port høy. Dette vil si at mikrofonen gir oss 0V helt til lyden blir over en viss verdi – da vil den gi 5V. På denne måten gjenkjenner vi den første toppen som vil nås. Det illustreres i figuren under:

FIGUR 3: FIGUR SOM VISER HVORDAN EN "GATE" VIL FUNGERE, HVOR DEN RØDE KURVEN ER GATESIGNALET.(SPARKFUN)



TEKNISKE KRAV

Det neste vi måtte undersøke var om arduino var i stand til å måle tidsforskjellen mellom mikrofonene. Klokkefrekvensen til Arduino er på 16 MHz. Det vil si at én operasjon tar cirka:

$$T = \frac{1}{16MHz} = 6{,}25 \times 10^{-8}s \rightarrow Antall\ pulser = 6{,}25 \times 10^{-8} \times 70 = 437{,}5\ \mu s$$

Ut ifra dette kan vi set at vi teoretisk sett skal kunne foreta 70 målinger innenfor dette område – som for oss vil gi en akseptabel presisjon. Vi måtte også foreta en måling per mikrofon, altså to for hvert plan. Hver mikrofon vil halvere vår nøyaktighet. Ved to mikrofoner blir det da:

$$\frac{70}{2} = 35\ målinger\ innenfor\ område$$

Likevel viste det seg at arduinos ADC er såpass treg at den bruker omtrent $100\mu s$ på en måling.

Noe som ville gi oss så lite som:

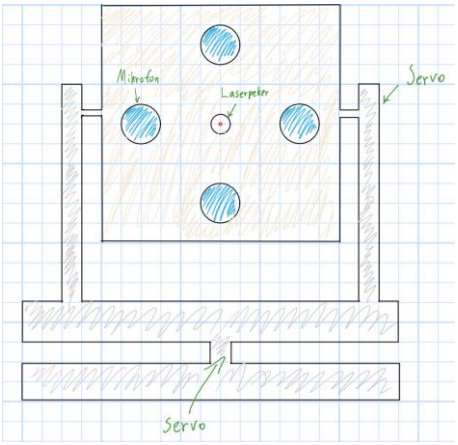
$$100 \times 2 = 200 \rightarrow \frac{400}{200} = 2\ målinger\ innenfor\ område$$

Med andre ord måtte vi finne en annen måte å løse det på enn arduinoens analoge innganger.

Vi måtte med andre ord finne en måte å løse dette og allikevel opprettholde Arduinos klokkehastighet.

ANDRE TEKNISKE KRAV

I tillegg til å kunne gjenkjenne og plassere lyd skulle vår enhet også kunne vende seg mot lydkilden. Dette i seg selv viste seg å være en utfordring. Enheten måtte kunne snu seg 360° både i horisontal og vertikal retning. Enheten måtte også kunne styres på en relativt enkel måte ved hjelp av arduino. De første tegningene for denne enheten er følgende:



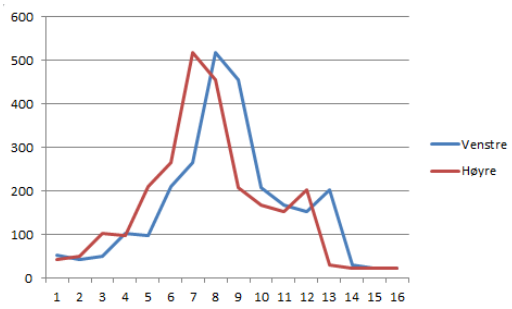
FIGUR 4: FØRSTE TEGNINGEN AV PROTOTYPEN

TESTSTRATEGI

Den første delen av prosjektet gikk med på å sjekke om prosjektet var mulig. Vi utførte noen tester. Den første gikk ut på å ta opp lyd på en PC fra forskjellige vinkler for så å plotte resultatene. Dette viste at prosjektet skulle være mulig, som vist i figurene under:

FIGUR 5: VISER DET FØRSTE TESTOPPSETTET MED TO MIKROFONER, LYDKILDE OG VINKLER MERKET.

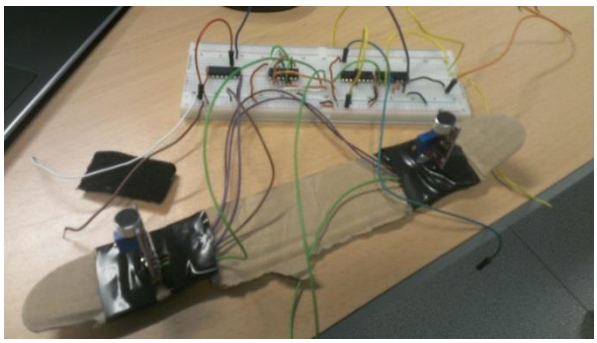
FIGUR 4: GRAF SOM VISER DATA TATT OPP MED ARDUINO, EKSPORTERT TIL PC OG DERETTER PLOTTET I EXCEL



Deretter gikk vi nærmere inn på dette ved å forsøke å måle ved hjelp av Arduino, med varierende resultat.

Etter dette konstruerte vi en binærteller med en klokke på 90MHz, for å være sikre på at presisjonen var god nok. Det viste seg at de varierende resultatene bestod. På bakgrunn av dette kunne vi konkludere med at det var mikrofonene eller lyden det var noe i veien med. Vi undersøkte deretter både kildelyden og kretsene nærmere med oscilloskop. Dette viste at det var mikrofonene som ga upresise resultat. Etter vi byttet ut disse med en annen type mikrofoner ble resultatene både bedre og enheten mer presis.

SYSTEMLØSNING



FIGUR 6 TIDLIG I TESTFASEN

Da vi skulle sette opp systemet begynte vi først å se på hvilke prosedyrer og funksjoner enheten måtte utføre for å kunne gjennomføre oppgaven:

- Les begge mikrofoner ved gate
- Registrer dette digitalt
- Regn ut tidsforskjell
- Regn ut hvor mange radianer lyden kommer fra
- Regn om radianer til grader
- Flytt motoren antall grader
- Vente litt

Dette kan igjen deles opp følgende hovedoppgaver

1. Programbaserte oppgaver:
   - Registrer mikrofonsignalene digitalt
   - Utregninger
   - Signal til motor
   - Pause

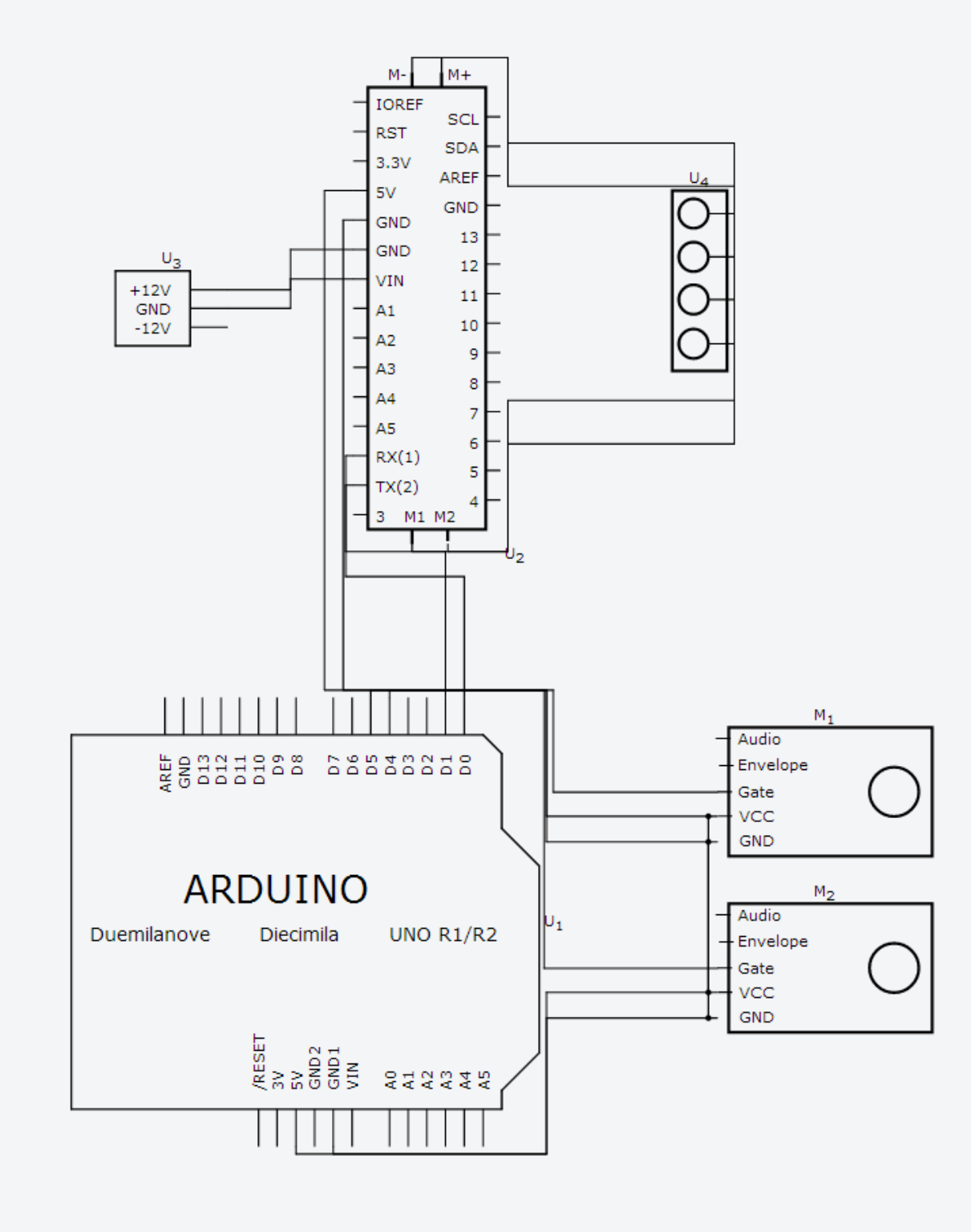
2. Andre oppgaver
   - Registrer lydnivå og send gate
   - Beveg enhet

For fullstendig oversikt over utstyr, se utstyrsliste.

ANDRE OPPGAVER

Vi brukte to mikrofoner av typen SEN-12642, også kalt «Sound Detector». Disse har en innebygd analog gate, noe som vil si at når mikrofonen oppdager lyd så sender de et høyt signal. Det at gaten er analog er viktig da det ikke involverer noe sampling og vi får derav ikke noe informasjonstap. For å bevege enheten valgte vi å bruke stepper motorer som er koblet til et Motor Shield tilpasset arduino.

Koblingsskjema for de to kretsene:



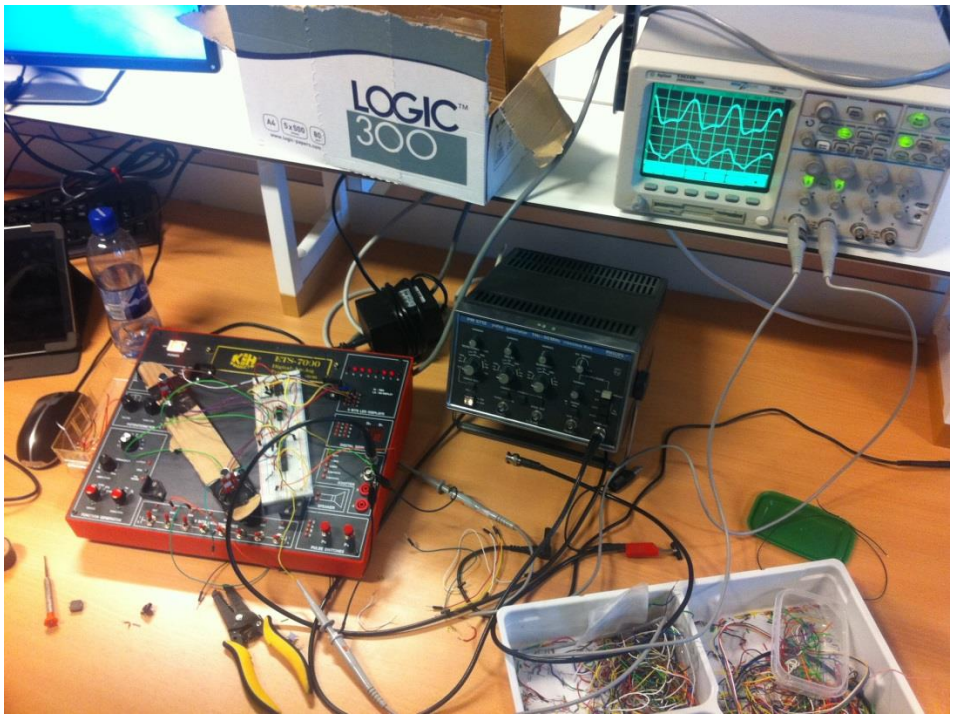
FIGUR 7: KOBLINGSSKJEMA FOR KRETSENE

SYSTEMPRØVING OG RESULTATER

Under testing av systemløsningen fant vi fort ut hva som ble utfordrende.

ADCEN TIL ARDUINO

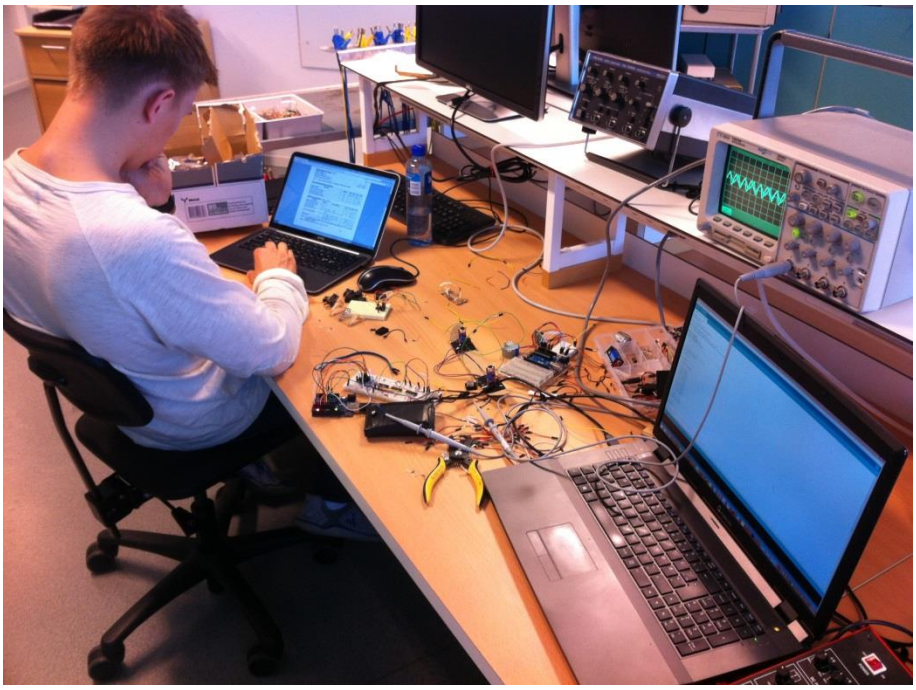
Arduinos ADC viste seg å være for treg. ADCen brukte omtrent 100µs per måling. Dette ga oss omtrent èn måling i mellom ved 180 grader, noe som ble alt for lite. Vi forsøkte først å endre registerfilene for å øke hastigheten på ADCen. Dette ga oss en 10 ganger raskere ADC, men likevel ble den for treg.



FIGUR 8: TESTING AV ADCEN TIL ARDUINO

KLOKKEHASTIGHETEN TIL ARDUINO

Klokkehastigheten til Arduino er som tidligere nevnt på 16MHz. Dersom vi skulle foreta målinger på alle fire mikrofonene med èn Arduino ville dette medføre en vesentlig forringning av måleresultatene. Vi konstruerte derfor en binærteller med klokkehastighet på 90Mhz, som ville økt presisjonen, men konkluderte med at dette ikke var nødvendig. Derfor løste vi dette med å bruke èn Arduino per akse. Vi leser også av porten direkte, istedet for å bruke Arduinos DigitalRead, for å forhindre overhead i programvaren.



FIGUR 9: KONSTRUKSJON AV BINÆRTELLER

TYNGDE

Selve vekten til enheten viste seg å være en utfordring. Selv om enheten ble bygget i aluminium klarte ikke motorene å kjøre kretsen på vanlig 5V via USB. I tillegg trakk motorene så mye at de senket spenningen i hele kretsen, noe som var ugunstig da mikrofonene ga ut en logisk 1 ved oppstart. Dette ble løst ved å kjøre de to arduinoene og mikrofonene i en separat 5V krets. På MotorShieldene fjernet vi kortslutningen på «Vin connect», dette gjør at motorene utelukkende kjører på ekstern strøm. Disse får strøm fra en ekstern 12V kilde.

MIKROFONER MED GATE

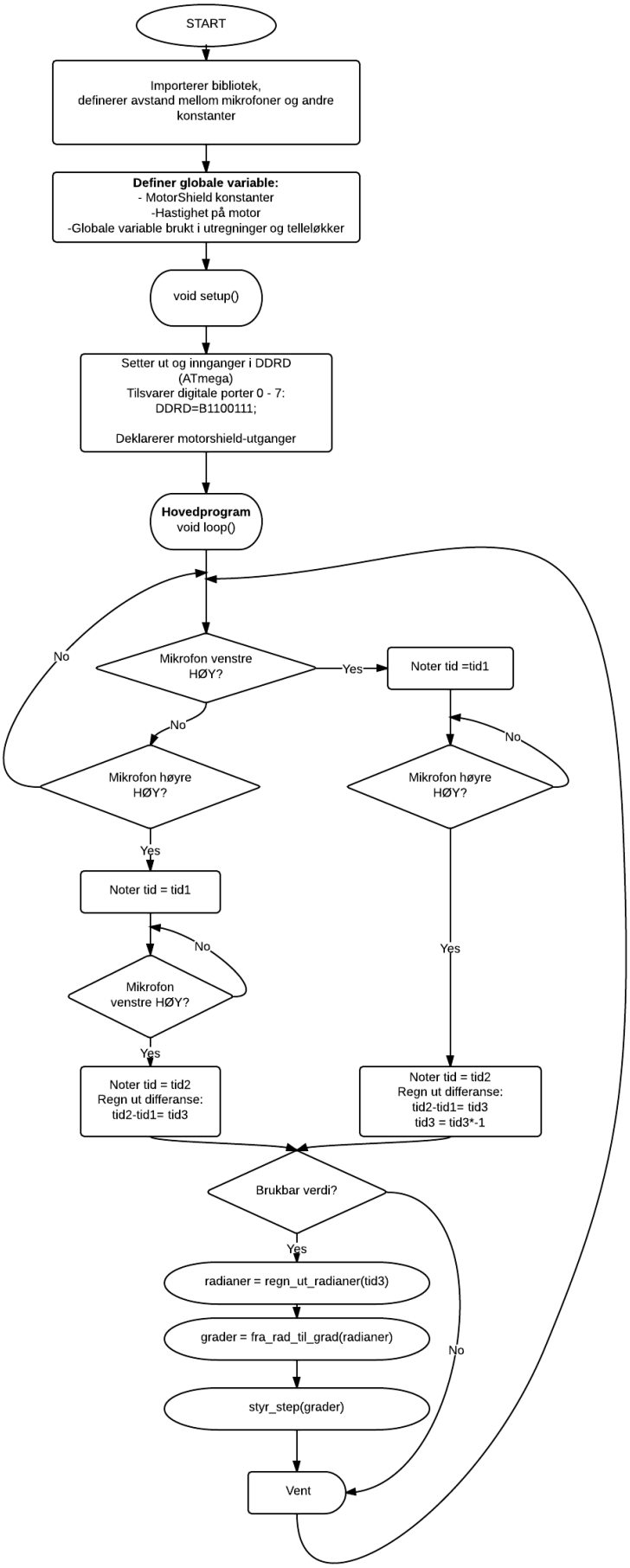
En annen utfordring var å få «gaten» på alle mikrofonene til å gi 5V ved samme lydintensitet. Variasjoner her vil medføre betydelige feilmålinger. Vi forsøkte først med en type mikrofoner fra EBAY, men variasjonen ble for stor. Deretter gikk vi over til mikrofoner av typen SEN-12642, som fungerte vesentlig bedre. Allikevel slet vi med at èn av mikrofonene trigger på ujevne tidspunkt.

FEILMÅLINGER

På grunn av triggermekanismen til mikrofonene får vi til tider feilmålinger. Triggerinformasjonen kommer til Arduino fra mikrofonene. Arduino har dermed ikke noe informasjon of hvorfor feilmålingen kommer. Vårt eneste valg med nåværende løsning er da kun å forkaste disse målingene.

PROGRAMBASERTE OPPGAVER

Programmer er satt opp og programmert på følgende måte (se figur 5). Sirkler representerer start og slutt på funksjoner, trekanter representerer en test, mens firkanter prosesser og annet. Se for øvrig *vedlegg 1* for programmet.



FIGUR 5 FLYTSKJEMA HOVEDPROGRAM

For å kunne benytte oss av arduinos klokkehastighet og ikke bli lammet av arduinoens trege konverteringen i de digitale portene, benytter vi oss av portmanipulasjon, noe som vil si at man programmerer ATmega kortet direkte. Mikrofonenes gateutgang er koblet til de digitale portene 4 og 5 på arduinoen, og programmet leter hele tiden etter endringer her. Dette gjøres ved å hele tiden sjekke PIND «logisk og ikke lik null» mot bitmønsteret som representerer port 4 og 5 på arduinoen. PIND er de digitale portene, 0 – 7, på arduinoen.

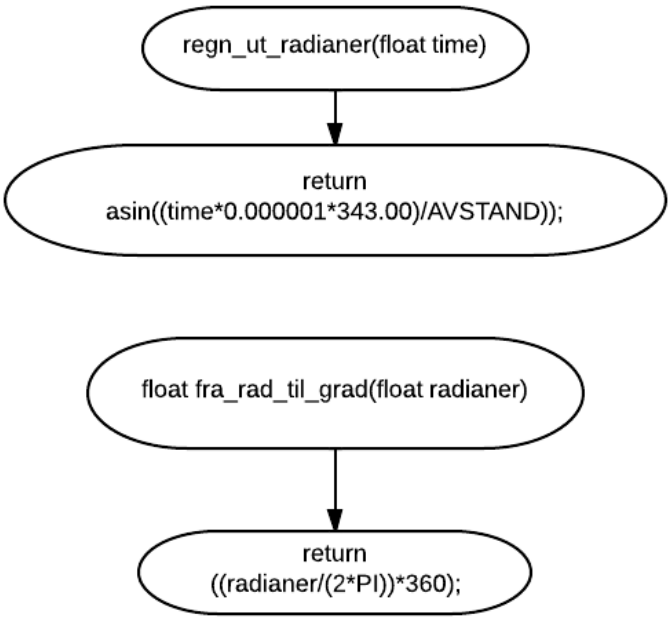
Om kortet merker en endring i en av portene noteres tiden, og programmet venter til neste port endres. Slik kan man regne ut tidsdifferansen $\Delta t = t_1 - t_0$. Hvilken mikrofon som sender signal først avgjør om lyden kommer fra venstre eller høyre. Dette markeres ved å la time3 være negativt hvis den kommer fra venstre, og positiv hvis den kommer fra høyre.

Programmet sjekker så om dette er en realistisk tid: hvis målingen gir absoluttverdi over 1000 mikrosekunder er det ikke tvil om at dette er en ugyldig måling. Dette kan forekomme da en mikrofon kan sende signal, for så at det ikke registreres av den andre mikrofonen.

Når programmet har fått inn en gyldig måling må tidsdifferansen settes inn i formelen:

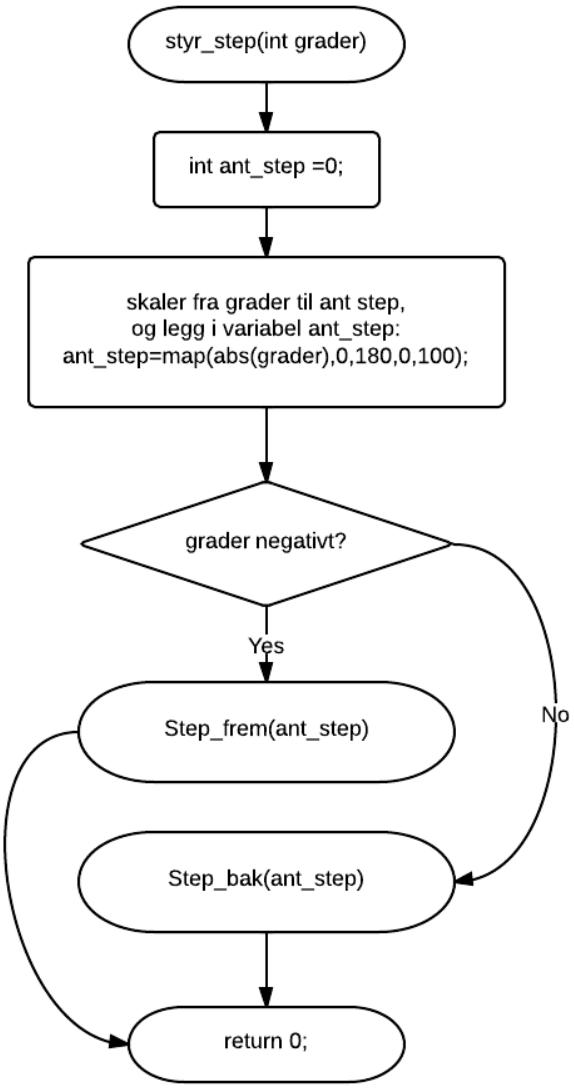
$\phi = \arcsin\left(\frac{\Delta t \times c}{d}\right)$, hvor c = 344 og d = 0.15

Dette gjøres ved hjelp av funksjonen regn_ut_radianer (se figur 6). Den returnerer antall radianer lyden kommer fra. Funksjonen fra_rad_til_grad regner om radianer til grader.



FIGUR 6 FLYTSKJEMA

Neste steg blir å styre step motoren for å få enheten til å peke på lydkilden. Funksjonen styr_Step gjør dette (se figur 7).



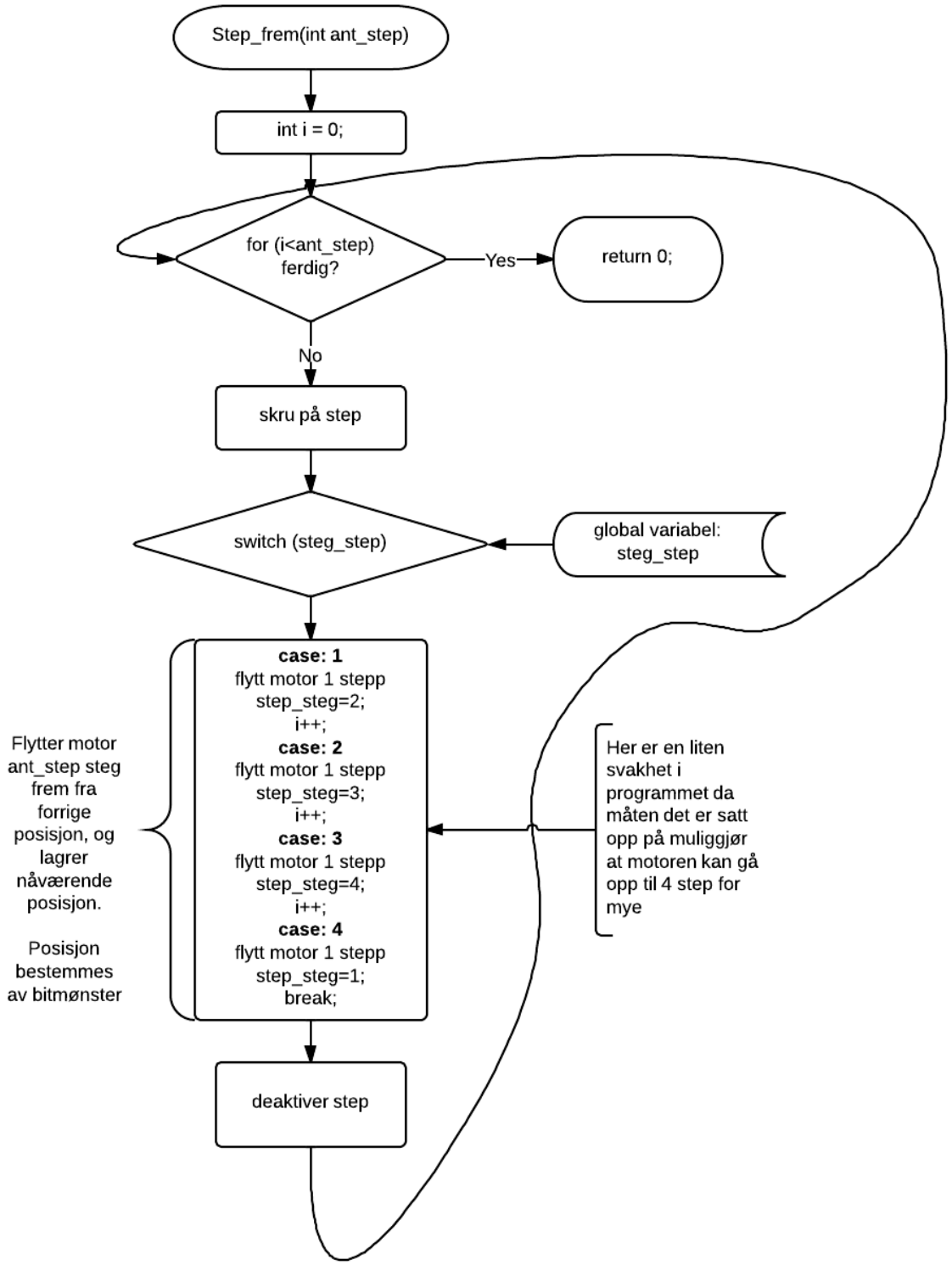
FIGUR 7 FLYTSKJEMA

Funksjonen styr_step er satt sammen av flere funksjoner. Disse funksjonene er henholdsvis Step_frem og step_bak. Før motoren kan styres framover eller bakover må antall grader skaleres til antall step. Stepmotoren vi har bruker 200 step på 360 grader, noe som vil si:

$$\frac{360grad}{200step} = 1.8grad/step$$

Denne skaleringen gjøres ved hjelp av «map». Map skalerer en skala fra a – b til x – y. Funksjonen sjekker så om mottatt verdi «grader» er negativt eller positivt. Om negativt, så skal den gå fremover, om positivt, bakover.

Step_frem og Step_bak er omtrent to like funksjoner. Forskjellen er at de bytter bitmønster. Siden vi har brukt motorshield er styringen av stepmotorer i vårt tilfelle litt spesiell (se figur 8).



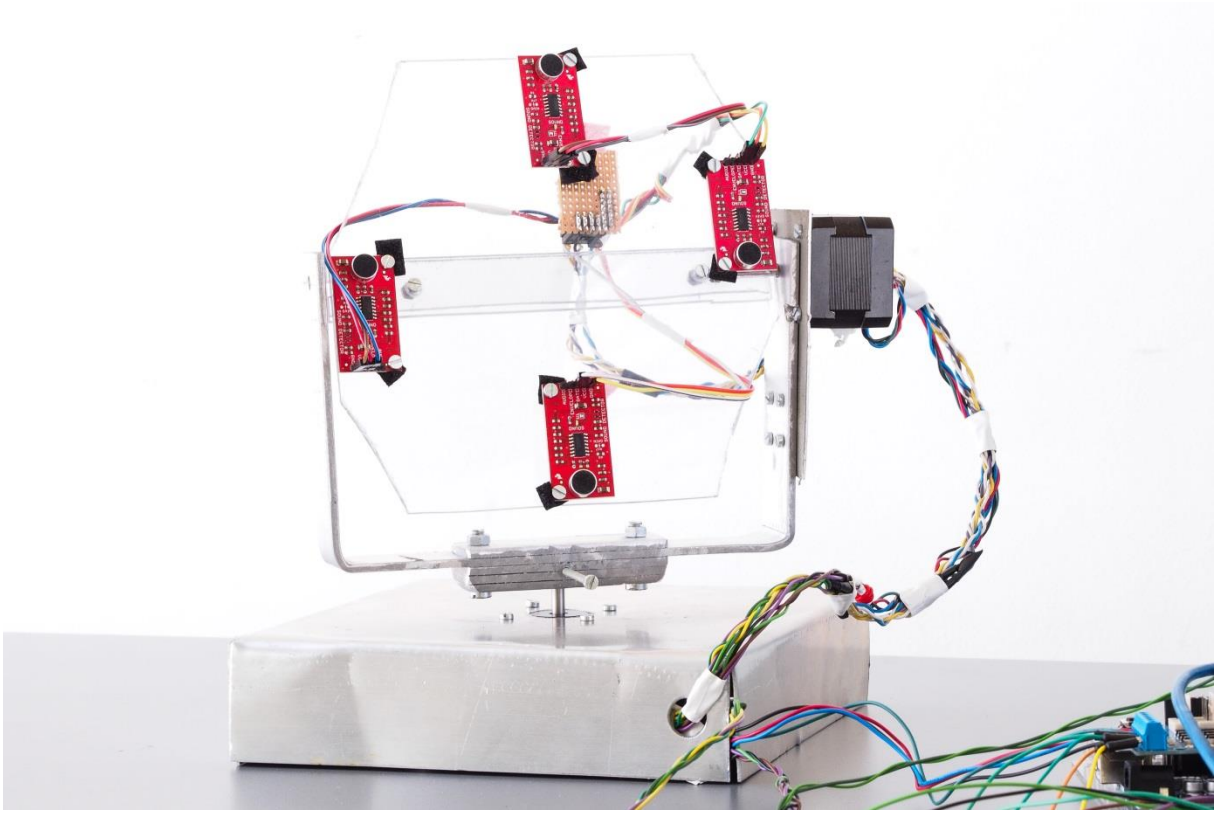
FIGUR 8 FLYTSKJEMA

Funksjonen step_frem flytter motoren ett steg frem ved å skyve bitmønsteret ett steg til høyre. Step_bak skyver bitmønsteret ett steg til venstre. Funksjonene tar vare på hvor de er i

bitmønsteret slik at man kan bestemme antall step og hoppe inn i funksjonene uansett hvor man er i bitmønsteret.

KONSTRUKSJON AV FYSISK ENHET

Den fysiske enheten ble konstruert i pleksiglass og aluminium. Pleksiglass for utseende, samt isolerende egenskaper. Aluminium valgte vi fordi det er enkelt å forme og lett i vekt, slik at belastningen på stepmotoeren skulle bli lav.



DISKUSJON OG KONKLUSJON

Resultatene fra prosjektet er gode. Enheten klarer å lokalisere lyd i to plan og rette seg mot lydkilden. Enheten klarer å gjenkjenne og rette seg mot omtrent ⅘ av våre lyder under testing. Vi laget også en høypresisjons binærteller underveis, men dette la vi fra oss for å gjøre prosjektet litt enklere. Presisjonen er bedre enn forhåpet, men kan likevel bli vesentlig forbedret. Her er noen forslag til forbedring:

- Dersom vi hadde brukt en enhet med høyere klokkefrekvens kunne vi fått flere målepunkter og dermed høyere presisjon.
- Vi kunne brukt en rask ADC for å kunne gjenkjenne bølger.

- Implementert fasedeteksjon og matriser.

- Mer avansert algoritme for å gjenkjenne og sammenlikne bølgetopper.

- Brukt mer egnede mikrofoner.

- Brukt alle fire mikrofoner og triangulert signalet.

- Brukt flere mikrofoner i et såkalt “array”.

Bruksområdene for enheten er mange og lite utforskede. Likevel kan vi se for oss eksempler som:

- Automatiske kamerahoder

- Videokonferanse-systemer

- Andre systemer som skal gjenkjenne bølger (radar og liknende)

- Lysbrytere
- Fjernkontroll

- Spillsystemer

- Trafikkameraer

For å konkludere har vi konstruert en prototype som klarer å registrere hvor en høy lyd kommer fra og rette seg mot kilden. Prototypen har tilfredsstillende presisjon for vårt formål.

LITTERATURLISTE

Sparkfun, 2014, *Sound Detector Hookup Guide* [Artikkel] Sparkfun. Tilgjengelig på: https://learn.sparkfun.com/tutorials/sound-detector-hookup-guide/all (Besøkt 30.04.2014)

University of Washington, *Sound Localization in the Auditory Scene*[PDF]. Tilgjengelig på: http://courses.washington.edu/psy333/lecture_pdfs/Week9_Day2.pdf (Besøkt 30.04.2014)

North Carolina State University, *Sound Localization*[Artikkel] NCSU. Tilgjengelig på: http://www.ise.ncsu.edu/kay/msf/sound.htm (Besøkt 30.04.2014)

Fakheredine Keyrouz, 2012 , *Efficient Binural Sound Localization for Humanoid Robots and Telepresence Applications*[PHD]. Tilgjengelig på: http://mediatum.ub.tum.de/doc/648977/648977.pdf

UTSTYRSLISTE

FERDIG ENHET

FYSISK

Type	**Mål**	**Antall**
Aluminiumsplate	3x40 cm	1
Aluminiumsplate	3x8 cm	4
Pleksiglass	17x17 cm	1
Pleksiglass	17x2 cm	2
Maskinskruer og mutter	M3	25
Stålplate	22x22 cm	1
Stålplate	4x9	1

TEKNISK

Type	**Mål**	**Antall**
Mikrofoner (SEN-12642)		4
Steppermotor		2
Arduino		2
MotorShield		2
Maskinskruer og mutter	M3	25
Strømforsyning 5V		1
Strømforsyning 12V		2
Kretskort til fordeling		1

TESTING OG PRODUKSJON

Type	Mål	Antall
Div. Verktøy til utforming av metall		
Oscilloskop		1
Kretskort for øving		4
PC		2
Excel, til plotting		
Frekvensgenerator		1