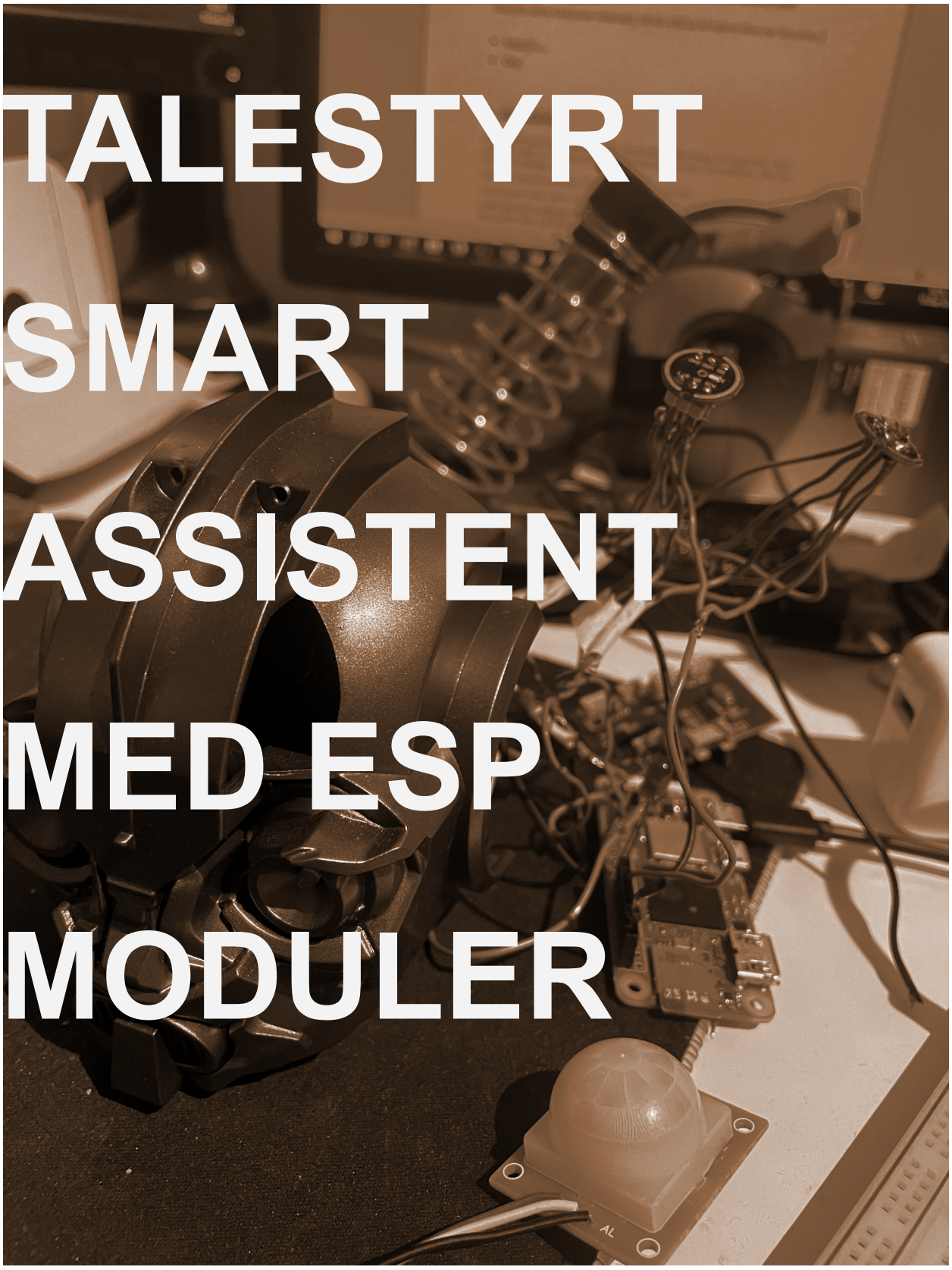

TALESTYRT
SMART
ASSISTENT
MED ESP
MODULER

Innholdsfortegnelse

Sammendrag

Produktet utviklet er en Virtuell Smart Assistent kalt Oda. Hovedmålet med produktet er at det skal være nyttig. Dette produktet skaper sin nytte på tre måter. Det er lett tilgjengelig og lett forståelig. Du snakker til en maskin og den snakker tilbake. Ingen koding eller skriving trengs.

Den er også veldig intelligent ettersom den er koblet opp mot en GPT modell. Dette gjør den til et ganske sterkt verktøy som kan brukes i mange sammenhenger. Har du for eksempel et spørsmål du bare plutselig ble litt nysgjerrig på men ikke helt gidder å google, for eksempel hvor stor er en blåhval i forhånd til en elefant, slike type bruksområder passer denne smart assistenten perfekt til.

Til slutt så er det oppkoblingen mot andre enheter som er veldig nyttig på denne smart assistenten. Sensorer koblet til ESP enheter kan plasseres rundt omkring for å skape et mye større nettverk og sette opp mange muligheter. Du kan spørre om enkelte mennesker er hjemme hvis du har en bevegelsessensor på rommet deres, Du kan sette en støv sensor bak pc en, eller under sofaen for å finne ut av når du må begynne å støvsuge. Eller så kan du sette et termometer i et loft eller skur for å finne ut av om det kommer til å fryse. Det er veldig mange muligheter som åpner seg med muligheten til å koble til enheter.

Dette produktet ble lagd i sammenheng med en 2 måneder lang oppgave i IELS1001 Ingeniørprosjekt. Denne oppgaven handler om smart-city og smart homes, men jeg fikk mye frihet rundt valg av problemstilling og produkt jeg skulle utvikle i perioden. Jeg valgte dette produktet fordi det var noe jeg synes virket svært interessant, og passer perfekt innenfor oppgaven.

Terminologi

EEROM = Noen få bytes du kan skrive til får å lagre ting mellom boots

Ping = Delay mellom server og klient

Pinge = Sende et kort melding i håp om et svar som tegn på liv

SSID = Nettverksnavn

Wake word = navnet på smart assistenten

IP = Adressen til enheten på internettet

EPS = Mikrokontroller

RNN = Recurrent Neural Networks, nettverk som husker litt mellom hver loop

CNN = Convolutional neural network - Konvolusjonelt nevraltnetverk

GRU = Type RNN

Problemstilling

Oppgaven handler om å lage en smart assistent. Denne smart assistenten skal ha flere funksjoner. En av de viktige funksjonene den må ha er evnen til å benytte seg av ESP moduler. Disse modulene kan for eksempel være sensorer som er festet til ESP-32 er. Målet er at produktet skal virke som en vanlig smart assistent, men i tillegg ha mulighet til å benytte seg av data eller andre funksjoner som ESP modulene kan bistå med.

For å oppnå dette overordnede målet er det mange punkter vi må treffe.

Den må kunne forstå hva som blir sagt og når det blir sagt og reagere ut ifra det. Dette er et av de viktigste punktene for at resten av assistenten skal virke så naturlig so mulig. Hvis teksten vi får fra starten av er feil, hjelper det ikke å gjøre masse med den senere. Er det et sted vi må satse på nøyaktighet så er det her.

Den må kunne tenke litt. Hvordan den tenker er ikke så farlig, det kan være en hau med if statements det kan være et nevralt nettverk, hva er ikke viktig. Hovedpoenget er at det høres naturlig ut. Dette er den store forskjellen på et produkt som blir brukt og et som ikke blir brukt;

hvor nyttig er faktisk produktet. Det kan argumenteres for at desto mer intelligent assistent er, desto mer nyttig er den. Det å lese av sensorer hjelper også mye med nytten til produktet.

Til slutt når vi har dataen som skal deles må den til brukeren på en måte. Det er mange forskjellige måter dette kunne vært på, for eksempel en skjerm eller det som er veldig naturlig for oss og som vi kommer til å bruke, at den kan snakke. Dette gjør at det blir mye mer naturlig å snakke til den (når den snakker tilbake). Og det gjør det også til et lettere verktøy å bruke. Når du kan snakke til den og bare høre på alt den har å si er det lettere å spørre den om ting uten å miste fokus på det du gjør fra før.

Teoretisk grunnlag

Fysiske domene

I det fysiske domene ble det mye jobb med maskinvare. Et av målene med denne smart assistenten var at den ikke skulle være for dyr men fortsatt kraftig. Denne balansen er vanskelig å finne, men viktig for å ende opp med et godt produkt.

I dette prosjektet har jeg ikke kjøpt inn noen nye komponenter, men heller brukt deler jeg har hjemme og deler som jeg har fått fra skolen. Dette har veldig sterk påvirkning på hvilke deler som er brukt i selve produktet.

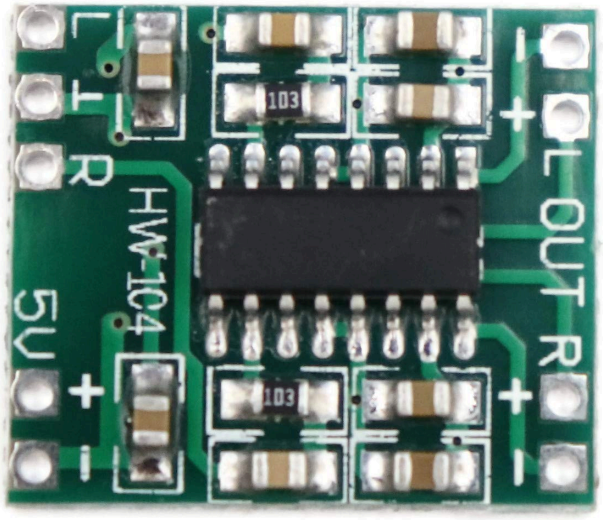
Nedenfor er en liste med alle komponentene og litt om hvorfor de ble brukt.

Høytaler

Jeg hadde liggende en gammel høyttaler som jeg kunne gjenbruke for dette prosjektet. Denne høyttaleren passer perfekt fordi det er en 3 ohm høyttaler som er lite nok til at jeg ikke trenger ekstra strømforsyning for bra lyd.

Denne høyttaleren er også humanoid som styrker følelsen om at man ikke bare snakker med en datamaskin som er nyttig og et pluss ved å bruke denne i dette produktet.
Bildet er hentet fra:
https://he.aliexpress.com/item/1005001297858932.html?gatewayAdapt=glo2isr

Denne høyttaleren kan kjøpes her for en pris rundt 200 kr.



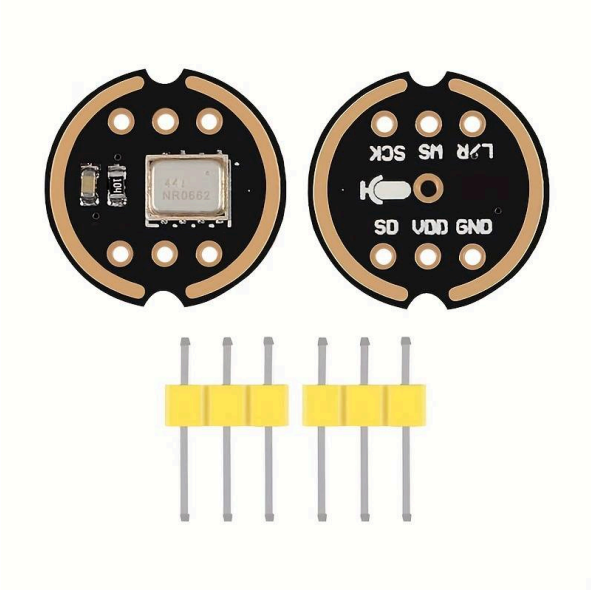
Oppamp

For å forstærke lydsignalet før jeg sendte det til høytaleren trenger jeg en forstærker. Jeg valgte da å bruke en HW-104 ettersom den var den eneste jeg hadde liggende og kan fint brukes i denne sammenheng.

Noe å merke seg er antall utganger og innganger på forsterkeren. Ettersom produktet bare benytter seg av en høyttaler trengs ikke stereo amplifikasjonen dette brette støtter. Senere under bruk av dette brettet vil jeg derfor bare bruke venstre kanal og koble dette direkte til høyttaleren. Dette vil ikke senke lydkvaliteten over en monoforsterker som er spesifisert til å bare forsterke signalet til en høyttaler.

Bildet er hentet fra
https://www.amazon.in/Electronicspices-PAM8403-HW-104-Digital-Amplifier/dp/B07YX6X2BG

HW-104 kan kjøpes her for rundt 12 kroner.



Mikrofon

Når det kommer til mikrofonen jeg skulle bruke var det mer usikkerhet rundt hva jeg skulle gå for. Jeg hadde enten muligheten til å gå for en MAX4466 Adafruit mikrofon eller INMP441.

Disse to har begge sine styrker og svakheter. MAX4466 Adafuit har dynamisk grain, noe som vil si at volumet automatisk stiller seg til hvor mye bråk det er. Dette er veldig nyttig for å høre noen snakke tydelig, enten de snakker veldig stille eller alt for høyt.

Bildet hentet fra:
https://www.temu.com/hr/5pcs-inmp441-omnidirectional-microphone-module-mems-high-precision-low-power-i2s-interface-support-esp32-g-601099518205196.html

Svakheten til MAX445 Adafruit derimot er at man må snakke i retning mikrofonen for at den skal fungere optimalt. Dette er litt motsatt av det andre alternativet jeg hadde, nemlig INMP441. INMP441 er en omnidirectional mikrofon. Noe som vil si at den plukker opp lyden i høy kvalitet uansett hvilken retning du snakker i. Denne mikrofonen funker kort sagt slik at den har et hull på den ene siden (opp) hvor alle lydbølger som går inn i det hullet blir tatt opp. Dette gjør at vinkelen til lyden er ekstremt fleksibel.

Når man benytter seg av dette produktet er det viktig at hvor man står i rommet ikke har veldig stort utslag på kvaliteten av kommando forståelsen. Derfor valgte jeg å benytte meg av to INMP441 mikrofoner med litt vinkel mellom hverandre for å ta opp så mye av lyden i rommet som mulig.

INMP441 kan kjøpes her for rundt 20 kroner stykk.

Raspberry PI

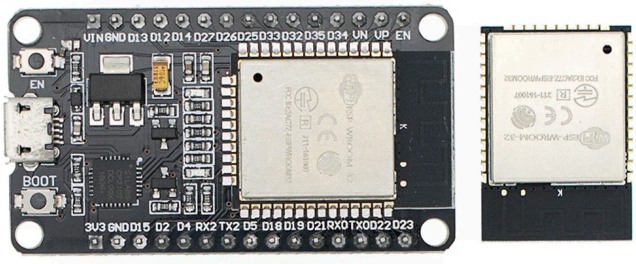
Hoveddata enheten min er en Raspberry Pi Zero W 2. Det er mange grunner til hvorfor denne passer perfekt til dette prosjektet. Et av de største fordelene med Raspberry Pi Zero W 2 er størrelse og pris uten å kutte kraften den har.

I tillegg til disse egenskapene er den også veldig robust og har mange tilkoblingspunkter inkludert 29 GPIO pins. Den får også strøm direkte fra USB, noe som gjør den lettere å finne strømforsyning til. Bildet hentet fra: https://www.raspberrypi.com/products/raspberry-pi-zero/

En annen positiv ting ved raspberry pi generelt er at de er svært populære og mye brukt. Dette bidrar til mye dokumentasjon og informasjon på internett som gjør de mye lettere å feilsøke. Zero W versjonen av raspberry pi har også innebygd bluetooth og wifi som begge blir tatt i bruk under dette prosjektet.

Raspberry Pi Zero W 2 kan kjøpes her for rundt 190kr.

TENSTAR ROBOT



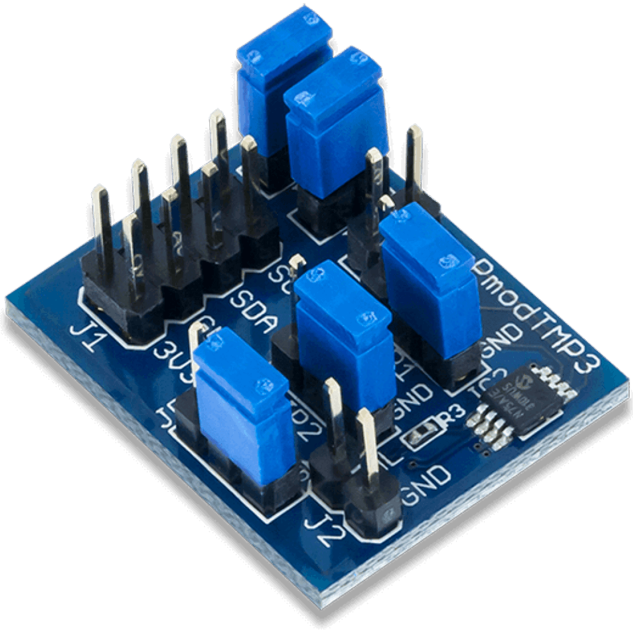
ESP32

Når det kommer til modular som skal kunne knyttes til smart assistenten må disse også ha en dataenhet for å overføre data, samt gjøre små kalkulasjoner. Her passer

ESP32 veldig bra for mye av de samme grunnene som Rapberry Pi'en. At den har mange pins gjør det lett å legge til ekstra maskinvare som sensorer. At den har wifi og bluetooth gjør at det er lett å starte/opprettholde kommunikasjon. Og til sist at den er lett og feilsøke, mye takket være dens massive popularitet.

Bildet er hentet fra:
https://ae01.alicdn.com/kf/S9132296fdeae4c4b8497310fd22c3e15o/ES32-wifi-bluet-ESP-32.jpg_.webp

ESP32 kan kjøpes her for rundt 40 kr.



Termometer

PmodTMP3 er at digitalt termometer, og en av sensorene lånt fra skolen.

Dette termometeret er overraskende presist og bruker i2C standarden som gjør den veldig lett å koble opp.

Den hadde også veldig fin dokumentasjon som gjør den lett å problemsøke.

Bildet hentet fra
:https://digilent.com/shop/pmod-tmp3-digital-temperature-sensor/

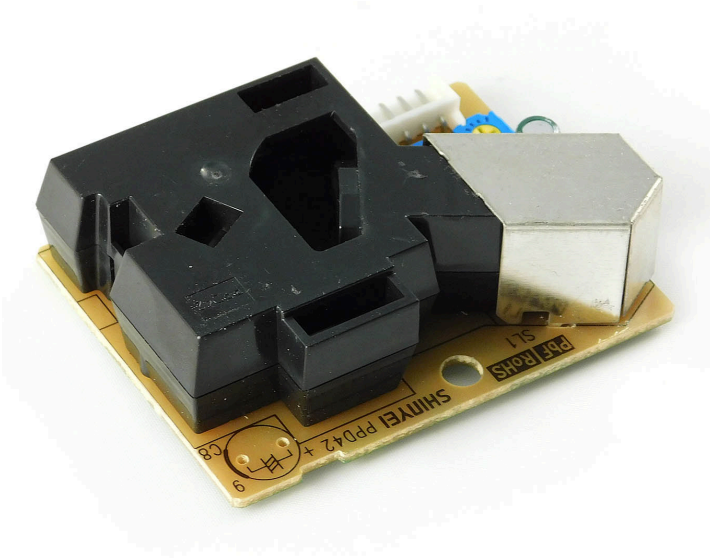
Kan kjøpes her for 120 kr

BevegelsesSensor

I produktet bruker vi en PIR Motion Sensor for å merke om det er bevegelse rund ESP'en sensoren er koblet til. Selve modellen av sensoren er uvisst ettersom den ikke er merket, men veldig lignende sensorer med samme funksjon og oppsett er lette og finne samt kan brukes om hverandre.

Bildet er hentet fra:
https://www.digikey.no/no/products/detail/adafruit-industries-llc/4871/1392205

Den kan kjøpes her for 52kr.



Støvsensor

Prosjektet benytter seg av en SHINYEI PPD42 som igjen er en sensor lånt av skolen.

Dette er som de andre en ganske robust sensor som opererer på 5 volt. Dette gjør den lett å koble opp og den trenger ikke noen ekstra komponenter for å snakke med ESP32'ene.

Bildet er hentet fra: https://nettigo.eu/products/shinyei-ppd42-air-quality-sensor

Den kan kjøpes her for rundt 120 kr

Andre komponenter

For å koble de før nevnte komponentene sammen trengs det andre mindre komponenter. Disse inkluderer resistorer, kapatisator og ledninger. Andre små komponenter som Led diode ble også brukt. Liste over dette kommer i regnskapsdelen.

Programvare

Raspberry Pi Software

Raspberry Pi'en er selve hjertet til prosjektet og trenger veldig mye software for at alt skal funke. Raspberry Pi'en er kodet i python og all koden som kjører er samlet i mappen som heter main. Her er det mye som skal skje. Først må lyden som kommer inn å styrer resten av systemet tas opp. Dette gjøres ved bruk av et python modul som heter pyaudio. Deretter sendes dataene videre over til serveren.

Serveren ser vi på senere, det vi skal fokusere mer på nå er kommando behandling når lyden er transformert til tekst av serveren.

Kommando behandlinger er delt inn i to deler. Det som trenger mer data og det som allerede er nok data som bare endres litt.

Kommandoene som ikke trenger mer data blir skrevet om til språk som høres mer naturlig ut, samt siste avrundinger på tall blir gjort. Etter at kommandoen har blitt gjort om til tekst som høres mer naturlig ut blir det sendt videre til talesyntese-modulen.

Talesyntese-modulen består for det meste av google text to speech som er en gratis talesyntese API. Det eneste problemet med bruk av denne er at de krever internett, noe som vi har som mål å unngå. Dermed har jeg laget et cache system hvor setninger som er repetitive kan lagres så vi unngår å be API'et om samme tekst flere ganger. Eksempel på data som kan være cachet er ting som “klokka er” og “temperaturen er – grader celcius”. For simple spørsmål vil cach systemet dekke teksten for repetitive kommandoer. Tall blir også cachet. Tallene er cachet på en slik måte at de er delt opp i alle tall under hundre, tusen, millioner og milliard. Dette gjør at av å sette sammen tall så kan den uten internett si alle tall under 10**11 noe som er mer enn bra nok for daglig bruk. For tall over 10**11 blir tallene delt opp, 10**12 blir dermed til “tusen milliarder” i stedet for “billion”. Cachen over alle tall pluss noen grunnleggende kommandoer er på 419 KB noe som knapt merkes på det 32GB store minnekortet.

Når det gjelder kommandoer som trenger mer data, må disse sendes videre. Ettersom kommandoene kommer fra serveren er det eneste stedet der det er å hente data, på forskjellige ESP’er. Raspberry pi’en kan dermed over local socket be de forskjellige espene om data’en sin for å så igjen gå tilbake til kommando behandling for å gjøre teksten mer naturlig før det til slutt blir sendt til talesyntese modulen og helt til slutt spilt av på høyttalerne.

ESP32

ESP32’ene er en viktig del av assistenten. Alle ESP’ene har spesifikk kode for å kunne si hvordan sensor de har og for å kunne lese av sensoren og få riktige verdier, men mye av koden er fortsatt lik. Spesielt i startfasen oppfører ESP’ene seg helt nesten helt like. De begynner med å teste om de kan bruke dataen lagret i EEROM for å koble til det lokale nettverket. Hvis dette ikke funker, går de videre til å bli en bluetooth-server som venter på tilsendt data. Når brukeren av produktet da senere sier “oppstart” vil raspberry pi’en starte en en søk etter alle esp’ene i nærheten og sende de SSID’en og passordet til det lokale nettverket. Denne dataen blir sendt over bluetooth og er også kryptert med svak kryptering. Dette er for å unngå at en ESP32 enhet som ikke er en del av produktet, men med en bluetooth server ikke skal få opp internett passordet og SSID’en i clear text. Når denne informasjonen er mottatt prøver ESP’en å koble seg til nettverket med data’en det nå har fått. Vær bevisst på at ESP32’en ikke kan koble seg til 5ghz nettverk. Hvis tilkoblingen er vellykket vil SSID’en og passordet lagres i EEROM til senere

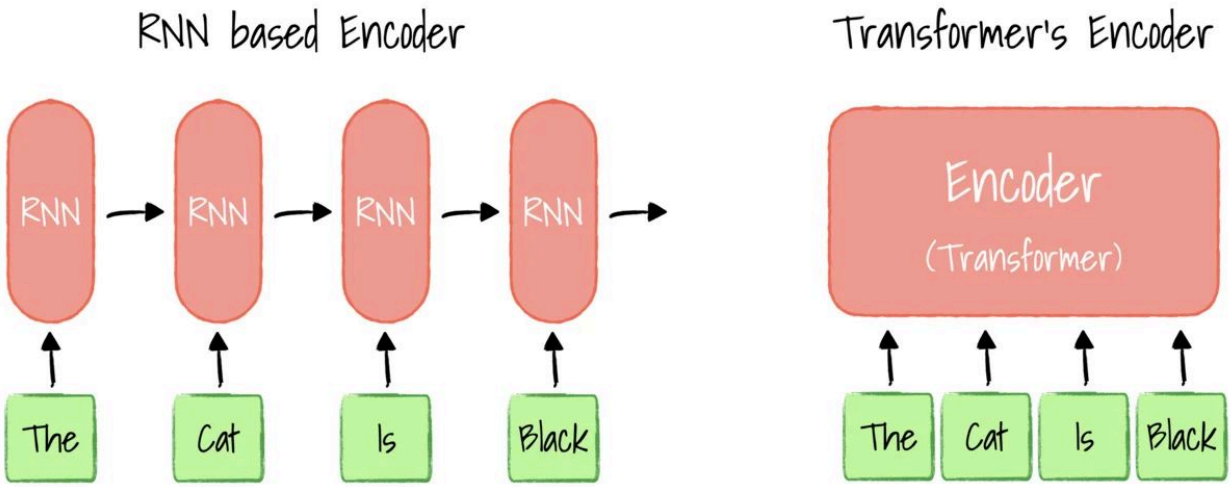
bruk.

Det er 3 hovedfunksjoner felles for alle ESP32'ene, hvis ESP'en motar 'u' over socket vil den svare tilbake med 'y'. Dette er veldig nyttig for å kunne pinge forskjellige moduler og se at de fortsatt er oppe, samt gi et si at de eksisterer under scan av nettverket. Den andre funksjonen er når den mottar 'n' over socket. Bokstaven 'n' står for Navn og dermed naturligvis sender den navnet på sensoren. Disse navnene blir lagret av Raspberry Pi'en sammen med ip'en til ESP'en så den senere kan kontakte riktig ESP når den trenger data. Denne daten får den i form av en int32 delt inn i 4 bytes som sendes når ESP'en mottar 'v' over socket.

Server

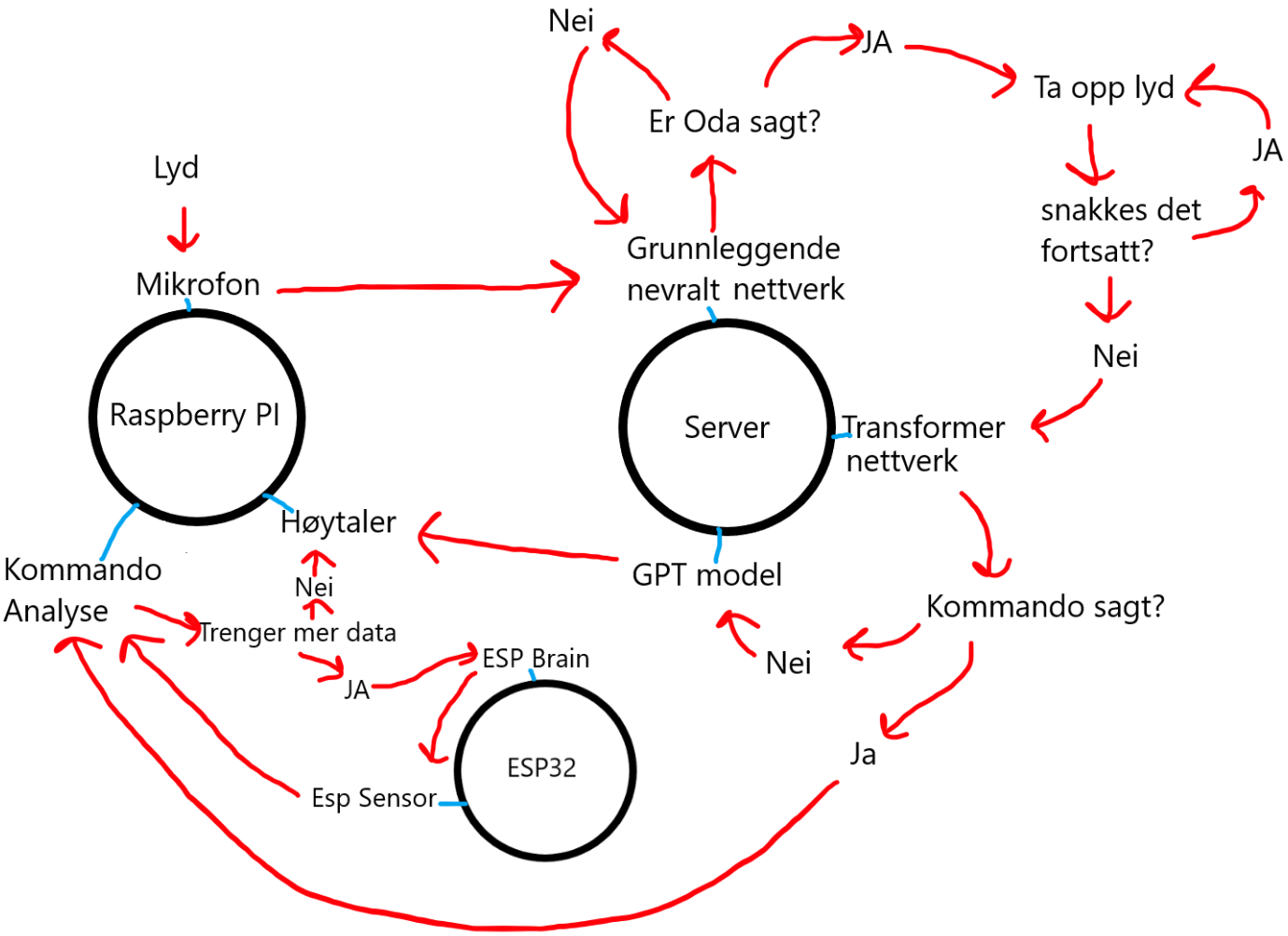
På serveren er det mye som skjer. Det meste komputalt tungt blir prosessert på serveren for å minske påvirking av raspberry pi'en.

Serveren sin jobb begynner med at raspberry pi'en sender den konstant lyd informasjon. Denne lydinformasjonen blir så behandlet av et Recurrent Nural Network (RNN). RNN'en kjører konstant med en jobb, og det er å merke når navnet til Smart Assistenten blir sagt. Standard navnet til Assistenten er Oda, fordi det er kort, tydelig, og ligner ikke på andre ord. Videre i rapporten vil jeg bruke Oda når jeg snakker om navnet til assistenten. Etter at RNN'en har merket at Oda har blitt sagt tar serveren opp alt som sies, dette symboliseres på raspberry pien med en led som lyser rødt. RNN nettverket fortsetter å ta opp helt til den merker at ingenting har blitt sagt på 3 ticks, (rundt 0.36 sekunder). Videre sendes all dataen til et Transformer Nettverk. Mer spesifikt whisper sitt medium store nettverk. Vi har gjort det slik for å få høy presisjon og hastighet, ettersom RNN'en kan raskt oppdage om Oda har blitt sagt og hvor lenge det snakkes, og Transformer nettverket bare trenger å aktiveres en gang per kommando.



Bildet hentet fra: https://jinglescode.github.io/2020/05/27/illustrated-guide-transformer/

Dataen fra transformer-nettverket blir så testet om det er en kommando eller ikke. Hvis teksten er en kommando (for eksempel “hva er klokka”, eller “hva er temperaturen i rommet”) blir det sendt videre til raspberry pi’en for behandling. Hvis det derimot ikke er en kommando blir det sendt videre til en gpt-modell fra gpt4all mer spesifikt orca-mini-3b-gguf2-q4_0. Vi har med vilje unngått å bruke api’er for å forsikre oss om at systemet funker offline også. Eksempel på tekst som kan bli sendt videre til gpt-modellen er ting som “hva er meningen med livet”. De tre før nevnte eksemplene blir alle vist fysisk i en video vedlegg til rapporten.
Orca-mini-3b-gguf2-g4 fungerer kort sagt ikke på norsk data. For å fortsatt bruke denne modellen oversetter jeg teksten til engelsk før jeg sender den gjennom før jeg så oversetter tilbake etterpå. For å gjøre dette bruker jeg google translate gjennom deep_translator modulen. Dette funker overraskende bra og merkes ikke under bruk. Under kan du også se et flyt diagram over systemet.



Datainnsamling

For å trene RNN nettverket til trengte jeg mye norsk data. Dette viste seg å være en av de vanskeligste delene av prosjektet. Det var få steder å finne god data. Med god data menes teksting av lyd/videoklipp. Den største kilden til dette jeg fant var den norske språkbank med totalt 27000 lydklipp med tilhørende tekst.

Denne dataen var veldig repetitivt og veldig varierende i klipplengde, med noen klipp på varighet under ett sekund.

Den andre muligheten jeg hadde var å hente data fra YouTube. Det er mange videoer på youtube og mange av de har også teksting. Denne informasjonen går det ann å laste ned og behandle for å gjøre om til klipp med tekst og tilhørende lyd. Jeg valgte så 251 norske youtube kanaler som ofte hadde teksting på videoene sine som hovedfokus for skriptet mitt. Disse kanalene er vedlagt som en pickled dict i div mappen. Vi kunne så få en liste over alle videoene alle kanalene noen gang hadde lagt ut og om de hadde tekst eller ikke. Etter det kunne vi gå en og en video bortover, laste ned all dataen vi trengte og gjøre det om til lyd- og tekst treningsdata. Totalt fikk vi lastet ned litt over 750000 lydklipp før jeg stoppet skriptet, noe som gjorde den totale mengden med unike lydklipp til litt over 777000. Denne dataen var fortsatt ikke helt bra. Teksting kvaliteten varierte i stor grad i nøyaktighet på hva som ble sagt i youtube videoen fra kanal til kanal. Noen ganger spesielt på tv-youtube kanaler som nrk og tv2 var det ofte engelske videoer med norsk tekst noe som skapte feil i datasettet. For å løse dette kjørte jeg dataen gjennom et text-to speech nettverk fra whisper. Mer spesifikt deres "large" nettverk, for den høyeste nøyaktigheten. Jeg testet så om dataen i outputen fra whisper var den samme som teksten jeg hadde lagret fra youtube. Hvis det var det, lagret jeg dataen. Litt over 30 % av dataen jeg gikk gjennom var lik. Ettersom whisper sin large model var ganske stor fikk jeg bare analysert rundt 3k lydklipp dagen hvordan rundt 1k var godkjent. Totalt fikk jeg filtrert fram 30k bra lydklipp før min siste treningsdag av det nevrale nettverket.

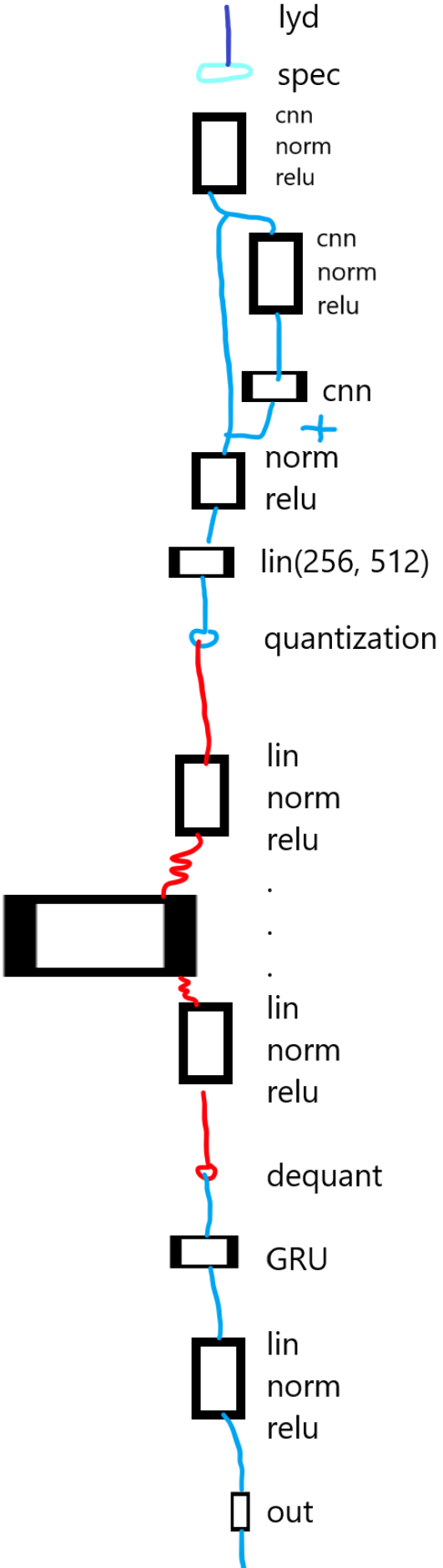
Trening

Trening av selve nettverket var ganske greit. Jeg valgte å begynne med å se på tidligere modeller brukt til samme oppgave og fant denne bloggen som virket lovende. Jeg trente en relativt lik modell som den vist i vloggen før jeg testet den på raspberry pien og fant ut at den brukte rundt 4 sekunder på å analysere 1 sekund med lyddata. Jeg gikk dermed videre til et nytt design med bare et skip layer og et stort quantizationed dense nett i midten før jeg avsluttet med et lite RNN mer spesifikt 2 layer GRU med input 512 og hidden size på 256, noe som hjelper veldig når man bruker Connectionist Temporal Classification (CTC) loss. Jeg valgte også å bruke GRU over LSTM på grunn av deres effektivitet.

Selve treningen gjorde jeg på relativt lav drop out (0.01), dette var mye på grunn av at jeg trengte et nett som utnyttet så mye som mulig av kraften den fikk, og det store datasettet. Jeg gjorde også om teksten til andre tegn, mer spesifikt brukte jeg:

['_', ' ', 'a','b','d','e','f','g','h','i','j','k','l','m','n','o','p',
'r','s','t','u','v','y','å','æ','ø','ɵ','ʂ']

alfabetet istedenfor det vanlige. Her ser du den første bokstaven er en blank (når modellen er usikker) andre er wordbreak (mellom hvert ord) , de andre bokstavene står for sine normale lyder unntatt de nye ɵ for “ng” og ʂ for “sj”, “kj”, “sjk”, “tj” lyder. Jeg fjernet også en del bokstaver som var underrepresentert i datasettet. Det jeg endte opp med da var et alfabet hvor man kunne skrive lydene i alle norske ord med færre og mer jevnt fordelte bokstaver. For mer info om modellen ligger selve class’en i model4_oneBatch.py i mappen main_server i vedlegget. Som input gjorde jeg om lyddataen til et melSpectogram for å gjøre treningen lettere. Under ser du en grov skisse av modellen.


lyd
spec
cnn
norm
relu
cnn
norm
relu
cnn
norm
relu
lin(256, 512)
quantization
lin
norm
relu
.
.
.
lin
norm
relu
dequant
GRU
lin
norm
relu
out

Regnskap

Dette er et GROVT estimat av priser. Jeg regner ikke med lønn ettersom jeg jobber gratis. Og strøm matten er grovt simplifisert.

Pris

Raspberry Pi zero W 2 kr190
https://raspberrypi.dk/en/product/raspberry-pi-zero/?src=raspberrypi

Mikrofon 20kr*2
https://he.aliexpress.com/i/4000001686978.html?gatewayAdapt=glo2isr

Høyttaler 200kr
https://he.aliexpress.com/item/1005001297858932.html?gatewayAdapt=glo2isr

ESP32 40kr*3
https://he.aliexpress.com/item/1005005970816555.html

Forsteker 12kr
https://www.amazon.in/Electronicspices-PAM8403-HW-104-Digital-Amplifier/dp/B07YX6X2BG

Termometer 120kr
https://digilent.com/shop/pmod-tmp3-digital-temperature-sensor/

Bevegelses Sensor 52kr

https://www.digikey.no/no/products/detail/adafruit-industries-llc/4871/1392205

Støv sensor 120kr

https://nettigo.eu/products/shinyei-ppd42-air-quality-sensor

Strøm 468kr

Strømpris(nov)	*	kwh brukt	*	tid	= pris
138.67 øre/kWh	*	0.5	*	24*30	= 468kr

Div småting 30kr

(Ledning 3m, 33nF capacitor, 270 ohm resistor, 150 ohm resistor, 1µF elektrolytt capacitor, tinn, led diode)

Totalpris

Raspberry pi	190kr
Mikrofoner	40kr
Høytaler	200kr
ESP32's	120kr
Forsterker	12kr
Termometer	120kr
Bevegelse	52kr
Støv	120kr
Strøm	468kr
Div Småting	30kr
—----------	
Total	1352kr
—----------	

Drøfting

Dette er første gang jeg lagde en modell som behandler lyd og det var uten tvil en lærerik erfaring. Jeg pleier å bruke TensorFlow som er python modul for trening og kjøring av nevrale nettverk, men jeg ble anbefalt Pytorch så jeg brukte det her. Og de to er veldig forskjellige i hvordan man snakker med de på. Jeg vil si at alt i alt virker det som det er mer vekst i pytroch spesielt i nyere tider, og det er mye lettere å plukke opp enn det TensorFlow er. Alt i alt minner torch meg mye mer om python og Tensorflow er har litt mer C følelse; veldig streng på typer, veldig dårlige error meldinger, alt kompiler hele tiden. Dette har sine fordeler, mest av alt hastighet. Det var også det jeg savnet mest av under denne oppgaven. Pytorch hadde en del optimaliseringsmuligheter og jeg prøvde en del av de, men kom uansett ikke i nærheten av hastighetene jeg kom når jeg brukte TensorFlow. Hvis jeg skulle delt det inn mer spesifikt vil jeg si pytorch er bedre til å trene og lage modeller på, mens TensorFlow er bedre når du først har modellen. Alt i alt hadde jeg brukt TensorFlow skulle jeg gjort denne oppgaven på nytt i dag.

Planen for oppgaven var å lage et nevralt nettverk som kjører lokalt på raspberry pi zero w 2. Dette skjedde jo ikke og det er en del grunner til det. En av de største grunnene er dårlig tilgang til hardware for å trene modellene. Jeg sover på samme rom som den stasjonære pcen, så trener ikke om natta, og strømprisene ble veldig dyre i desember, så jeg måtte stoppe å trene da. Annet enn trening av modellen var det også noen problemer med selve modellen. Mer spesifikt CTC loss. En plan jeg begynte på etter at den nåværende modellen ikke funket var å trene en framewise modell som hadde en mindre label (Trenger ikke blank lenger i framewise) . Men fortsatt beholdt wordbreak (“ “). Denne modellen kunne gitt sannsynlighet for hvilken av lydene som ble lagd for hver frame. Og dermed kunne jeg trent en til mindre modell som estimerte om to ord var like.

Dette har jeg allerede testet hvor jeg brukte CTC modell outputet til å generere framewise treningsdata. Her begynte den og overfitte til bokstavene “e”, “t”, “å” og “s” som alle dukker opp ofte i datasettet. Men ettersom ctc modellen jeg genererte treningsdata på ikke var ferdig trent og eksperimentell liten virker det som det er mye håp rundt denne ideen. Jeg testet også et lite nevralt nettverk som sammenligner to ord. Dette nettverket var primært bare fem layers; conv,

conv, biGRU, biGRU, linear, og den trente også på data generert fra den lille ctc modellen. Dette nettverket gjorde det ganske dårlig, men fortsatt overraskende bra (75%acc) med tanke på treningsdataen den trente på.

Jeg kan også lett laste ned / filtrere mer treningsdata ved å bare la scriptene så å gå etterhvert som strømmen blir litt billigere. Med alt dette i tankene tenker jeg det er meget sannsynlig mulig å trene en modell som er sterk nok til å kjøre en smart assistent på raspberry pi zero w2.

Det siste problemet jeg hadde var voltage drop og crashes, spesielt når jeg prøvde å kjøre det nevrale nettverket på raspberry pi'en. Det virker som at hvis jeg skal drive med konstant tunge kalkulasjoner og en høytaler at jeg må se litt mer på strømtilførselen, men det blir et problem for en annen gang.

Konklusjon

Alt i alt er jeg meget fornøyd med produktet. Produktet er noe jeg kommer til å bruke selv og noe jeg synes er nyttig generelt. Ikke alt gikk etter planen, men der det kom problemer fant vi nye løsninger.

Jeg har lært mye om bruk av Pytroch under prosjektet samt mye om språk og talesyntese samt talegjenkjenning og om å fullføre et prosjekt innenfor en vis tidsramme. Dette prosjektet var et to måneders prosjekt og jeg føler jeg kan være fornøyd med framgangen jeg fikk i de to månedene.

KILDER

Figur 1: https://he.aliexpress.com/item/1005001297858932.html?gatewayAdapt=glo2isr

https://blog.research.google/2019/04/specaugment-new-data-augmentation.html?m=1

https://pytorch.org/docs/stable/quantization.html

https://microdigisoft.com/esp32-wifi-manager-to-manage-ssid-and-password-using-eeprom/

https://randomnerdtutorials.com/esp32-i2c-communication-arduino-ide/

https://www.dfrobot.com/blog-994.html

https://www.nb.no/sprakbanken/

https://www.assemblyai.com/blog/end-to-end-speech-recognition-pytorch/

https://github.com/prataffel/deep_translator