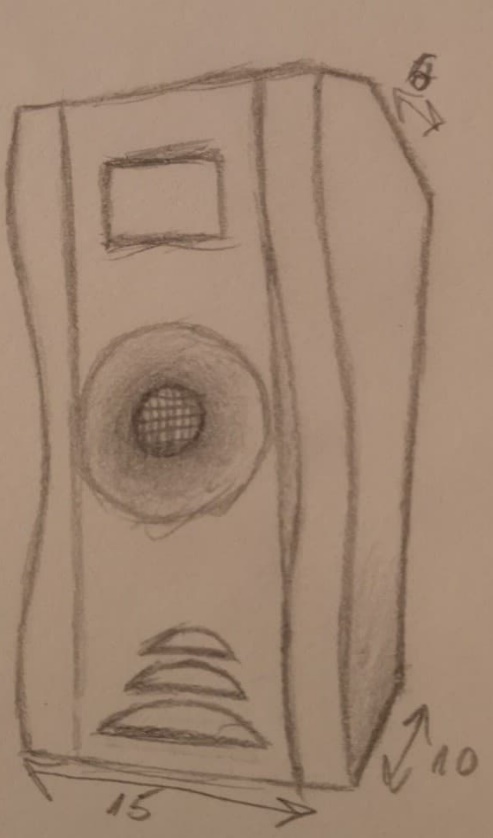
Rapport om utviklingen av JETS

**Dato:** 12.04.2019

**Av:** Hilmir Straumland og Åsmund Runningen



Innhold

[Introduksjon 1](#_Toc6064283)

[Valg av Arduino 1](#_Toc6064284)

[Strømforsyning 2](#_Toc6064285)

[Valg av høyttaler 3](#_Toc6064286)

[WIFI kort 3](#_Toc6064287)

[Innhenting av vær data 3](#_Toc6064288)

[Kommunikasjon mellom Arduinoen og Nodemcuen 4](#_Toc6064289)

[Beregning av volum ved hjelp av ultralydsensor 4](#_Toc6064290)

[Bluetooth kortet 5](#_Toc6064291)

[Skjerm 6](#_Toc6064292)

[Appen 8](#_Toc6064293)

[Google Speech-to-Text API 8](#_Toc6064294)

[Vurdering av produktet 9](#_Toc6064295)

[Hva vi har lært 10](#_Toc6064296)

[Videre lesning 10](#_Toc6064297)

# Introduksjon

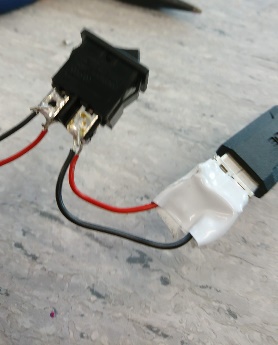
JETS eller “just enough to succeed’’ er en “smart høyttaler” hvor vi har prøvd å lage en høyttaler som ikke bare er fin å se på, men som også gjør hverdagen enklere for brukeren ved å ha et enkelt brukergrensenit og ekstrafunksjoner som løser hverdagslige spørsmål som hvordan været er. Dette har vært 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, fordi brukeren står på feil siden av høyttaleren eller har hendene fulle. Denne rapporten skal derfor sette dere inn i utviklingen av denne høyttaleren.

# Valg av Arduino

Til dette prosjektet hadde vi opprinnelig tenkt å bruke to Arduinoer. Grunnen til at vi tenkte å bruke to Arduinoer var at vi planla å bruke en touchskjerm som trengte en del av de digitale portene. Dette gjorde at det ble for få digitale porter på Arduinoen som gjorde det nødvendig med 2 Arduinoer. Planen vår var dermed å ha skjermen koblet til den ene Arduinoen og ha koblingene til Bluetooth kortet, WIFI kort og ultralydsensoren på den andre Arduinoen. Men ettersom skjermen ikke kom innenfor tidsfristen droppet vi å ha to Arduinoen ettersom erstatningsskjermen bare krever to digitale porter.

Vi vurderte også at vi ikke trengte en Teensy fordi vi bare skulle kjøre programmer for Bluetooth kortet og ultralydsensroen på Arduinoen. I ettertid har vi lagt merke til at Arduinoen har problemer med å kjøre det ferdige programmet. Det har vi observert ved at Arduinoen i perioder senker antall sykluser per sekund som vi kan observere ved at vi får færre oppdateringer av verdier via Seriellportene. Dette skjer med ganske jevne mellomrom og det hjelper ikke å ta av og på strømmen til Arduinoen. Vi tror derfor at det er et problem med at temperaturene blir for høye på Arduinoen. Dette problemet er heller ikke et spesifikt problem med vår Arduino, dette vet vi fordi vi har kjørt programmet på noen Arduinoen vi har hjemme og det er samme problemet. Derfor ville vi valgt å bruke en Teensy om vi skulle laget høyttaleren på nytt fordi dette kan være problematisk for brukeren hvis brukeren interagere med høyttaleren når den får disse problemene.

# Strømforsyning

For å levere strøm til høyttaleren valgte vi å bruke en vanlig Samsung USB reise adapter som vi kobler til høyttaleren via en USB-A port på baksiden av høyttaleren. Det er to grunner til at vi valgte å bruke en USB lader for å gi strøm til høyttaleren istedenfor batterier. Den første grunnen er at vi ønsker å ha muligheten til å ha skjermen på når høyttaleren ikke spiller lyd. 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 å være et møbel. 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.

Figur USB-A porten koblet til strømbryteren

Figur USB-A porten før vi loddet på kablene

Den opprinnelige planen var å bruke en USB-A port på høyttaleren. Det hadde med at USB-A har en passe størrelse som gjør den enkel å jobbe med og den er veldig vanlig. Når vi fikk delene så vi at Johannes isteden hadde kjøpt en USB-Micro port. Denne porten klarte vi å miste, ettersom Johannes ikke hadde noen reserver liggende måtte vi komme opp med en ny løsning. Vår første ide var da å lodde av USB-B porten på en gammel telefon Åsmund hadde hjemme. Dette fungerte ikke fordi loddebolten tinnet på kretskortet i telefonen ikke lot seg smelte. 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 klippet vi av kretskortet ettersom dette var gammelt og sprøtt og vi satt igjen med en USB-A port.

Figur Reise adapteren fra Samsung

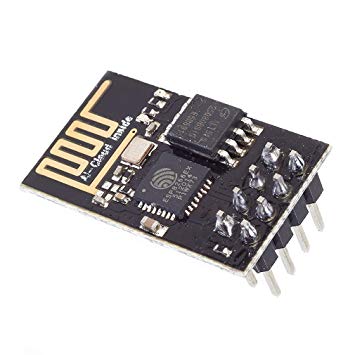
Når vi skulle lodde på strømkablene klarte vi å knekke av metallstengene som stak ut av porten for å koble seg på USB porten. Dette skjedde fordi pinnene var bøyd nedover og vi trengte å ha dem rett ut for å være sikre på at de ikke ville komme i konflikt med treverket. Noe pinnene ikke tålte. Dette 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 lodde sammen kabelen fra en gammel USB-A datamus og USB-A porten på en gammel telefonlader.

# Valg av høyttaler

Til dette prosjektet valgte vi å bruke en 10w 2,5 tommer høyttaler. Grunnen til at vi valgte akkurat denne høyttaleren var at designet ikke hadde plass til en større høyttaler. Dette er også grunnen til at vi ikke valgte å bruke stereo høyttalere. Den andre grunnen til at vi valgte akkurat denne høyttaleren er at strømstøpselet vi bruker kan levere 10W. Vi viste at resten av elektronikken ikke dro så veldig mye strøm og det var ønskelig å komme nærmest 10W grensen sånn at vi får høyest mulig lyd.

# Image result for nodemcu 1.0WIFI kort

For dette prosjektet har vi valgt å bruke en skjerm for å vise 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 leit at den er unyttig når man ikke hører 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 lar oss også vise volumet på høyttaleren.

Figur Nodemcu 1.0

Nodemcu 1.0 er en microkontroller med en innebygd esp8266 modul. Dette kortet kan ved å bruke den innebygde esp8266 modulen koble seg på nettverk, fungere som tilkoblingspunkt og være en liten sørver. Dette kortet har også en rekke digitale porter og støtter en del protokoller som i2c protokollen som lar oss kommunisere med skjermen. Ettersom denne er kraftig nok til å kjøre hele programmet for å kommunisere med appen og yr samt kontrollere en skjerm droppet vi den ene Arduinoen og kontrollerer skjermen direkte fra WIFI modulen, så kommuniserer Arduinoen med WIFI modulen ved å bruke SoftSerial biblioteket. Vi kan programmere denne modulen ved å bruke Arduino Ideen bare vi laster ned esp8266 utvidelsen.

Figur esp8266-01

# Innhenting av vær data

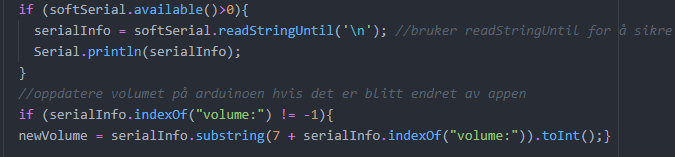
For å hente værdata valgte vi å bruke yr. Dette er fordi yr har gratis værinformasjon fra hele Europa som er fritt tilgjengelig for alle som ønsker å bruke det. Det er også veldig enkelt å bruke ettersom man bare bruker den vanlige nettadressen til et sted og legger på ‘’/varsel.xml’’ for å få en XML fil som man kan lese av.

For å lese av XML filene modifiserte vi et program for å vise værdata på en Nokia skjerm laget av Kjartan Michelsen som vi fant på GitHub (link under ‘’videre lesning’’). Dette programmet var spesielt designet for nettopp å lese av XML så modifiseringen 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 å 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 klasse med en funksjon som henter værdata fra yr og oppdaterer alle verdiene som lagrer informasjonen. Grunnen til at vi valgte å gjøre det på denne måten er at vi fikk vite 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 en stor skjerm ettersom det er enklere å lese. så da gjorde vi koden 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. Noe som lar oss gjøre endringer på koden raskere og teste ut hver bit av koden for seg selv.

# 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. SoftwareSerial er et bibliotek som lar deg bruke valgfrie digitale porter for kommunikasjon mellom Serial enheter. Grunnen til at vi valgte å bruke SoftwareSerial fremfor i2c eller vanlig Serial er at i2c kommunikasjon blir brukt for å kontrollere skjermen og vi ønsket å ha Serial portene frie sånn at vi har muligheten til å sende meldinger til en pc når vi feil søker koden. SoftwareSerial fungerer som vanlig Serial men man må spesifisere i starten av programmet hvilke porter man bruker. Da skaper man et objekt for klassen SoftwareSerial som kan sende og motta Serial informasjon. Dette bruker vi til å sende kommandoer frem og tilbake. Disse kommandoene skrives som sin egen linje så vi kan bruke readStringUntil() funksjonen som lar oss lese av en string som er lagret i Serial bufferen helt til en ny linje. Denne funksjonen returnerer det den har lest som en string som vi så kan teste om inneholder noen kommandoer ved hjelp av string.indexof(kommando) som returnerer -1 hvis kommandoen ikke er i stringen. Dette lar oss sende kommandoer til motparten i koden uansett hvor vi er i koden.



Figur 6 henter først string fra serial bufferen og tester om stringen inneholder kommandoen "volume"

# Beregning av volum ved hjelp av ultralydsensor

Ettersom vår ide bak dette prosjektet var å bygge en veldig bruker vennlig høyttaler 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/starte musikken.

For å få til dette bruker vi en ultralydsensor. En ultralydsensor fungerer ved at man sender pulser med lyd som reflekterer av objekter sånn at man kan registrere ekkoet. Dette gjør man ved å sende ut lyd med høyere frekvens enn det menneskelige øret kan høre. Så tar man tiden fra pulsene blir sendt til man registrerer ekkoet. Tiden som måles i mikrosekunder kan man gange med 0,034, siden lydens hastighet er 340 m/s, for å få antall cm det er frem og tilbake fra der lyden ble reflektert fra. Så deler man det på 2 for å få avstanden fra sensoren til objektet.

Dataene vi leser av ultralydsensoren lagrer vi i en liste med 11 verdier hvor verdi 0 er hvilken verdi som er den eldste av verdiene som er lagret og verdi 1-10 er verdiene som er lest av ultralydsensoren. Grunnen til at vi bruker en liste istedenfor å bare lagre den siste verdien som ble avlest er at vi la merke til at det er veldig vanskelig å justere volumet hvis man bare lagrer den siste verdien. Det har med at det er enkelt å komme utenfor området ultralydsensoren registrerer.

Vi har også en variabel for maksverdien som er en godkjent verdi. Dette er for at høyttaleren ikke skal registrere taket. Hvis en verdi er over maksverdien lagrer vi isteden 0 sånn at det er enkelt å senere i programmet søke gjennom listen og se hvor mange verdier som er over og under maksverdien. Etter at vi har registrert en ny verdi (eller 0) i listen regner vi ut gjennomsnittet av verdiene som er under maksverdien. Grunnen til at vi regner ut gjennomsnittet og lagrer det som en variabel er at vi da kan redusere tyngden til programmet ved å redusere antall regneoperasjoner ettersom vi bruke den verdien flere plasser i programmet.

Neste operasjon i programmet er å se om det er mer eller mindre en 5 verdier som er under maksverdien. Hvis det er endret til å være 5 verdier under maksverdien siden forrige syklus programmet kjørte lagrer den den gjennomsnittlige avstanden. Grunnen til det er at det da er noe over sensoren og vi trenger en utgangsposisjon som vi kan regne volumendringen ut ifra. Grunnen til at det er 5 verdier er at vi ønsker å ha nok verdier til at man kan stabilisere avlesningene før man begynne å endre volumet.

Når noe er over sensoren regner den ut delta y mellom objektet og utgangsposisjonen til objektet. Dette lar oss beregne hvor stor endringen i volumet skal være. Vi ganger delta y med en konstant som er sensitiviteten for volumjusteringen. Så legger vi det sammen med originalvolumet som vi lagret når objektet ble tatt over sensoren. Dette lar oss gjøre en oppdatering av verdien newVolume som er hva volumet skal endre seg til. (se Bluetooth kort for info om hvordan vi endrer volumet)

# Bluetooth kortet

Bluetooth kortet har en rekke forskjellige funksjoner. Vi bruker funksjonen for å sende lyd fra telefonen til Bluetooth kortet som så spilles av. Ettersom vi ikke bruker noen lydfiler på Arduinoen gikk det veldig greit. Bluetooth kortet har også en rekke porter som man kan sette strøm på for å endre på lyden som sendes ut. Opprinnelig hadde vi bare tenkt å kontrollere start/stopp, volum opp og volum ned på dette kortet. Så skulle vi ha funksjoner for å gå til forrige eller neste sang i appen. Men når Hilmir skrev koden for appen oppdaget han at det var noen feil med APIen til Spotify som gjorde at det ikke gikk å hoppe til neste sang eller forrige sang direkte fra appen. Dette var ikke et problem med koden i seg selv for det fungerte med andre strømmemedier men dette førte til at vi isteden sender beskjed gjennom Nodemcuen om å hoppe til neste eller forrige sang til Arduinoen som så setter strøm på den aktuelle porten.

For å koble sammen Arduinoen og Bluetooth kortet brukte vi opprinnelig et kretskort med en rekke NPN transistorer og motstander for å hente strøm ut fra en av portene på Bluetooth kortet (k com) og sende det tilbake på den ønskede porten. Men etter at noen andre i klassen testet uten transistorer viste det seg at det var unødvendig med transistorer. De testet da med motstandere mellom Arduinoen og Bluetooth kortet men vi hadde problemer med at signalene vi da sendte ble mer ustabile så vi droppet motstandere helt og koblet portene direkte til Arduinoen ved hjelp av kabler. Dette har fordelen at det blir færre komponenter som gjør at det er mindre som kan gå galt med høyttaleren og det gjør dermed høyttaleren mindre kompleks som er en fordel for alle som skal håndtere elektronikken.

Under dette prosjektet hadde vi en del utfordringer med Bluetooth kortet. Det er blant annet mangel på dokumentasjon om kortet som gjør at den eneste måten å finne ut hvordan det fungerer er ved å teste og feile. Etter testing fant vi ut at det er en treghet på kortet som gjør at det tar mellom 120 og 110 millisekund for at kortet registrerer et pause/start/neste sang/forrige sang signal. Så er det en treghet på sirka 2 sekunder fra signalet registreres til handlingen blir utført. Dette er en utfordring med pause signalene spesielt fordi man forventer at musikken pauses når man tar hånden over ultralyd sensoren men på grunn av denne tregheten kan det ta et par sekunder ekstra som gjør at brukeren kanskje drar hånden over sensoren på nytt fordi man antar at signalet ikke ble registret.

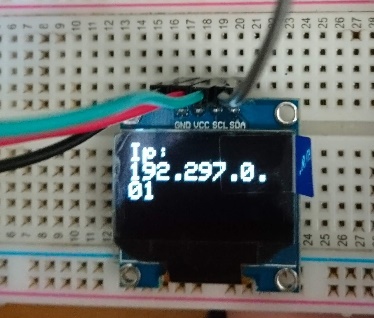
For å justere volumet på dette Bluetooth kortet må man sende strøm på portene en vis tid. Denne tiden beregner vi ved å ta forskjellen mellom det nye volumet (newVolume) og volumet vi startet med. Når vi starter å gjøre en endring i volumet lagrer vi starttiden for endringen. Så bruker vi formelen: og ser om den verdien er mindre en den nåværende tiden. Hvis den nåværende tiden er større enn denne formelen betyr det at volumet er justert til riktig punkt så oppdaterer vi volumet til å være det nye volumet og stopper å endre volumet på Bluetooth kortet. Grunnen til at vi ganger delta volum med 50 er at det tar sirka 5000 millisekund å endre volumet fra 0%-100% og 5000/100=50.

En utfordring med å endre volumet på Bluetooth kortet har vært å få endringen av volumet til å være presis. Grunnen til at justering av volumet ikke er presist er at det er mangel på dokumentasjon for dette Bluetooth kortet som gjør at vi har måtte testet oss frem til hvordan det fungerer. Vi har en mistanke om at det er en forsinkelse på sirka 1 sekund før den begynner å registrere en volumendring og vi tror hvor lang tid den bruker på en endring av volumet er eksponentiell med tiden den har justert volumet. Grunnen til at vi tror dette er at vi har lagt merke til at den endrer volumet veldig litte i starten for så å endre det veldig brått. Men ettersom vi ikke har noe dokumentasjon på Bluetooth kortet fant vi ut at vi fikk best resultater med å beregne at det er en linjer endring av volumet uten forsinkelser. Dette er grunnen til at man ikke kommer på helt samme volum om man justerer volumet noen ganger opp og noen ganger ned. Vi har derfor gjort det sånn at hvis appen er koblet til sender den volumoppdateringene til appen isteden sånn at man endrer volumet direkte på appen. Hvis man derimot ikke har appen koblet til vil den endre volumet på Bluetooth kortet.

# Skjerm

Vi hadde opprinnelig tenkt å bruke en 3 tommers touch skjerm fra eBay til dette prosjektet. Grunnen til at vi valgte denne skjermen var at den var stor nok til at det var mulig å vise informasjon om vær, volum og potensielt hvilken sang som blir spilt av uten at skriften blir veldig liten. Det at skjermen er en touch skjerm er også positivt ettersom det ville gi brukeren flere valgmuligheter for hvordan brukeren ønsker å kommunisere med høyttaleren. En touch skjerm vil også gi oss muligheten til å lage et oppsetts program hvor man legger inn posisjonen man ønsker å hente værdata fra og SSID og passord for nettverket høyttaleren skal koble seg til. Men denne skjermen kom dessverre ikke innenfor tidsfristen. Vi bruker derfor en 0,9 tommers 128\*64 pikslers OLED 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.

Figur skjermen som viser været og volum

En OLED skjerm er satt sammen av mange små organiske dioder laget av organisk materiale. Diodene ligger mellom to elektroder som gir strøm til diodene og dermed kontrollerer hvilke dioder som lyser. OLED skjermen vi har kontrolleres via I2C protokollen. I2C protokollen er en sjef/tjerner protokoll som bruker serial kommunikasjon til å kommunisere mellom to enheter som ikke trenger stor båndbredde. Serial kommunikasjon er at man sender dataen over en kabel i rekkefølge. I2C protokollen krever to kabler hvor den ene kabelen sender data (SDA) og den andre kabelen fungerer som en felles klokke (SCL).

Figur skjermen som viser IP adressen som skrives inn i appen

For å kommunisere med skjermen bruker vi bibliotekene Wire, Adafruit\_GFX og Adafruit\_SSD1306. Wire er et innebygd bibliotek i Arduinoen som brukes for å kommunisere med I2C og TWC enheter. Dette biblioteket er nødvendig for at Adafruit bibliotekene skal fungere. Adafruit\_GFX er et grunnbibliotek for andre Adafruit biblioteker som gjør en del felles ting som å lage grafikken som skal vises på skjermen. Vi kaller ingen funksjoner direkte fra Wire eller Adafruit\_GFX biblioteket men det er nødvendig for at Adafruit\_SSD1306 skal fungere. Adafruit\_SSD1306 er et program som er laget for å kontrollere 128\*64 piksler og 128\*32 pikslers OLED skjermer. Det er dette biblioteket vi henter alle funksjonene vi bruker for å skrive tekst på skjermen fra.

Når man lager grafikk på en skjerm gjør man to ting. Først lager man et nytt bilde over hvordan teksten skal se ut. Dette bilder kan man lage ved hjelp av funksjoner som print() og clearDisplay(). Bilder blir så lagret i minnet på mikroprosessoren helt til man kaller på funksjonen display(). Display() oppdaterer det som vises på skjermen til det nye bildet man har lagret i minnet på maskinen. Dette gjør at man kan oppdatere hele bildet samtidig istedenfor å måtte oppdatere bildet i flere operasjoner som brukeren kunne lagt merke til som uferdige bilder og rott på skjermen når bildet oppdateres.

I løpet av programmet bruker vi skjermen til 2 ting. Det første å vise IP adressen til Nodemcuen 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 IP adressen til høyttaleren for å kunne koble seg til den. 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. Vi hadde opprinnelig tenkt å vise værmeldingen som vi får fra yr som et av de standardiserte værsymbolene og ha beskrivelsen under symbolet men dette var det ikke plass til på skjermen.

Under testingen av skjermen oppdaget vi at det er en feil med kretskortet på skjermen. Når bildet ikke er blitt oppdatert på en stund hender det at kretskortet stopper registrere nye beskjeder som kommer fra Nodemcuen. Dette ser ut til å være et problem med kretskortet for når man tar strømmen av og på fungerer den helt fint igjen. Dette er ikke noe vi kan gjøre noe med ettersom vi ikke kan detektere at problemet har oppstått fra Nodemcuen. Den eneste løsningen er dermed å skru høyttaleren helt av og på igjen når dette skjer.

# Appen

Ettersom vår ide bak denne høyttaleren er å gjøre den mest mulig brukervennlig burde man ha muligheten til å kontrollere høyttaleren når man har hendene fulle eller ikke står i nærheten av høyttaleren. Vi lagde derfor en app med stemmestyring som gjør at man kan kontrollere høyttaleren gjennom enkle kommandoer som ‘’music start’’ og ‘’next song’’. Appen ble programmert i AndroidStudio og bruker Java. Den kan derfor bare brukes med Android telefoner. Grunnen til at vi valgte Java er at det er et lett programmeringsspråk som ikke krever mye av telefonen.

Kommandoene man kan si til appen er ‘’music stop’’, ‘’music start’’, ‘’music next song’’ og ‘’music previous song’’.Appen gjenkjenner kommandoene ved å bruke Google Speach-to-Text API. Denne APIen gjør om lydene til en string som vi så søker gjennom etter ordet ‘’music’’ hvis ordet ‘’music’’ kommer opp vil appen så se om ordet etter ‘’music’’ er en av kommandoene. Hvis det er en av kommandoene vil den sende beskjed til Nodemcuen. Fordi systemet har noen ganger problemer med å forstå hva som blir sagt og da printer ut for eksempel stock i stedet for stop, så har vi trengt å sette inn flere ord inn som enten stopper eller starter musikken. Dette gjør at det er større sannsynlighet at det “hører” riktig.

På appen så ser i øverst hva som blir sagt så man vet hva APIen tar opp. Nederst er det et felt som man skl skrive inn IP-adressen også trykker man på connect knappen ved siden av så appen kan prøve å koble seg til node mcuen ved å sende en kommando som heter connect også hvid node mcuen mottar det sender den “ok” tilbake til appen og knappen blir grønn.

Node mcuen fungerer som en webserver på det lokale nettverket ved at den lager en nettside som man kan å tilgang til gjennom lokalt nettverk. Dette er grunnen for at vi trenger den lokale IP-adressen til node mcuen.Node mcuen starter med en nettside som vi da kan bruke appen med en http request til å hente data fra siden. Dette gjør at vi kan hente volumjusteringer fra arduinoen også forandre på lyden på telefonen i stedet for å gjøre det på bluetooth kortet. Dette gjør at vi får en bedre justering av volumet enn hvis det hadde blit gjort på bluetooth kortet. Det er et bibliotek som følger med Android Studio som heter Volley som gjør det lettere å gjøre nettverks forespørsler inne i appen. Dette gjør også det lettere å sende kommandoer fra telefonen til node mcuen fra for eksempel stemmestyringen som skjer på telefonen.

[overgang]

Planen var opprinnelig å bruke et bibliotek i Java som heter Sphinx hvor man kan forandre lyd til tekst. Utfordringen med dette biblioteket er at man må trene stemmestyring selv som ville tatt tid og lagrings plass. Vi prøvde derfor ut IBM sitt sky-system kalt IBM Watson. Dette systemet har en Speech-to-Text API som man kan bruke gratis opp til 100 minutter API tid som er total tid av det serverne bruker på å gjøre overføring fra lyd til tekst. Problemet oppstod først når jeg skulle autorisere IBM Watson. IBM Watson har tidligere brukt en autoriserings metode som inneholdt å putte inn brukernavn og passord som var spesielle API nøkler. Men det hadde kommet en ny metode med at man bruker en API-nøkkel med en URL. Her var det lite dokumentasjon om hvordan man skulle implementere dette inn i Java til å bruke det i Android Studio. Til slutt så klarte vi å få det til men da kom det opp at APIen ikke kunne brukes i denne regionen. Det viste seg at man kunne kun bruke IBM Watson under regionene Sydney, Frankfurt, London, Dallas, Tokyo og Washington DC. Det kan være at vi gjorde en feil og at det er derfor det ikke fungerte, men vi bestemte å finne en annen tjeneste med muligheten for stemme til tekst overføring.

Vi hadde tidligere sett på Google Cloud, men der må man registrere seg med kort nummer osv. Hilmir kjenner en som er registrert og har Google Cloud, så vi fikk bruke hans systemer. Google sin Text-to-Speach API fungerte som den skulle og det var mye mer og tydeligere dokumentasjon tilgjengelig som gjorde det enklere å sette opp. Google sin API er også veldig rask fordi den bruker maskinlæring og den sender teksten tilbake i sanntid som er en fordel. Google Cloud Speech-to-Text er også gratis opp til en bestemt API tid som er 60 minutter.

### Google Speech-to-Text API

Google sine sky systemer har en stemme til tekst API som man kan bruke for å forandre lyd til tekst. Her sender man lyden gjennom APIen som google har også får man ut teksten som blir sagt. Denne APIen krever at man er på nett ettersom man laster opp lydfilen til google sine sørvere som gjør kalkulasjonene og sender tilbake teksten. Dette gjør at programmet er raskere enn om man gjorde det lokalt på enheten.

Etter å ha testet dette så ser vi at appen fungerer som den skal med å gjenkjenne det som blir sagt også sende kommandoer via wifi til node mcuen. Også på grunn av at innenfor teknologi bransjen så er engelsk som er det hoved språket så blir det slik at man må si alt til appen på engelsk. Dette er også på grunn av at norsk kan være lettere å misforstå hva som blir sagt, så da var engelsk det valget vi tok som språk på appen.

# Vurdering av produktet

Tatt i betraktning tiden og ferdighetene våre er dette et produkt med meget høy kvalitet. Produktet har avanserte funksjoner som stemmestyring og justering av volumet ved å bruke ultralyd. Dette gjør produktet til noe utenom det vanlige som gir brukeren flere valg for hvordan den ønsker å kommunisere med høyttaleren. Koden holder gjennomført høy kvalitet med god seperasjon som gjør den mer moduler og dermed enklere å feilsøkke og modifisere for andre bruksområder.

Det største problemet med dette produktet er Bluetooth kortet. Hadde vi hatt et vanlig Bluetooth kort kombinert med en Teensy kunne vi fått et mye bedre produkt ettersom det er en del problemer med treghet og unøyaktig volumjustering på Bluetooth kortet. Men dette var vi ikke klar over når vi startet dette prosjektet. Hadde vi gjort det igjen ville vi mest sannsynlig ha droppet Bluetooth kortet og sendt lyd signalet over WIFI kortet vi uansett har. Men dette kunne vi ikke endre på når vi oppdaget problemet. Dette vil kraftig redusere brukervennligheten til dette produktet ettersom brukeren vil tro at signalene for stopp og start sang ikke er blitt registret ettersom det vil ta 1-2 sekunder før musikken stopper. Det vil også være et stort problem for brukeren når brukeren justerer volumet opp og ned og volumet ender opp med å være noe helt annet en Arduinoen sier at det skal være.

En annen ting som trekker dette produktet litt ned er loddingen. Loddingen er ikke spesielt dårlig men det var noen utfordringer med at vi ikke fikk loddingen av kabler på Bluetooth kortet til å sitte. Dette er ikke et problem for brukeren men det gjør at man må være ekstra forsiktig når man opererer elektronikken ettersom det er noen kabler som er festet med krympestifter som kan løsne og play pinnen på Bluetooth kortet kan være litt løs. Som sagt er ikke loddingen veldig dårlig men sammenlignet med resten av utførelsen vil vi påstå at det er det svakeste leddet av dette produktet.

Det siste som reduserer kvaliteten på høyttaleren er skjermen. Ettersom dette ble en såpass liten skjerm som ikke er touch har ikke brukeren muligheter for å legge inn passord og ssid til nettverket eller hvor brukerne ønsker å hente værdata fra. Dette ville åpenbart ikke vært akseptabelt i et ferdig produkt men vi hadde ikke så mye valg ettersom skjermen ikke kom. Det er også noen utfordringer med stabiliteten til stemmestyringen. Under testingen av stemmestyringen fant vi ut at den veldig ofte oppfatter kommandoer som ord som ligner på kommandoene vi ønsket å si. Vi fikk lagt inn en del kommandoer som ligner på de ønskede kommandoene sånn at vi minimerer problemet men det er fortsatt en utfordring ettersom vi bare har hatt muligheten til å teste hvilke ord den hører feil når vi sier kommandoene og det kan dermed være en utfordring for en del brukere.

# Hva vi har lært

Dette prosjektet har lært oss en rekke nyttige ting. Vi hadde allerede litt erfaring med programmering av Arduinoen før vi startet dette prosjektet og begge to hadde stor erfaring med andre programmeringsspråk. Vi hadde også en del kunnskap om elektronikk. Men selv om vi kunne en del fra før av har vi lært mye nytt. Vi har blant annet utviklet ferdighetene våre innenfor lodding ettersom vi hadde liten erfaring med det fra før. Dette er noe vi merket i slutten av prosjektet hvor vi trengte færre forsøk for å lodde sammen forskjellige deler og loddingene holdt generelt bedre kvalitet. En annen ting som ingen av oss hadde noe erfaring med fra før av er å arbeide med designere. Det var ganske interessant å jobbe sammen med designerne fordi dette gjorde at man måtte forholde seg til hvor designeren var i prosessen og gjøre kompromisser for at produktet skal fungere. Det var ikke store uenigheter mellom oss ettersom designet hadde ganske stort rom for elektronikken vi ønsket å ha. Men det er allikevel en nyttig erfaring å ha med seg videre.

Vi har også utvidet vår erfaring med programmering av microkontrollere. Ettersom Åsmund har hatt hovedansvaret for programmene av microkontrollene har han utvidet sin kunnskap om kommunikasjon mellom forskjellige microkontrollere ved å ha laget kode for kommunikasjon mellom Arduinoen og Nodemcuen og ved å skrive kode for å kontrollere skjermen. Åsmund har også utvidet kunnskapen rundt WIFI moduler ved å bruke forskjellige typer WIFI moduler. Han har også lært hvordan man søker gjennom XML filer etter å ha modifisert koden for å lese av XML filene fra yr. Åsmund har også fått mer kunnskap om programmering i C++ ettersom han hovedsakelig har kunnskap fra Python. Han har også lært mer om hvordan man bruker GitHub som er et veldig vanlig verktøy for at flere kan arbeide på samme prosjekt og ha versjonskontroll. Dette brukte Hilmir fra før av så da var det naturlig at vi brukte det programmet til dette prosjektet.

Hilmir har også forbedret kunnskapen om å programmere microkontrollere, og det å la dem kommunisere mellom hverandre. Og det å kommunisere et wifi kort med en telefon igjennom en app og mer om hvordan wifi og nettverk fungerer. I tillegg til det har Hilmir lært mer hvordan stemmestyring fungerer og hvordan man kan bruke det i ulike applikasjoner. Også har han lært mer om hvordan man skal bruke APIer i prosjektene sine og at Google Cloud har mange ulike kraftige AI APIer som kan brukes av alle. Hilmir har også forbedret kunnskapen i par programmering og det å jobbe sammen med andre. Andre ting som han har lært er mer som elektronikk og det å sette sammen noe til

# Videre lesning

**Utgangsprogrammet for innhenting av værdata fra yr:**

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

**Koblingsskjema:**



**Kode for Arduino**

**Kode for Nodemcuen**

**Kode for appen**