Documentatie Tehnica: Termometru Smart Web (IoT)

1. Introducere

Acest proiect abordeaza nevoia de monitorizare a parametrilor ambientali (temperatura si umiditate) de la distanta, intr-un mod accesibil si usor de utilizat. Intr-un context in care sistemele smart home devin tot mai raspandite, solutia propusa ofera o alternativa DIY (Do It Yourself) robusta, separand partea de achizitie de date de partea de conectivitate.

Problema: Monitorizarea temperaturii intr-o incapere necesita de obicei prezenta fizica pentru a citi un termometru clasic sau utilizarea unor sisteme comerciale costisitoare si inchise (proprietary).

Solutia: Am dezvoltat un sistem distribuit bazat pe doua microcontrollere:

1. Arduino Mega: Responsabil strict pentru interfatarea cu senzorii si procesarea primara a datelor (fiabilitate ridicata la nivel hardware).
2. ESP32: Responsabil pentru conectivitatea Wi-Fi si gazduirea serverului Web (interfata grafica).

Obiective atinse (High Level):

- Achizitia in timp real a datelor de temperatura si umiditate.
- Transmiterea securizata a datelor intre module prin protocol serial (UART).
- Afisarea datelor intr-o interfata web moderna (HTML5/CSS3), accesibila de pe orice telefon sau laptop conectat la aceeasi retea.
- Actualizarea dinamica a valorilor fara reincarcarea paginii (AJAX).

2. Arhitectura Hardware

In acest capitol sunt descrise componentele fizice utilizate si modul in care acestea interactioneaza electric. Sistemul este modular, permitand inlocuirea usoara a senzorilor sau a modulelor de comunicatie.

Lista Componentelor:

- Placa de dezvoltare Arduino Mega 2560: "Creierul" de achizitie. A fost aleasa pentru numarul mare de porturi Serial Hardware, ceea ce permite debug simultan cu transmiterea datelor.
- Placa de dezvoltare ESP32 (Dev Module): Modulul de comunicatie Wi-Fi si Web Server.

- Senzor DHT11: Un senzor digital compozit pentru masurarea temperaturii si umiditatii relative.
- Fire de conexiune (Jumper wires) si Breadboard.

Detalii despre Conexiuni si Pinout:

Conexiunea dintre cele doua placi se realizeaza prin protocolul UART (Universal Asynchronous Receiver-Transmitter). Arduino Mega trimite date, iar ESP32 le receptioneaza.

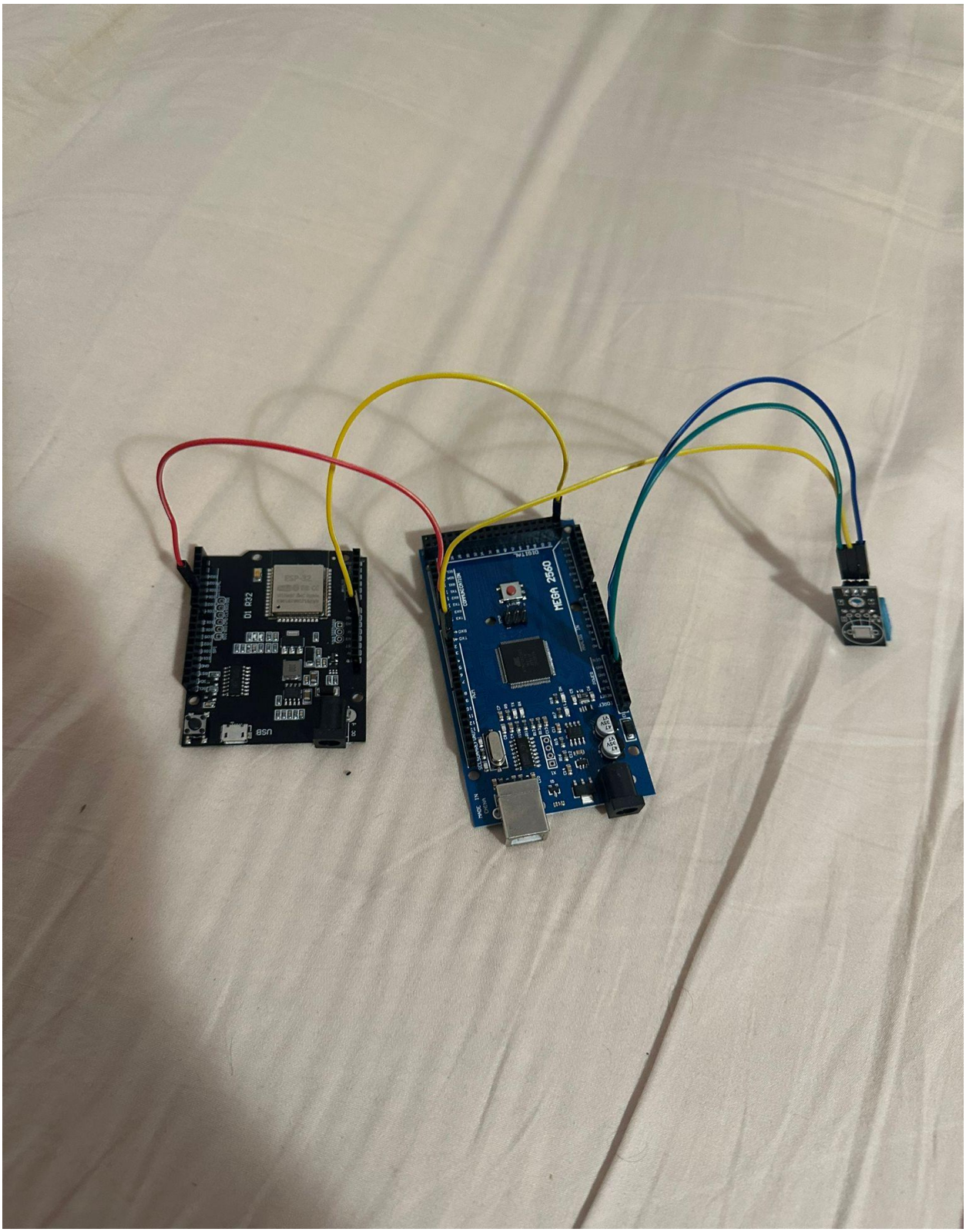
1. Conectarea Senzorului DHT11 la Arduino Mega:

- VCC: 5V
- GND: GND
- DATA: Pin Digital 2 (Definit ca DHTPIN in cod).

2. Conexiunea Seriala (Inter-board): Am folosit Serial3 pe Arduino Mega si Serial2 pe ESP32.

- Arduino Mega TX3 (Pin 14) -> ESP32 RX2 (Pin 16): Aceasta este linia prin care Mega trimite datele masurate catre ESP32.
- GND (Mega) -> GND (ESP32): Obligatoriu pentru a avea o referinta comuna de tensiune.

Nota tehnica: Nu a fost necesara conectarea liniei RX a Arduino-ului la TX-ul ESP-ului, deoarece comunicarea este unidirectionala (Mega doar transmite, nu primeste comenzi de la web server in aceasta versiune).


MEGA 2560

3. Descriere Software

Partea de software este impartita in doua entitati distincte care comunica asincron.

A. Codul pentru Arduino Mega (Firmware de Achizitie)

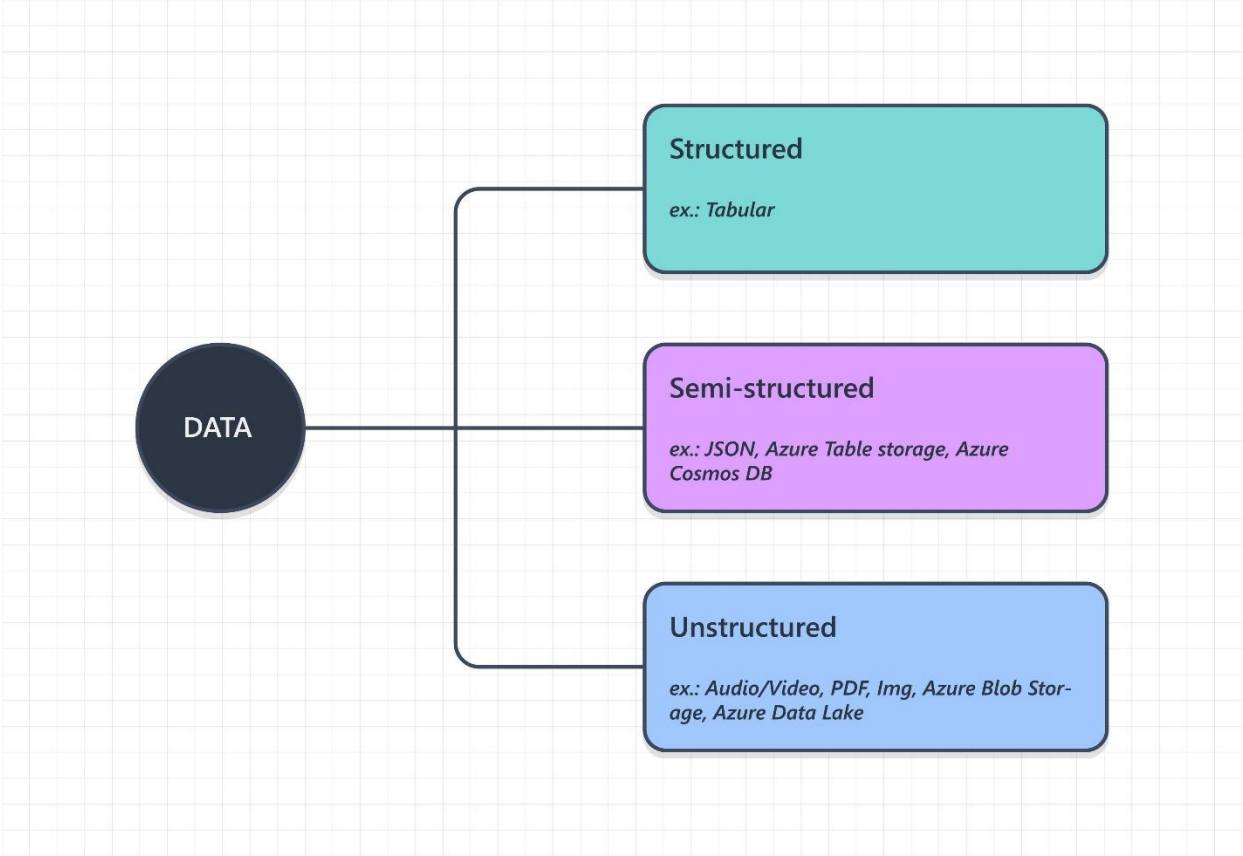
Codul ruleaza o bucla infinita care interogheaza senzorul DHT11 la fiecare 2 secunde.

- Biblioteci: Se utilizeaza DHT.h pentru gestionarea protocolului specific "single-wire" al senzorului.
- Logica principala:
    1. Initializeaza comunicatia Seriala: Serial (115200 baud) pentru monitorizare PC si Serial3 (9600 baud) pentru legatura cu ESP32.
    2. Citeste umiditatea (h) si temperatura (t).
    3. Verifica validitatea datelor folosind isnan(). Daca senzorul este deconectat, afiseaza o eroare in consola.
    4. Serializarea datelor: Datele sunt formatate intr-un sir simplu de caractere, separate prin virgula (CSV), de forma: TEMPERATURA,UMIDITATE (ex: 24.5,50.2). Acest string este trimis pe Serial3.

B. Codul pentru ESP32 (Web Server & Gateway)

ESP32 actioneaza ca un server HTTP si procesor de date seriale.

- Gestionarea Seriala: Se defineste o instanta HardwareSerial MySerial(2) pe pinii 16 (RX) si 17 (TX). Functia MySerial.readStringUntil('\n') captureaza pachetul trimis de Mega.
- Parsing: Stringul primit este "spart" la pozitia virgulei. Prima parte devine variabila temperatura, a doua umiditate.
- Web Server:
    - Ruta / (Root): Serveste interfata HTML.
    - Ruta /style.css: Serveste fisierul de stilizare CSS pentru un design responsive si modern (Carduri, Gradienturi).
    - Ruta /data: Returneaza un obiect JSON (application/json) cu valorile curente. Aceasta este ruta apelata de scriptul JavaScript din browser.
- Frontend (HTML/JS): Pagina web contine un script JavaScript care ruleaza setInterval la 2 secunde. Acesta face un fetch catre /data, preia JSON-ul si actualizeaza DOM-ul paginii instantaneu, calculand si nivelul de confort (Rece/Cald/Confortabil).



4. Testare si Functionalitate

Testarea proiectului s-a realizat in mai multe etape, validand fiecare modul individual si apoi sistemul integrat.

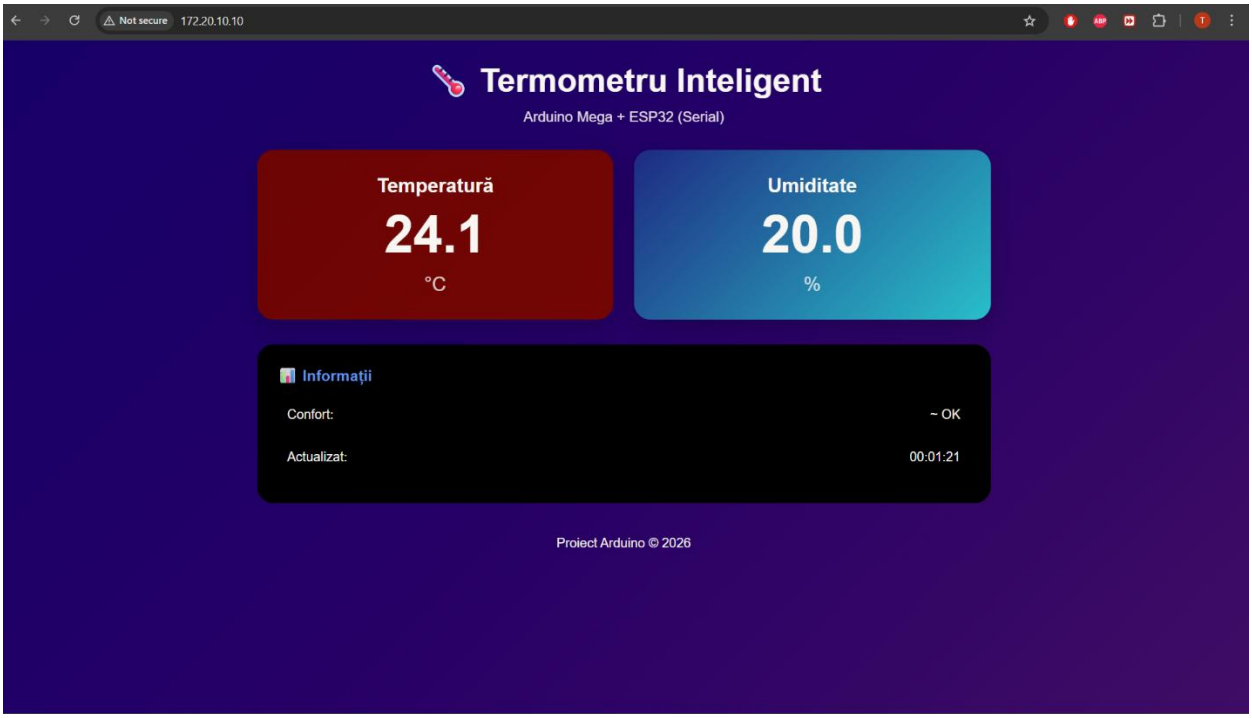
Scenariu de Testare:

1. Alimentare: La alimentarea sistemului, Arduino Mega initializeaza senzorul, iar ESP32 se conecteaza la reteaua Wi-Fi iPhoneCezar_14. In monitorul serial al PC-ului se confirma obtinerea adresei IP.
2. Achizitie Date:
    - Arduino Mega citeste temperatura ambientala (ex: 22°C).
    - LED-ul TX de pe Mega clipeste scurt, indicand transmisia catre ESP32.
3. Verificare Receptie:
    - Pe Serial Monitor-ul conectat la ESP32, apar mesajele de debug: Update Serial: T=22.0 H=45.0. Aceasta confirma integritatea conexiunii fizice UART.

4. Accesare Web:
    - Utilizatorul acceseaza IP-ul ESP32 in browser. Se incarca interfata grafica.
    - Test functional: Am suflat aer cald spre senzorul DHT11.
    - Rezultat: In mai putin de 2 secunde, valoarea temperaturii de pe ecran a crescut, iar indicatorul de confort s-a schimbat din "Confortabil" in "Cald".

Gestionarea Erorilor:

- In cazul in care senzorul DHT11 este scos din priza, Arduino Mega detecteaza eroarea (isnan) si nu trimite date eronate catre ESP32, prevenind afisarea unor valori de "0" sau "NaN" pe pagina web.
- Daca reteaua Wi-Fi cade, ESP32 va incerca reconectarea la urmatorul restart.



5. Concluzii si Directii de Dezvoltare Ulterioara

5.1. Concluzii

Proiectul „Termometru Smart Web" a reusit sa atinga toate obiectivele propuse initial, demonstrand viabilitatea unui sistem IoT (Internet of Things) modular, construit cu resurse accesibile.

Prin separarea sarcinilor intre Arduino Mega (dedicat achizitiei de date si interfatarii cu senzorii) si ESP32 (dedicat comunicatiei si interfetei Web), am obtinut un sistem stabil si performant. Aceasta arhitectura distribuita ofera avantajul ca o eventuala blocare a conexiunii Wi-Fi nu afecteaza procesul de masurare a temperaturii, asigurand continuitatea monitorizarii.

Din punct de vedere al utilizatorului, interfata Web creata este intuitiva, moderna si nu necesita instalarea unei aplicatii dedicate, fiind accesibila direct din browserul oricarui dispozitiv mobil conectat la retea. Testele efectuate au confirmat acuratetea transmisiei datelor prin protocolul Serial UART si rapiditatea actualizarii informatiilor in interfata grafica.

5.2. Propuneri de Imbunatatire (Future Work)

Desi proiectul este functional in forma curenta, arhitectura sa modulara permite numeroase extinderi hardware si software pentru a-l transforma intr-un sistem complet de automatizare a locuintei:

La nivel Hardware:

- Upgrade Senzor: Inlocuirea senzorului DHT11 cu DHT22 sau BME280 pentru o precizie mai mare si un interval mai larg de masurare a temperaturii.
- Adaugare Actuatori: Integrarea unui modul releu conectat la Arduino Mega. Astfel, sistemul ar putea nu doar sa monitorizeze, ci si sa actioneze: de exemplu, pornirea automata a unui ventilator daca temperatura depaseste 28°C.
- Carcasa 3D: Proiectarea si imprimarea 3D a unei carcase compacte care sa protejeze circuitele si sa permita o circulatie corecta a aerului pentru senzor.

La nivel Software:

- Istoric si Grafice: Implementarea unei baze de date (ex: InfluxDB sau stocare locala SPIFFS pe ESP32) pentru a salva istoricul temperaturilor si afisarea unor grafice de evolutie pe ultimele 24 de ore.
- Notificari Push: Integrarea unui serviciu de notificari (ex: Telegram Bot sau Email) pentru a alerta utilizatorul pe telefon atunci cand parametrii ies din zona de confort.
- Comunicare Bidirectionala: Modificarea codului pentru a permite trimiterea de comenzi din interfata Web inapoi catre Arduino Mega (ex: buton de ON/OFF pentru un sistem de ventilatie).