NTNU

elsys
ELEKTRONISK SYSTEMDESIGN & INNOVASJON

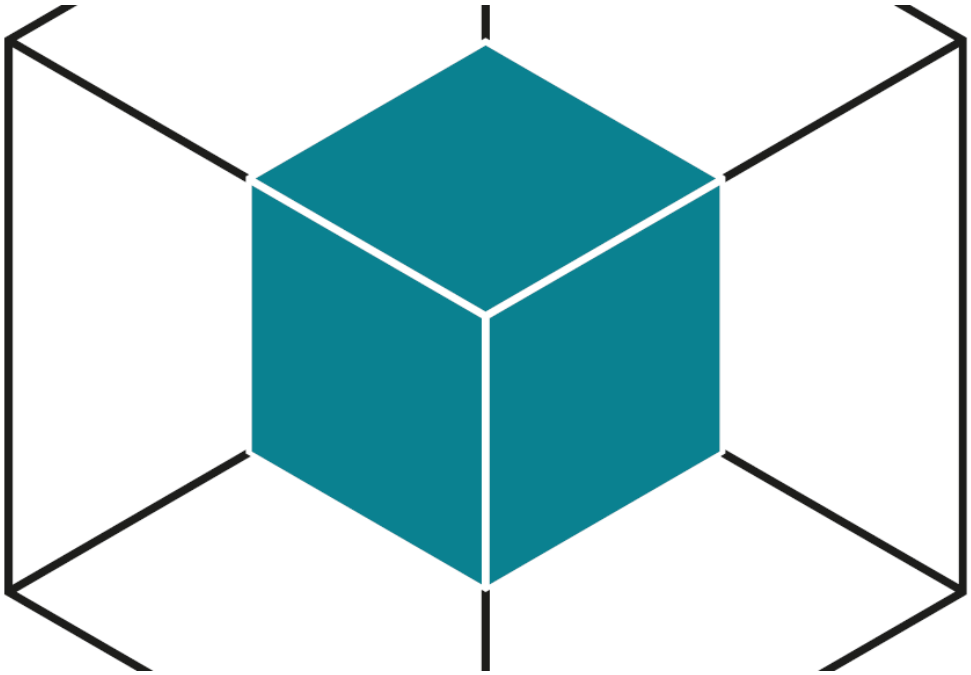
TTT4255 Elektronisk systemdesign, grunnkurs

P2: Kommunikasjon mellom to ESP32

Elektronisk systemdesign og innovasjon

Ida Bjørnevik, Sven Amberg, Amalie Fridfeldt Hauge og Peter Magerøy

29.06.2023

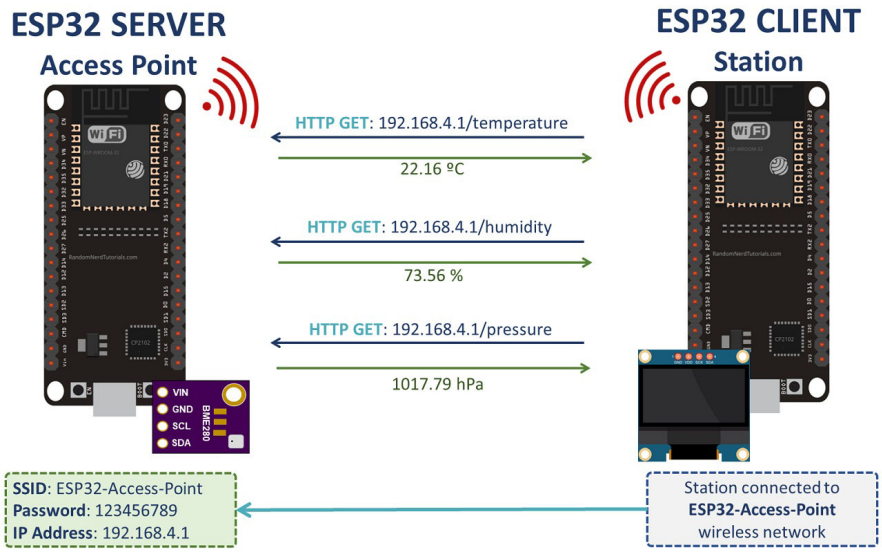


Innhold

Introduksjon

Advarsel: Ikke testet for ESP32-S2. Disse instruksjonene er ikke skrevet for deres versjon av ESP, og kan derfor være utfordrende.

I dette prosjektet skal vi bruke wifi for sette opp kommunikasjon mellom to ESP32. Den eine ESP32 skal hente data frå ein sensor, denne dataen blir send til den andre ESP32 som viser fram dataen på ein skjerm.



Figur 1: Kommunikasjon over wifi mellom to ESP32.

Bakgrunnsinformasjon (ikkje nødvendig å kunne)

Ein ESP32 blir satt opp som server. Den opprettar sitt eige trådlause nettverk (ESP32 Soft-Access Point). Den andre ESP32, klienten, kan så kople seg til det nettverket (SSID: ESP32-Access-Point, Passord:123456789). Klienten sender så HTTP GET-førespurnadar til serveren for å be om sensordata. Dette gjer den ved å bruke IP-adressa til serveren.

Serveren lyttar etter innkomne førespurnadar og sender svar som inneheld sensordata-en. Klienten mottek målingane og viser dei på OLED-skjermen.

Koden i denne modulen er i stor grad henta frå `https://randomnerdtutorials.com/esp32-client-server-wi-fi/`. For å unngå å skrive av mykje kode kan du gå inn på nettsida og hente ut kode derfrå, legg merke til at der ligg mykje overflødig kode vi ikkje treng sidan vi berre brukar ein sensor.

Prosjektet

Utstyrsliste

- 2 ESP32
- Digital temperatur sensor (DS18B20)
- OLED-skjerm

Steg 1: Installere nødvendige bibliotek

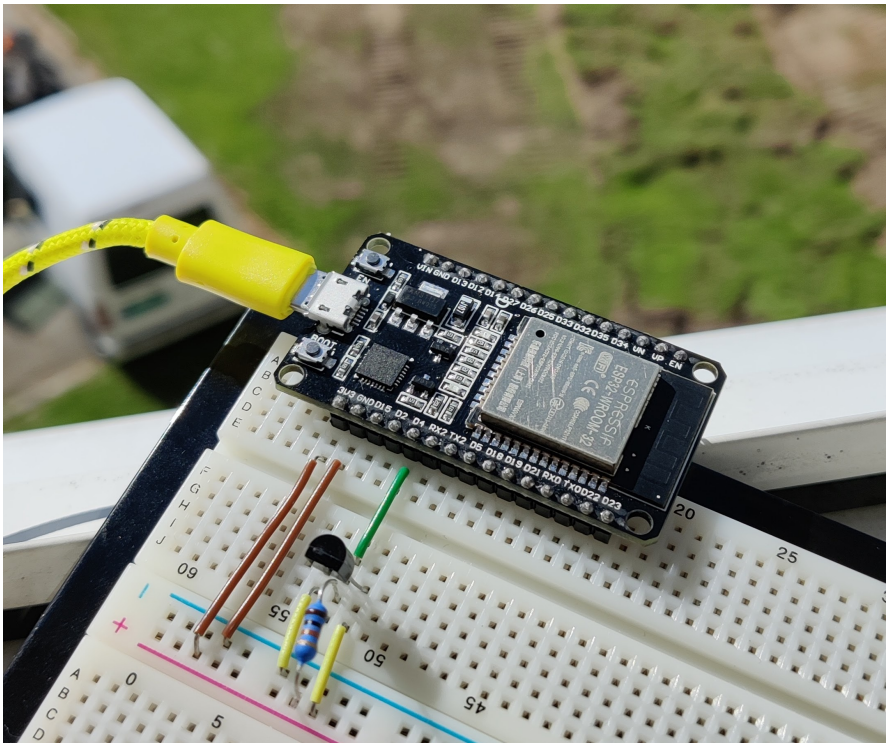
For å sette opp serveren må vi først intallere to bibliotek.

Installere bibliotek

1. Last ned zip-fil frå `https://github.com/me-no-dev/ESPAsyncWebServer` og `https://github.com/me-no-dev/AsyncTCP` ved å trykke på "CODE" og deretter "Download ZIP".
2. Opne Arduino IDE, gå i menyen **Skisse > Inkluder Bibliotek > Legg til .ZIP Bibliotek** og vel .ZIP-fila du nettop lasta ned. Gjer dette med begge .ZIP-filene.
3. Restart Arduino IDE.

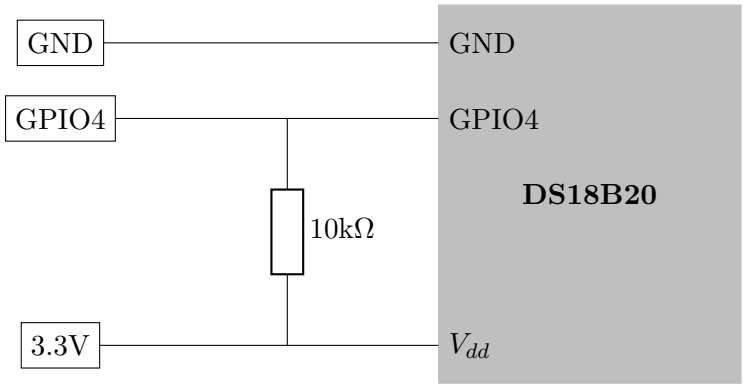
Steg 2: Server: Krets

Serverdelen av systemet består av ein ESP32 og ein temperatursensor, og er vist i figur 2.



Figur 2: ESP32-server med temperatursensor.

Temperatursensoren koplast til ESP32 på samme måten som i modul S3, vist i figur 3.



Figur 3: Kretsskjema temperatursensor.

Steg 3: Server: Programmering

I begynnelsen, sjå figur 4, av koden inkluderer vi nødvendige bibliotek og deklarerer variablane som blir brukt i programmet. Nedst lagar vi også ein funksjon (dette lærer dykk meir om

i ITGK), denne funksjonen hentar ut sensordata og konverterar det til ein tekststreng som serveren kan sende til den andre ESP32.

```
// Import required libraries
#include "WiFi.h"
#include "ESPAsyncWebServer.h"

#include <OneWire.h>
#include <DallasTemperature.h>

// Set your access point network credentials
const char* ssid = "ESP32-Access-Point";
const char* password = "123456789";

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

// GPIO where the 18B20 is connected to
const int oneWireBus = 4;

// Setup a oneWire instance to communicate with any OneWire
   devices
OneWire oneWire(oneWireBus);

// Pass our oneWire reference to Dallas Temperature sensor
DallasTemperature sensors(&oneWire);

String readTemp() { //Dette er ein funksjon som hentar
   temperaturen som ein tekststreng.
  sensors.requestTemperatures();
  return String(sensors.getTempCByIndex(0));
}
```

Figur 4: Inkludering av bibliotek og variabeldeklarasjon

I setup-funksjonen, sjå figur 5, startar vi opp wifi-nettverket og serveren.

```
    void setup(){
      // Serial port for debugging purposes
      Serial.begin(115200);
      Serial.println();

      // Setting the ESP as an access point
      Serial.print("Setting AP (Access Point)...");
      // Remove the password parameter, if you want the AP (Access
         Point) to be open
      WiFi.softAP(ssid, password);

      IPAddress IP = WiFi.softAPIP();
      Serial.print("AP IP address: ");
      Serial.println(IP);

      server.on("/temperature", HTTP_GET, [](AsyncWebServerRequest
         *request){
        request->send_P(200, "text/plain", readTemp().c_str());
      });

      bool status;

      sensors.begin();

      // Start server
      server.begin();
}
```

Figur 5: Setup-funksjonen

Loop-funksjonen skal stå tom, sjå figur 6.

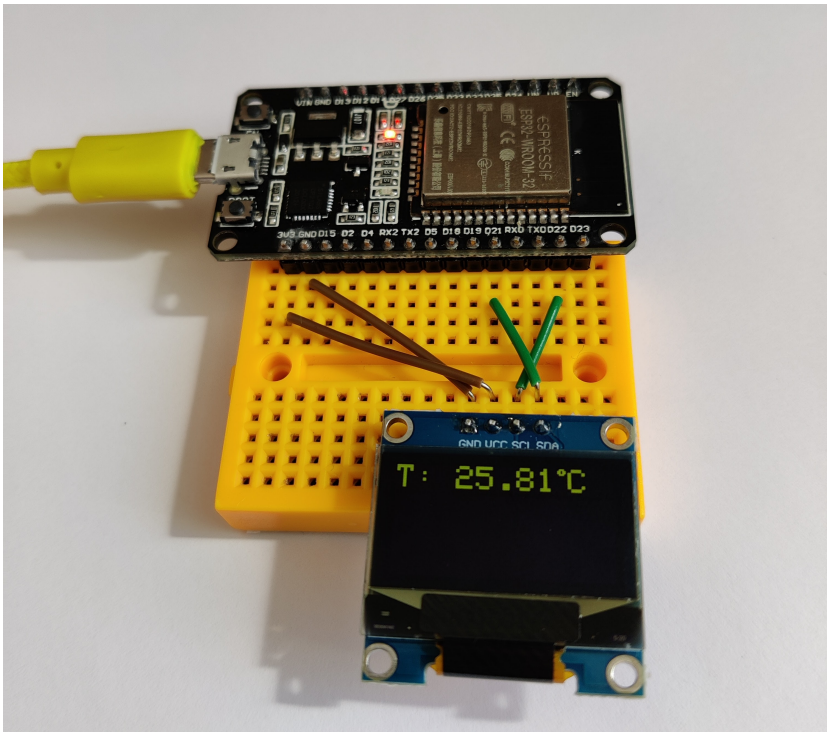
```
    void loop(){

    }
```

Figur 6: Loop-funksjonen

Last så opp koden på ESP32, for å sjekke om WiFi-oppsettet fungerte kan du sjå om nettverket dukkar opp i lista over wifi-nett på PC eller mobil. Dersom det gjer det kan du kople deg til nettverket og opne `192.168.4.1/temperature`. Dersom alt er kopla rett skal sensordataen vise her.

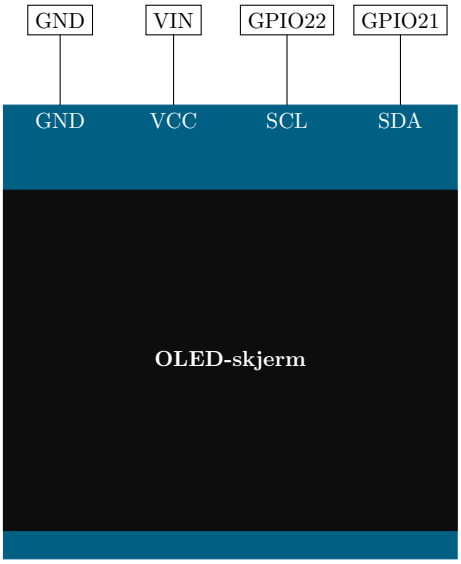
Steg 4: Klient: Krets

Klientdelen av systemet består av ein ESP32 og ein OLED-skjerm, og er vist i figur 7.



Figur 7: ESP32-klient med OLED-skjerm.

OLED-skjermen koplast til ESP32 same måte som i modul A4, vist i figur 8.



Figur 8: Kretsskjema OLED-skjerm.

Steg 5: Klient: Programmering

I første delen av koden (figur 9) inkluderer vi to bibliotek for å sette opp ESP32 som klient. Deretter deklarerer vi variablar som vi brukar seinare i programmet.

```
#include <WiFi.h>
#include <HTTPClient.h>

const char* ssid = "ESP32-Access-Point";
const char* password = "123456789";

//Your IP address or domain name with URL path
const char* serverNameTemp = "http://192.168.4.1/temperature";

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA,
   SCL pins)
#define OLED_RESET     -1 // Reset pin # (or -1 if sharing
   Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
   OLED_RESET);

String temperature;

unsigned long previousMillis = 0;
const long interval = 5000;
```

Figur 9: Inkludering av rette bibliotek og variabeldeklarasjonar.

I setup-funksjonen startar vi opp seriellovervåkaren og OLED-skjermen. Det er også her vi koplar til WiFi frå den andre ESP32.

```
void setup() {
  Serial.begin(115200);

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }
  display.clearDisplay();
  display.setTextColor(WHITE);

  WiFi.begin(ssid, password);
  Serial.println("Connecting");
  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to WiFi network with IP Address: ");
  Serial.println(WiFi.localIP());
}
```

Figur 10: Setup-funksjonen.

I loopfunksjonen hentar vi data som serveren sender ut, vi skriv så den dataen til skjermen.

```
void loop() {
  unsigned long currentMillis = millis();

  if(currentMillis - previousMillis >= interval) {
     // Check WiFi connection status
    if(WiFi.status()== WL_CONNECTED ){
      temperature = httpGETRequest(serverNameTemp);

      Serial.println("Temperature: " + temperature);

      display.clearDisplay();

      // display temperature
      display.setTextSize(2);
      display.setTextColor(WHITE);
      display.setCursor(0,0);
      display.print("T: ");
      display.print(temperature);
      display.print("");
      display.setTextSize(1);
      display.cp437(true);
      display.write(248);
      display.setTextSize(2);
      display.print("C");

      display.display();

      // save the last HTTP GET Request
      previousMillis = currentMillis;
    }
    else {
      Serial.println("WiFi Disconnected");
    }
  }
}
```

Figur 11: Første del av loop-funksjonen.

```
String httpGETRequest(const char* serverName) {
  HTTPClient http;

  http.begin(serverName);

  // Send HTTP POST request
  int httpResponseCode = http.GET();

  String payload = "--";

  if (httpResponseCode>0) {
    Serial.print("HTTP Response code: ");
    Serial.println(httpResponseCode);
    payload = http.getString();
  }
  else {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
  }
  // Free resources
  http.end();

  return payload;
}
```

Figur 12: Siste del av loop-funksjonen.

Last opp koden og sjå kva som skjer, opne seriellovervåkaren for å sjå om ESP32-klient klarer å kople seg til nettverket ESP32-server sender ut. Spør kvarandre og læringsassistentane om hjelp dersom dykk står fast!