NTNU
Kunnskap for en bedre verden

Institutt for elektroniske systemer

IELET1002 - Datateknikk

Hackathon 2 Arduino

Authors:
Karl Emil Ingebrigtsen
Ellen Iren Johnsen
Knut Ola Nøsen
Petter Wandsvik
Elias Loe Eritsland
Johannes Ravn Munkvold
Trond Forstrøm Christiansen

March 2, 2023

Mål for dette hackathonet:

- Forstå hvordan man kan bruke ESP-NOW
- Forstå grunnleggende hva en motordriver er og hvordan den brukes
- Anvende ulike programmeringskonsepter til et større system
- Sette seg inn i ukjent kode og videre sy sammen de ulike filene/kodesnuttene

Generelle krav:

- Hackathon 2 skal være et individuelt gruppearbeid. Det vil si at gruppene jobber hver for seg. Juks og plagiat kan du lese mer om på ntnu.no.
- Hackathon 2 gjennomføres i to deler: Del 1 består av obligatorisk oppmøte og demonstrasjon av løsningen til stud.ass på sal. Del 2 består av refleksjonsnotat.
- Oppgaven skal løses ved å, ikke skrive av, men **kopiere** og deretter **endre** eksempelkodene. Grunnen til dette er for å spare tid på feilsøking av eksempelkoden, slik at dere kan fokusere på oppgaven, som er å integrere kodefilene, gjøre forbedringer i ettertid og legge til egne ideer etter dere har fått testet at kommunikasjonen fungerer.
- Frist for innlevering av refleksjonsnotat: mandag 06/03, kl. 23.59. Se informasjon rundt refleksjonsnotat på BB.

WiFi.h

WiFi.h er et ESP32-bibliotek som følger med når du laster ned 'Arduino core for the ESP32'. Dette biblioteket lar oss koble ESPen til WiFi enten som STATION eller ACCESS POINT. Generelt bruker vi STATION, da vil ESPen fungere som en enhet som kan koble seg til en ruter.

WebServer.h

WebServer.h er et bibliotek som gjør det mulig for ESPen å være vert for HTML kode. Dette gjør at man enkelt kan lage en webside på ESPen. En eksempelkode på dette ligger i WebServerWebsite[3].

ArduinoJson.h

ArduinoJson er et bibliotek der du kan lagre data som en Json tekst. Denne strukturen gjør det enklere å sende, og dekode informasjon som vi sender mellom ESPene

ESP-NOW

ESP-NOW er en «low-power» kommunikasjonsprotokoll som likt Wi-Fi/Bluetooth opererer på 2.4 GHz båndet [2].
Protokollen tillater flere enheter å snakke direkte med hverandre «over Wi-Fi» og er et slags «peer-to-peer» nettverk. Siden nettverket har ressursbegrensinger så er det satt en maks «payload» på 250 bytes. Ønsker du å sende mer data må en annen protokoll brukes.
For å oppnå kommunikasjon mellom enhetene må de først pares ved bruk av den unike MAC-adressen hver enhet har [1]. Denne MAC adressen kan finnes ved hjelp av Wifi-biblioteket. Et enkelt eksempel kan ses i listing 1. Tabell 1 viser noen funksjoner fra ESP-NOW biblioteket.

```
#include <WiFi.h>
void setup(){
  Serial.begin(115200);
  Serial.print("MAC-Addresse til ESP32:  ");
  Serial.println(WiFi.macAddress());
}
```

Listing 1: Hvordan finne MAC-adresse.

esp_now_init()	Initialiserer ESP-NOW. MERK! WiFi må være initialisert først
esp_now_add_peer(MAQ-adresse)	Denne funksjonen parer sammen ESPer
esp_now_send (mottakeradresse, (uint8_t *) &melding, sizeof(melding))	Send data med ESP-NOW
esp_now_register_send_cb(funksjonsnavn)	Registrerer en callback-funksjon som blir kjørt når data blir sendt fra ESPen.
esp_now_register_rcv_cb(funksjonsnavn)	Registrerer en callback-funksjon som blir kjørt når data blir mottatt til ESPen.

Table 1: Funksjoner fra esp-now

Serial1

ESPen har flere kanaler for Serial kommunikasjon. Dette kan være nyttig for å koble til ulike sensorer med serial kommunikasjon. I eksempelkodene WebServerSerial og Router setter vi opp en ny Serial Port (Serial1) for kommunikasjonen mellom ESPene[3]. Grunnen til at vi bruker en ny Serial Port i stede for den vanlige Serial er fordi vi ønsker å sende debug meldinger til den vanlige porten som er tilkoblet USB porten, og protokoll meldinger til Serial1 uten at de skal påvirke hverandre.

Eksempelkode

All Eksempelkode som blir brukt i oppgaveteksten er tilgjengelig på dette **Github-repoet** [3].

Oppgaver

Etter å ha implementert en fungerende løsning for de håndholdte viftene, ønsker Arne å utvide løsningen slik at han kan fjernstyre viften. Siden han er interessert i trådløs kommunikasjon og ikke liker å rote med ledninger, tenker han at et trådløst system burde gjøre susen. Dette betyr at han må finne noe annet enn Arduino Uno til styring.
Heldigvis har Adrian en løsning: ESP32. Siden Arne har lite erfaring med ESP trenger han derfor hjelp. Dere skal derfor utvikle et trådløst peer-2-peer nettverk ved bruk av 3 + 1 ESP32. De forskjellige funksjonalitetene fra hackathon 1 skal integreres på hver sin trådløse node, slik det er illustrert i figur 1. De ulike funksjonalitetene skal være som følger:

- En ruter
- En web server
- En fjernkontroll
- En vifte

Ruteren og web serveren utgjør til sammen huben. Web serveren som koples til ruteren via UART skal vise nødvendig systeminformasjon i nettleseren mens fjernkontrollen kontrollerer viftehastighet på vifte-noden.

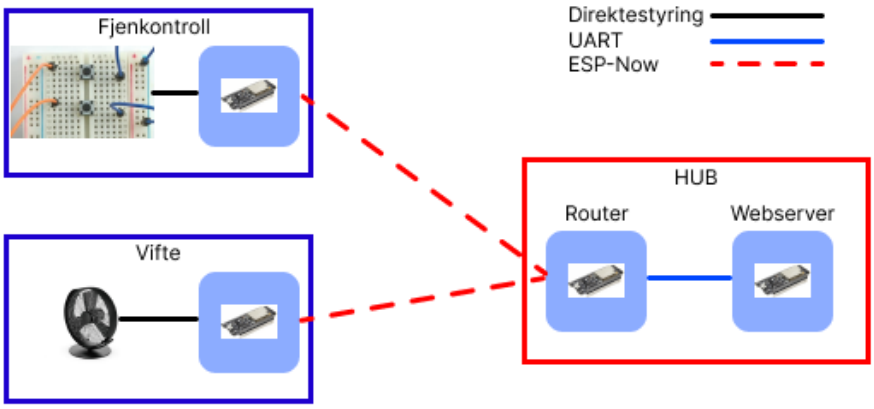


Figure 1: ESP32 Nodediagram

1 MAC-adresser

Før dere begynner må dere finne MAC-adressene til ESPene dere skal bruke ettersom dere trenger de senere. Pass på og få med dere hvilken MAC adresse som tilhører hvilken ESP.
For å finne MAC-adressen til ESPene kan dere kopiere **denne** koden og kjøre den på ESPene dere ønsker vite MAC-adressen til.
Skriv ned MAC-adressene dere finner.
Husk å stille inn Serial monitoren til 115200, og ikke 9600

2 ArduinoJson

a)

Dere skal nå teste ut ArduinoJson biblioteket. Last ned biblioteket ved å gå til **Tools − > Manage Libaries...**, søke etter **Arduinojson** og trykke **INSTALL**.
Last opp **denne** eksempelkoden på en ESP32 og se på outputen i seriemonitoren.
Prøv å endre på verdiene for å se hvordan resultatene forandrer seg. Legg så til et nytt element **"voltage" = 3.3**

b)

Se på **dette** eksempelet for avlesning av json fra serial, og prøv å forstå hva den gjør. Test koden ved å laste den opp til en ESP32 og lime inn resultatene dere fikk fra deloppgave a) i serie monitoren.

3 Test ferdigkode

Dere skal nå teste at kommunikasjonen med eksempelkodene fungerer som de skal. Last opp kodeeksemplene: **Remote.ino**, **Fan.ino**, **Router.ino** og **WebServerSerial.ino** til hver sin ESP32.
For og sende data mellom ESP-Now routeren og WebServeren må disse kobles sammen, det gjøres slik:

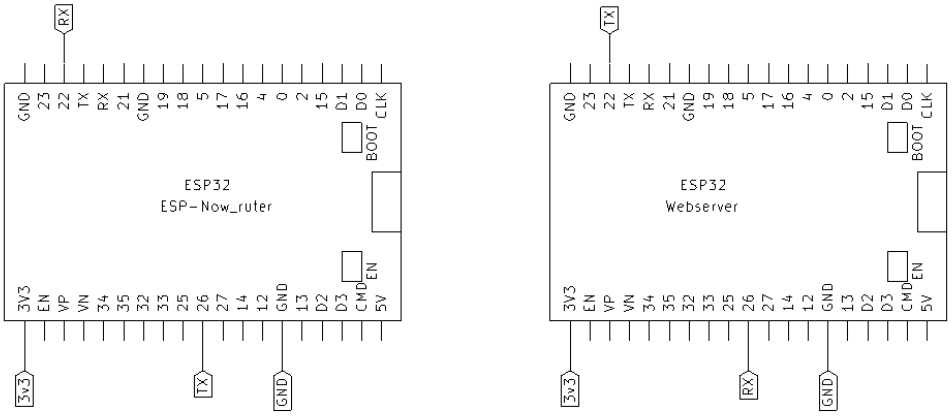


Figure 2: ESP32 UART oppkobling

Merk at her krysses ledningene på RX og TX mellom enhetene så pin 22 på ruteren går til pin 26 på webserveren, og pin 26 på ruteren går til pin 22 på webserveren

Med alle kodeeksemplene lastet opp på hver sin ESP, og alle ESPene på samtidig vill man kunne følge med på Serial Monitoren fra de forskjellige nodene og se at meldingene som sendes blir printet.

Remote sender en melding til Router. Sjekk at Router mottar kommandoene fra Remote ved å lese det som kommer ut av Serial Monitoren til Router.

Router videresender meldingen fra Remote til Fan. Meldinger fra Remote skal inneholde feltene "sender"="remote" og "target"="fan". Se på Serial Monitoren til Fan og sjekk at meldingen blir mottatt.

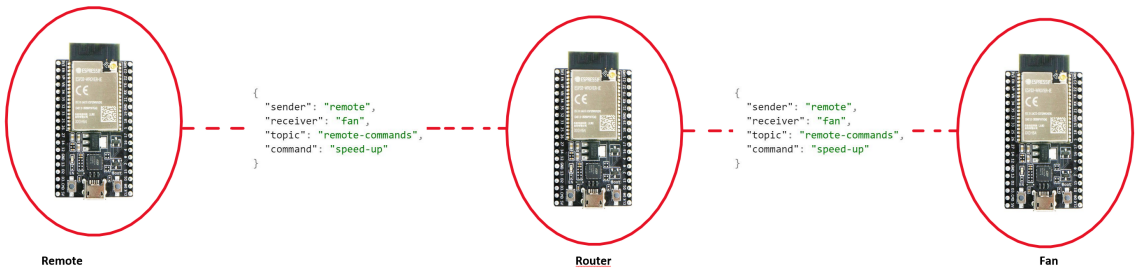


Figure 3: ESP32 kodeflyt fra remote til fan

Når Fan mottar en kommando for å endre farten, vil den også sende en melding tilbake til Router, med feltet "target"="webserver". Sjekk i Serial Monitoren til Router og WebServer at meldingen blir mottatt.

Her er det viktig at dere husker å sette mottaker MAC-adressene til MAC-adressene dere fant i oppgave 1

4 Fjernkontroll

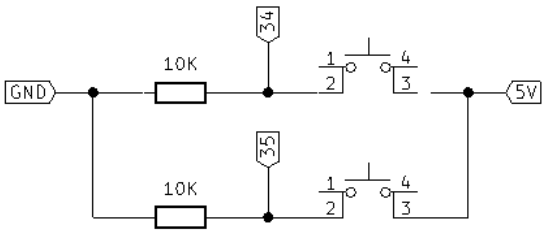


Figure 4: L293DPinout

a)

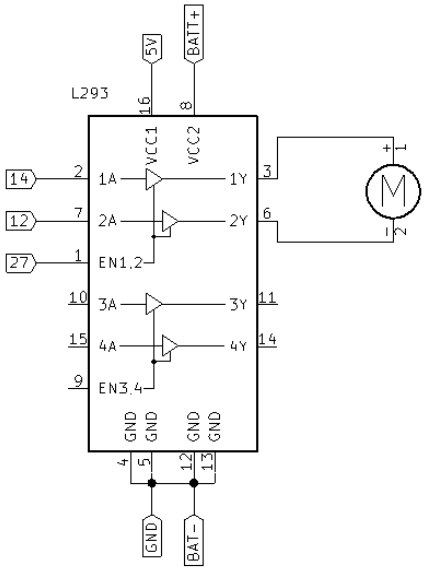
Fjernkontroll noden skal bestå av en ESP32 og to knapper. Ta kretsen for knappene fra Hackaton 1 og koble den opp til en ESP32. Denne kretsen skal stå på sitt eget koblingsbrett separat fra resten av systemet. Kopier koden for knappetrykk fra **løsningsforslaget på Hackaton 1** inn i **Remote.ino** og test at både kode og oppkobling fungerer ved og printe resultatene fra knappetrykene til Serial Monitoren. I denne deloppgaven skal dere kun teste at knappkoden og oppkoblingen fungerer.

b)

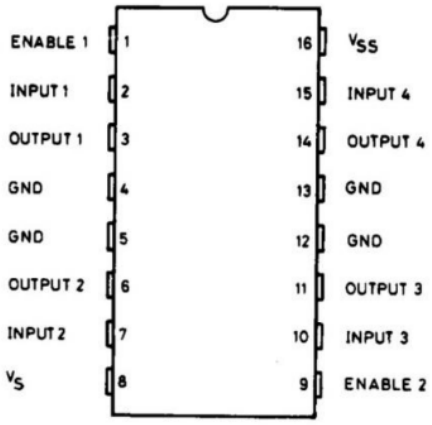
Remote.ino inneholder kode i void loop() som sender kommandoer over ESP-NOW. Nå skal dere modifisere koden slik at de respektive kommandoene sendes 1 gang når knappene trykkes.

Her er det viktig at dere husker å sette mottaker MAC-adressen til MAC-adressen til Router dere fant i oppgave 1

5 Vifte



(a) Motordriver oppkobling



(b) L293d Pinout

a)

Vifte-noden skal bestå av en ESP32, en L293D motordriver og en motor. Denne kretsen skal være lik den dere brukte i hackaton 1 så dere kan ta den og flytte den til en separat ESP32. I likhet med fjernkontroll-noden skal denne kretsen stå på et separat koblingsbrett fra resten av systemet. Kopier koden for motorstyringen fra **løsningsforslaget på Hackaton 1** og test at både kobling og kode fungerer.

> Om det er problemer med å få god tilkobling til motoren er en mulig løsning og putte ledningen fra motoren i samme hull som en jumper wire. Om dere gjør dette pass på at andre ende av ledningene ikke kortslutter med noe annet.

b)

Sett inn koden for viften med **Fan.ino**. Resultatet av disse to kodene skal være et program som leser av dataen den mottar fra Fjernkontrollen for å øke farten på motoren hvis den mottar "speed-up" og senke farten hvis den mottar "speed-down".

Her er det viktig at dere husker å sette mottaker MAC-adressen til MAC-adressen til ESP-Now huben dere fant i oppgave 1

6 Webserver

Dere skal nå sette opp en ESP32 med et aksesspunkt og en nettside som mottar data over Serial1 og fremstiller den på nettsiden.

> På grunn av at det er noen små forskjeller på de forskjellige ESP32 utviklingskortene er ikke Serial1 tilgjengelig på alle versjoner. Koden for ruter og webserver er utviklet og testet for **ESP32-WROVER-E DevKitc V4** (de som kom i kitet fra Elektra), og det er derfor sterkt anbefalt at dere bruker disse

a)

last opp eksempelkoden for aksesspunkt på ESP og test at koden fungerer som den skall. Eksempelkoden er tilgjengelig på github som **WebServerWebsite.ino**.

b)

Slå sammen kodene **WebServerWebsite.ino** og **WebServerSerial.ino** slik at det ferdige programmet venter på mottatt data på `Serial1`, leser av dataen, og oppdaterer nettsiden med den mottatte motorhastigheten.

7 Ekstraoppgave

Noen ganger er det en fordel og vite om man har klart og sende dataen eller ikke. Koble til en rød og en grønn led på både fjernkontroll noden, og vifte noden.

utvid `void onDataTransmitted()` funksjonene på begge nodene slik at den grønne LEDen blinker hvis datasendingen er suksessfull, og den røde LEDen blinker hvis sendingen feiler.

Referanser

[1] Martin Haverholm. *Slik finner du MAC-adressen.* URL: https://komputer.no/internett/nettverk/slik-finner-du-mac-adressen. (Sjekket: 23.02.2023).

[2] Adrian Heyerdahl. *Intro til ESP32 og tingenes internett (IoT)*, p. 20. (Sjekket: 23.02.2023).

[3] Knut Ola Nøsen. *Github: Kode for Hackathon 2.* URL: https://github.com/nosknut/arduino-course-v2023/blob/main/IELET1002/ArduinoHackathon2/links.md. (Sjekket: 1.03.2023).