### **Del 1: Raspberry Pi-kode (C++)**

Denne del læser sensordata og sender det til ESP32 via en seriel forbindelse.

| #include <wiringPi.h> #include <wiringPiI2C.h> #include <iostream> #include <unistd.h> #include <iomanip> #include <sstream> #include <string> #include <wiringSerial.h>  // Konstanter til sensor og seriel kommunikation const int HTU21\_ADRESSE = 0x40; const int TEMP\_MÅL\_KOMMANDO = 0xE3; const int HUMID\_MÅL\_KOMMANDO = 0xE5; const std::string SERIEL\_PORT = "/dev/ttyS0"; // Seriel port forbundet til ESP32 const int BAUD\_RATE = 9600;  // Funktion til at initialisere sensoren int initSensor(int adresse) {  int fd = wiringPiI2CSetup(adresse);  if (fd == -1) {  std::cerr << "Sensorinitialisering mislykkedes." << std::endl;  }  return fd; }  // Funktion til at læse og konvertere temperatur og luftfugtighed float læsTemperatur(int fd) { /\* (samme som i tidligere pseudokode) \*/ } float læsLuftfugtighed(int fd) { /\* (samme som i tidligere pseudokode) \*/ }  int main() {  int fd = initSensor(HTU21\_ADRESSE);  if (fd == -1) return -1;   int serial\_fd = serialOpen(SERIEL\_PORT.c\_str(), BAUD\_RATE);  if (serial\_fd == -1) {  std::cerr << "Kunne ikke åbne seriel port." << std::endl;  return -1;  }   while (true) {  // Læs temperatur og luftfugtighed  float temperatur = læsTemperatur(fd);  float luftfugtighed = læsLuftfugtighed(fd);   // Formater data som en JSON-streng  std::ostringstream dataStream;  dataStream << "{ \"temperatur\": " << std::fixed << std::setprecision(2) << temperatur  << ", \"luftfugtighed\": " << luftfugtighed << " }";  std::string data = dataStream.str();   // Send data til ESP32 via seriel port  serialPuts(serial\_fd, data.c\_str());  serialFlush(serial\_fd);   usleep(2000000); // Vent 2 sekunder før næste læsning  }   return 0; } |
| --- |

### **Del 2: ESP32-kode (Arduino-stil C++)**

Denne del kører på ESP32, som læser data fra Raspberry Pi via seriel forbindelse, viser det på en tilsluttet skærm og leverer det via HTTP.

| #include <WiFi.h> #include <WebServer.h> #include <TFT\_eSPI.h> // Bibliotek til ESP32-skærm #include <ArduinoJson.h>  // WiFi-oplysninger const char\* ssid = "your\_ssid"; const char\* password = "your\_password";  // Sæt webserver op på port 80 WebServer server(80);  // Initialiser skærm og seriel TFT\_eSPI display = TFT\_eSPI(); String temperatur = ""; String luftfugtighed = "";  void setup() {  Serial.begin(9600);  WiFi.begin(ssid, password);   // Initialiser skærmen  display.init();  display.setRotation(1);   // Vent på WiFi-forbindelse  while (WiFi.status() != WL\_CONNECTED) {  delay(500);  }  Serial.println("Forbundet til WiFi!");   // Definer HTTP-rute  server.on("/", HTTP\_GET, []() {  String jsonSvar = "{ \"temperatur\": " + temperatur + ", \"luftfugtighed\": " + luftfugtighed + " }";  server.send(200, "application/json", jsonSvar);  });   server.begin(); }  void loop() {  server.handleClient(); // Håndter HTTP-forespørgsler   // Tjek om der er data tilgængelig på Serial  if (Serial.available()) {  String jsonData = Serial.readStringUntil('\n');   // Pars JSON-data  StaticJsonDocument<200> doc;  DeserializationError fejl = deserializeJson(doc, jsonData);  if (!fejl) {  temperatur = doc["temperatur"].as<String>();  luftfugtighed = doc["luftfugtighed"].as<String>();   // Vis data på skærmen  display.fillScreen(TFT\_BLACK);  display.setTextSize(2);  display.setCursor(10, 10);  display.print("Temp: ");  display.println(temperatur + " C");   display.setCursor(10, 40);  display.print("Luftfugt: ");  display.println(luftfugtighed + " %");  }  } } |
| --- |

Første kode skulle være en udviklet form fra sensor koden.

**Raspberry Pi-kode**:

* Initialiserer temperatur- og fugtsensoren samt seriel kommunikation.
* Læser temperatur- og luftfugtighedsværdier, formaterer dem som JSON (f.eks. {"temperatur": 23.45, "luftfugtighed": 56.78}) og sender data til ESP32 over seriel port.

**ESP32-kode**:

* Forbinder til WiFi og sætter en HTTP-server op.
* Lytter kontinuerligt efter JSON-data på seriel port fra Raspberry Pi.
* Parser JSON-data og udtrækker temperatur- og luftfugtighedsværdier.
* Viser disse værdier på en tilsluttet skærm (f.eks. TFT eller OLED) og gør dem tilgængelige på webserveren.
* Når en webklient tilgår ESP32’s IP-adresse, får de en JSON-svar med de nyeste temperatur- og luftfugtighedslæsninger.