Høyttaler rapport

# Innledning

I dette prosjektet skal vi lage en smart høyttaler. Denne oppgaven valgte vi å se på som en utfordring til å lage en høyttaler som skal være enkel for brukeren å bruke uten å måtte knote med brytere og knapper for å kontrollere den. Dette har vært tankegangen vår gjennom hele prosjektet. For det ville vært en skam å lage et så fint møbel som denne høyttaleren er hvis brukeren ble frustrert hver gang han skulle gjøre enkle ting som å endre volumet eller sette sangen på pause. Dette er ideen vi har hatt bak dette prosjektet. I denne artikkelen skal vi derfor sette dere inn i prosessen bak utviklingen av denne høyttaleren.

# Strøm forsyning

For å levere strøm til høyttaleren valgte vi å bruke en vanlig Samsung USB lader som vi kobler til via en USB-A han til USB-A han kabel inn i en USB-A port på baksiden av høyttaleren. Grunnen til at vi valgte å bruke en USB lader for å gi strøm til høyttaleren istedenfor batterier er to ting. Den første grunnen er at vi ønsker å ha muligheten til å ha skjermen på når høyttaleren ikke spiller lyd og en Arduino aktiv sånn at man kan skru på høyttaleren med å legge hånden på overflaten av høyttaleren som vil detekteres av en ultralydsensor. Dette ville vært upraktisk med en batteridreven høyttaler fordi det ville tappet batteriene veldig fort å ha høyttaleren i standby konstant. Den andre grunnen er at denne høyttaleren er ment for å stå på et sted. Den er ikke ment for å fraktes rundt og da er det bare en ulempe for brukeren at han eller henne må bytte ut batteriene på høyttaleren når man bare kan ha en kabel i veggen som sikrer at den alltid er klar når man ønsker å bruke den.

Den opprinnelige planen var å bruke USB-A port på høyttaleren. Men når vi fikk delene så vi at Johannes isteden hadde kjøpt en USB-Micro port. Denne porten klarte vi å miste på grunn av at vi så litt på den for å finne en løsning for hvordan vi skulle lodde på kablene ettersom det var veldig litte mellomrom mellom kontaktene. Så må vi ha klart å dytte den ned på gulvet og ikke lagt merke til det. Dette førte til at vi neste gang ikke fant USB-B porten og dermed måtte komme opp med en ny løsning. Da tenkte vi at vi isteden skulle lodde av USB-B porten på en gammel telefon Åsmund hadde hjemme. Dette fungerte ikke fordi loddebolten ikke ble varm nok til å smelte tinnet. Vi er ikke helt sikre på om de festene som holdt porten fast var et annet metall eller ikke men dette førte til at vi måtte finne en annen løsning.

Løsningen på dette problemet ble å skjære opp en gammel USB-A lader sånn at vi fikk ut kretskortet. Så prøvde vi å fjerne alt loddet fra fire tagger som holdt porten fast til kretskortet. Vi fikk av det meste tinnet men det satt alltid litt for mye tinn igjen så vi ikke fikk porten løs. Løsningen på problemet ble å bruke en tang og klippe porten ut av kortet, ettersom dette var et gammelt og sprøtt kretskort.

Når vi skulle lodde på strømkablene klarte vi å knekke av metallstengene som stak ut av porten for å koble seg på USB porten. Dette skjedde fordi pinnene var bøyd nedover og vi trengte å ha dem rett ut for å være sikre på at de ikke ville komme i konflikt med treverket. Noe pinnene ikke tålte. Dette førte til at vi måtte lodde strømkablene direkte på metallplatene som kobler seg på USB-kabelen. Noe som var ganske utfordrende ettersom vi måtte koble oss til helt på tuppen av platene for å ikke komme i veien når man plugger en USB kabel inn i porten. Noe som førte til at vi måtte putte elektrisk teip rundt kablene for så å teipe rundt hele porten og kablene for å sikre at loddingen ikke rykker når vi setter den inn i høyttaleren.

Ettersom USB-kablene vi hadde tilgjengelig på skolen og hjemme var USB-A til USB-Micro måtte vi lage vår egen kabel. Dette løste vi ved å bruke en gammel USB-A mus samt koblingen på en annen USB-A kabel som vi loddet sammen og isolerte med krympeplast.

# Nodemcu

For dette prosjektet har vi valgt å bruke en skjerm for å kunne gi en del informasjon til brukerne. Grunnen til at vi implementerte disse funksjonene er at en høyttaler er et potensielt dyrt møbel som står ganske åpent i et rom. Da er det syn at det ikke skal kunne være i bruk når man ikke vil høre på musikk. En veldig nyttig funksjon da er at den kan gi brukeren informasjon om været. Dette gjør at brukeren kan høre på musikk mens han spiser frokost og ta en raks tit bort på høyttaleren istedenfor å måte ta opp telefonen. Denne skjermen gir også ekstra nytte ved at vi kan vise volumet til høyttaleren.

**Valg av WIFI modul**

Opprinnelig hadde vi tenkt å bruke en ESP-8266-01 istedenfor Nodemcu 1.0. forskjellen er at ESP-8266-01 er en liten WIFI modul som kan festet på en Arduino mens Nodemcu 1.0 er en microkontroller som kan operere selvstendig. Begge disse to er Hilmir sine moduler som vi lånte. Vi tenkte opprinnelig å bruke Softwareserial til å sende kommandoer til ESP-8266-01 og så kontrollere skjermen med Arduinoen. Men det var noe feil med minne på ESP-8266-01 modulen som gjorde at komandosoftwaren som var på modulen ikke fungerte som den skulle. Vi byttet den derfor ut med Nodemcu 1.0 modulen. Ettersom denne kan operere individuelt fant vi ut at den andre Arduinoen var unødvendig og droppet derfor den Arduinoen og kontrollerte skjermen direkte fra Nodemcuen.

**Innhenting av vær data**

For å innhente værdata valgte vi å bruke yr. Dette er fordi yr har gratis værinformasjon som er fritt tilgjengelig for alle å bruke sånn de ønsker. Det er også veldig enkelt å bruke ettersom det er skrevet sånn at man kan få informasjonen om et sted som en XML-fil som man så må lese av.

For å lese av XML filene som vi får fra yr tok vi utgangspunkt i et program laget av Kjartan Michelsen som vi fant på GitHub. Dette programmet var spesielt designet for nettopp å lese av XML filene fra yr så vi måtte bare modifisere det sånn at det passet til vår bruk. Det gikk hovedsakelig ut på å fjerne den biten av koden som leste av en XML-fil han hentet fra sin egen server for å få lokale temperaturer og fjerne alt som kontrollerte en skjerm sånn at vi kunne skrive våre egne programmer for det.

Når vi var ferdig med å modifisere programmet for å hente værdata gjorde vi det om til en klase med en funksjon som måte kalles på i starten av loopen for å hente værdata. Grunnen til at vi valgte å gjøre det på denne måten er at vi fikk vite på den tiden at skjermen ikke kom til å komme i tide. Ettersom vi fikk en reserve skjerm var ikke dette et stort problem. Men vi ønsket gjerne å ha den store skjermen så da gjorde vi om denne delen til en klasse sånn at vi ville ha muligheten til å modifisere programmet til Nodemcuen uten at vi risikerte å ødelegge noe av koden for vær innhentingene ettersom det da er et lukket program.

**Kommunikasjon mellom Arduinoen og Nodemcuen**

Ettersom vi tenkte å vise frem volumet på skjermen og synkronisere volumet mellom appen og Arduinoen måtte vi kommunisere mellom Arduinoen og Nodemcuen. Dette løste vi ved å bruke biblioteket softwareserial. Grunnen til at vi valgte å bruke softwareserial fremfor i2c eller serial systemet er at i2c systemet ble brukt til å kontrollere skjermen og vi ønsket å ha serial portene fri sånn at vi hadde mulighetene for å sende meldinger via usb portene som er koblet opp til serial portene hvis vi måtte feilsøke noe. Dette gjorde at vi valgte å bruke softwareserial ettersom vi er ganske godt kjent med serial interfjeset og dette er ganske likt.

**Justering av volumet ved bruk av ultralyd**

Ettersom vår ide bak dette prosjektet var et det skulle være minst mulig bry for brukeren å bruke høyttaleren burde det være enkelt å justere volumet. En veldig elegant og stilig måte å løse det på er at man justere volumet ved å ta hånden opp og ned over høyttaleren. Vi valgte å legge de to viktigste funksjonene til en høyttaler inn i dette grensesnittet, justere volum og stoppe eller starte musikken.

For at vi skulle kunne bruke disse to funksjonene måtte vi vite hvor langt over høyttaleren hånden er. Til dette brukte vi et av eksemplene som Johannes hadde lagt ut på itslearning for å finne ut hvor høyt oppe noe er over sensoren. Men under testingen fant vi ut at man ikke klarte å holde hånden helt stabilt og at man noen ganger ente opp med å komme litt utenfor der ultralydsensoren registrete som gjorde at vi trengte en metode for å unngå at disse unøyaktighetene til brukeren skulle få volumet til å variere veldig mye. Dette valgte vi å løse ved å lagre de 10 siste verdiene som ble avlest i en array. For at vi kunne gjøre det trengte vi en 11 verdi som forteller hvilken verdi som er den eldste sånn at man kan erstatte den. Vi la også inn en funksjon som sikret at hvis verdien som skulle bli skrevet inn var over en gitt maksverdi så ble det satt 0 som verdi. Dette hadde to årsaker. For det første måtte man sikre at man kunne skille mellom hva som var avlesninger av taket og hva som var av hånden. Derfor hadde vi en maksverdi for hvor høyt det er rimelig at brukeren rekker hånden oppover. Grunnen til at vi satte de til 0 vis de var utenfor maksverdien er at det gjør det enklere senere i koden å telle opp antall verdier som er utenfor maksverdien.

**Skjermen**

Vi hadde opprinnelig tenkt å bruke en 3 tommers touch skjerm fra Ebay i dette prosjektet. Grunnen til at vi valgte denne skjermen var at den var stor nok til at det var mulig å vise informasjon om været, volum og potensielt hvilken sang som ble spilt uten at skriften ble veldig liten. Det at den er en touch skjerm var også positivt ettersom det ville gi brukeren flere valgmuligheter for hvordan han ønsker å kommunisere med høyttaleren. Det ville også gi muligheter for å lage et oppsetts program med verdier som hvor værdataen skal hentes fra og at man selv oppgir SSIDen til nettverket og passordet. Men denne skjermen kom dessverre ikke innenfor tidsfristen så vi måtte isteden bruke en liten 0,9 tommers skjerm. Denne skjermen er såpas liten at det bare er plass til informasjon om temperatur, hva slags vær det er og volum. Det er heller ikke en touch skjerm så vi ble nødt til å programmere inn hvilket nett man skal bruke og hvor man henter værdata fra direkte inn i programmet.

Den nye skjermen bruker i2c protokollen til å kommunisere med omverdenen. Dette er en ganske vanlig protokoll for å kommunisere mellom små elektronikk og krever bare 2 kabler. En kabel til å sende data (SDA) og en kabel for å ha en felles klokke (SCL). I2C protokollen er en slave og sjefs protokoll hvor Nodemcuen er sjefen og skjermen er slaven. SDA og SCL portene er spesifikke porter på Nodemcuen. Man kan bruke andre porter også med litt ekstra kode men vi valgte å bruke standarportene som er D1 (SCL) og D2 (SDA). Når vi kommuniserer med I2C enheter må vi spesifisere hvilken adresse som slaven har. For denne skjermen står det på baksiden at adressen er 0x78 men dette er en feil fordi her har de tatt med en ekstra bite som forteller om enheten er i skrive eller mottaker modus. Dette fant vi ut ved å bruke en I2C skanner som sjekker om det er noen I2C enheter tilkoblet og hvilken adresse enhetene har. Vi fikk da vite at skjermen har adressen 0x3C

Det er også en annen feil med skjermen vi har, det er at skjermen henger seg opp med jevne mellomrom sånn at den ikke vil motta kommandoer fra Nodemcuen. Dette problemet har vi funnet ut at vi bare får fikset ved å kutte strømmen til skjermen.

For å kommunisere med skjermen bruker vi 3 biblioteker

* Wire
* Adafruit\_GFX
* Adafruit\_SSD1306

Wire er et bygd inn program i Arduino som brukes for å kommunisere med I2C og TWC enheter. Dette biblioteket er nødvendig for at Adafruit bibliotekene skal fungere. Adafruit\_GFX er et program som brukes av alle andre Adafruit programmer for å ha et felles program for grafikken på skjermen. Vi kaller ingen funksjoner direkte fra det biblioteket i programmet vårt men det er nødvendig for at Adafruit\_SSD1306 skal fungere. Adafruit\_SSD1306 er et program som er laget for å kommunisere med OLED skjermer. Det er dette biblioteket vi henter alle funksjonene vi bruker for å skrive tekst på skjermen fra.

For å skrive tekst til programmet er det en ganske enkel prosess. Først må man spesifisere tekststørrelse, farge på tekst, font og start posisjon for å skrive men etter det er det bare å kalle på ‘’display.print(‘’tekst’’) for å skrive tekst på skjermen. Men dette lagrer bare teksten i minnet sånn at når vi kaller på ‘’display.display()’’ kan det hente alle endringene vi har gjort med det som står på skjermen samtidig og oppdatere det som står på skjermen. Vi bruker også funksjonen ‘’display.cleardisplay()’’ for å fjerne alt som står på skjermen fra før av sånn at vi kan skrive ny informasjon.

I løpet av programmet bruker vi skjermen til 2 ting. Det første er at når høyttaleren starter opp vil IP adressen til Nodemcuen vises på skjermen helt til en telefon kobler seg til Nodemcuen eller man tar hånden over ultralydsensoren. Grunnen til at vi valgte å gjøre det på denne måten er at appen må vite hvilken enhet som er høyttaleren. Da er det enkleste å bare taste inn IP adressen. Dette må ikke gjøres på nytt hvis man ikke endrer på nettverket så det er ikke et stort problem for brukeren. Etter at IP adressen er blitt vist på skjermen vises det 3 linjer med tekst hvor den øverste linjen er temperaturen ute. Andre linje er teksten som vi har fått fra yr som beskriver hvordan været er og tredje linje er volumet på høyttaleren.

**Link til nettstedet for å laste ned programmet for avlesning av XML filer fra yr.**

<https://github.com/kjartanmichalsen/Netver>