Rapport om utviklingen av JETS

**Dato:** 12.04.2019

**Av:** Hilmir Straumland og Åsmund Runningen

Innhold

[Introduksjon 1](#_Toc5794766)

[Strømforsyning 1](#_Toc5794767)

[Nodemcu 2](#_Toc5794768)

[Innhenting av vær data 2](#_Toc5794769)

[Kommunikasjon mellom Arduinoen og Nodemcuen 3](#_Toc5794770)

[Justering av volumet ved bruk av ultralyd 3](#_Toc5794771)

[Skjermen 4](#_Toc5794772)

[Valg av Arduino 5](#_Toc5794773)

[Valg av høyttaler 5](#_Toc5794774)

[Bluetooth kortet 6](#_Toc5794775)

[Vurdering av produktet 7](#_Toc5794776)

[Hva vi har lært 7](#_Toc5794777)

[Videre lesning 8](#_Toc5794778)

# 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.

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.

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 fikk høyest mulig lyd.

# WIFI 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.

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.

# 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 lott 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. 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 å få til dette brukte vi en ultralydsensor. En ultralydsensor fungerer ved at vi setter strøm på en av portene. Da sender ultralydsensoren pulser med lyd. Denne lyden har en høyere frekvens en det menneskelige øret kan høre som gjør at det ikke er forstyrende for de i nærheten. Når vi har sendt pulsene med lyd kan vi vente på å registrere ekkoet til signalene. Når det kommer et ekko betyr det at signalet har truffet noe og blitt reflektert. Ultralydsensoren sender da et signal tilbake på en port når den registrerer et ekko. Da kan vi bruke tiden mellom signalet ble sendt til vi mottok ekkoet til å bestemme hvor langt unna det objektet som lyden reflekterte fra var. Dette kan vi gjøre ved å dele tiden på lydens hastighet som er en konstant. Når vi har gjort det må vi dele på 2 fordi vi tok tiden lyden brukte frem og tilbake. Det vi da får er lengden lyden gikk.

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 registrerte 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 standard portene 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.

# 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 hvis 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.

En annen utfordring med dette kortet er justering av volumet. Etter mye testing kom vi frem til at det tar sirka 5 sekunder med strøm på en av volum portene for å justere volumet fra 0 til 100%. Men det er også en hvis treghet før den registrerer et signal som gjør at hvis man justerer volumet flere ganger men ikke helt ned og helt opp vil ikke volumet være på det nivået man forventer. Selv om vi brukte desibel målere på internettet i kombinasjon med en konstant pipelyd på høyttaleren for å finne hva forsinkelsene ligger på klarte vi ikke å få finne ut hvor lang tid det tar. En av grunnene til at det var så problematisk å få justeringen av volumet til å bli presis er at det ikke virker som det er en linjer justering av volumet. Dette legger man veldig godt merke til for når den er på maks volum er det en stund hvor den nesten ikke justerer volumet før kortet plutselig justerer volumet veldig raskt. Dette er grunnen til at vi valgte å sende oppdateringer om volumet til appen hvis den er koblet til. Da kommer vi utenom problemet med presisjonen på volum justeringen ved at Bluetooth kortet kan stå på maks volum så justerer isteden appen volumet på telefonen. Men dette fungerer ikke når appen ikke er koblet til telefonen som spiller av musikk så da må vi isteden justere volumet på høyttaleren. Men ettersom appen er en grunnleggende del av høyttaleren forventer vi at de som bruker høyttaleren også har appen installert som gjør at de ikke vil merke noe til disse problemene.

# Vurdering av produktet

Denne høyttaleren har ikke en kvalitet som tilfredsstiller salg til forbrukere. Det er blant annet en del feil i koden som vi verken har tid eller ressurser til å fikse. Vi burde blant annet fått testet stemme gjenkjenningen på mange flere personer sånn at vi kunne vært mer sikre på at vi ikke får store problemer med at en gruppe med mennesker ikke klarer å bruke stemmekommandoene. Dette kommer av problemene som vi har skrevet om høyere oppe i rapporten om hvordan google sin stemmegjenkjennings servise ikke klarer å fange opp nøyaktig hva vi sier. Vi er ikke et representativt utvalg av språklige nyanser hos befolkningen og har derfor begrenset med muligheter til å forbedre appen.

Skulle dette vært et ferdig produkt vi kunne solgt måtte det også vært muligheter for å gjenkjenne en høyttaler på nettverket istedenfor å måte skrive inn IP adressen. Dette var dessverre ikke mulig fordi **‘’Hilmir sett inn grunn’’** For at dette skulle vært et ferdig produkt måtte vi også hat en touch skjerm sånn at vi kunne kommet utenom å kode SSID og nettverkspassordet rett inn i koden.

Det er også noen feil med skjermen men dette er ikke noe som trekker ned kvaliteten på arbeidet som er gjort med høyttaleren ettersom det mest sannsynlig er en produksjonsfeil.

Hadde vi hatt intensjoner om å selge høyttaleren videre ville vi også byttet ut Bluetooth kortet. Det er en uakseptabel treghet fra et signal sendes fra Arduinoen til Bluetooth kortet, som gjør at produktet er mindre brukervennlig. Det er også et problem med at volum justeringen er upresis som gjør at vi må skape kompromisser som å sende volum justeringene til telefonen.

Alt dette i kombinasjon med at elektronikken burde sitte på et eller to kretskort uten mesteparten av kablene vi har i dag gjør at dette ikke er et produkt som er klart for masse produksjon. Men produktet holder en høy kvalitet tatt i betraktning utstyret og tiden tilgjengelig. Det viser også hvordan man kan bygge en høyttaler som har noen unike funksjoner uten at den trenger å koste så veldig mye mer. Så med litt mer tid og litt bedre utstyr burde dette konseptet kunne gjøres om til et produkt folk har lyst til å kjøpe.

# 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 fra før av. Vi hadde også en del kunnskap om elektronikk. 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 av og nå har loddet ganske mye. 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.

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 og 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 skriv hva du har lært. (bruk 3 person sånn at det er tydelig for leseren hvem det er snakk om).**

# Videre lesning

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

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

**Koblingsskjema:**



**Kode for Arduino**

**Kode for Nodemcuen**

**Kode for appen**