RoAlert

Introducere

Prin proiectul meu numit RoAlert, îmi propun să creez un sistem de alertă meteo inteligent folosind ESP32, care monitorizează în timp real umiditatea aerului,temperatura și detecția ploii în cazul unei furtuni. Prezentare pe scurt:

* Ce face: Măsoară condiții atmosferice (umiditate și ploaie) și trimite o alertă online dacă valorile depășesc praguri critice.

* Scopul: Anticiparea condițiilor meteo periculoase și notificarea rapidă printr-o interfață web.

* Ideea: Am pornit de la sistemele de avertizare națională (RO-ALERT) și am dorit să creez o versiune DIY pentru învățare și aplicații personale.

* Utilitate: Este un sistem educativ și practic, util pentru zone unde nu există monitorizare meteo locală sau pentru hobby-uri (ex: agricultură, tabere, drumeții etc).

Descriere generală

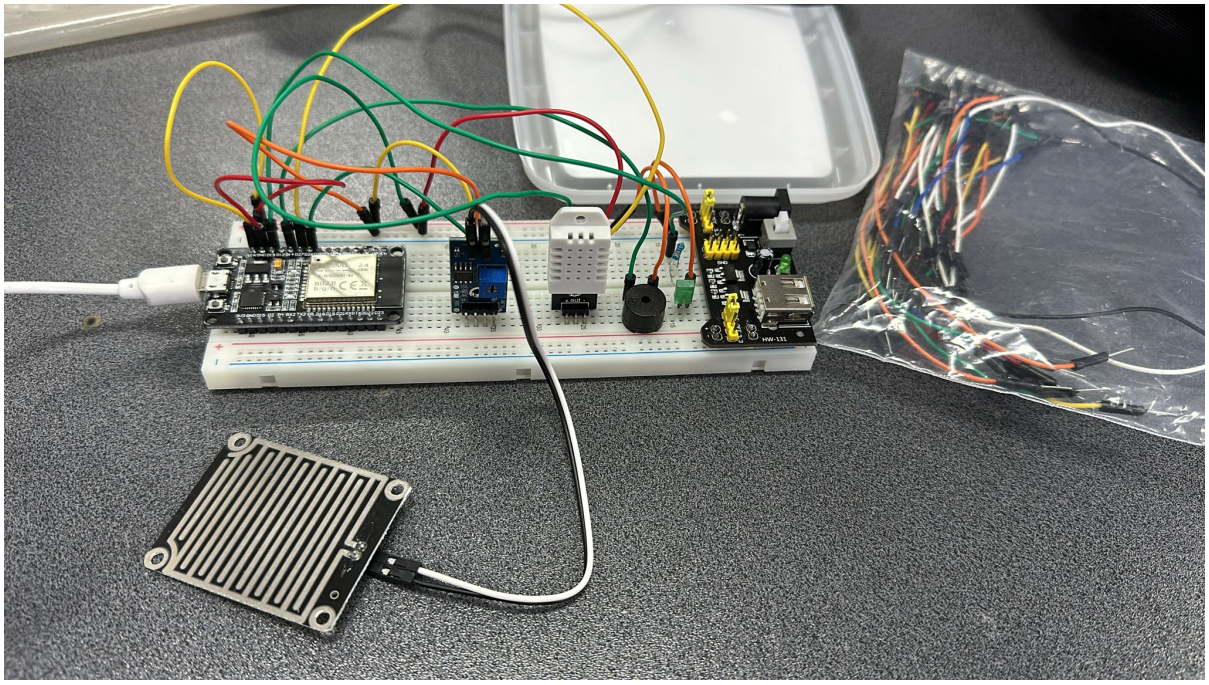
Voi folosi un senzor de umiditate și temperatură și un senzor de picături de ploaie. Acestea vor trimite date din 5 în 5 secunde către un server și se va calcula un procent pe baza căruia se va estima venirea unei furtuni. Dacă procentul depășește 50%, atunci pe site-ul web fontul se va schimba, va începe semnalul sonor și voce care va atenționa, precum și buzzerul si led-ul de pe breadboard vor emite semnale.

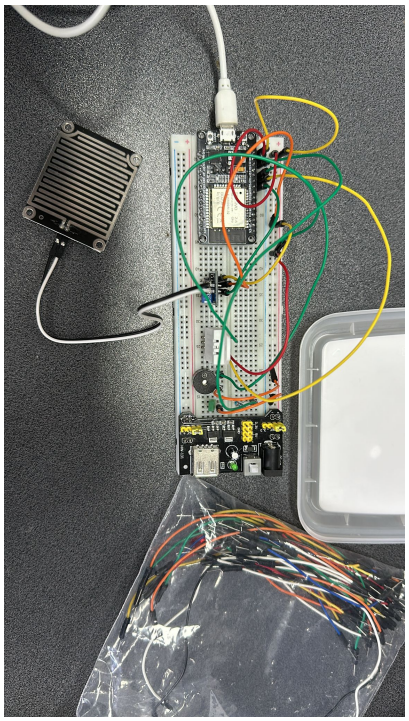
Hardware Design

- listă de piese :

1. ESP32

2. Sursă de tensiune
3. Fire jumper
4. Buzzer
5. LED
6. Modul Senzor Ploaie
7. DHT22





Software Design

Descrierea codului aplicaţiei (firmware):

- mediu de dezvoltare : VS Code (PlatformIO)
- librării şi surse 3rd-party (Flask pentru server-ul web, Arduino, Wifi, Dht, HTTPClient pentru trimiterea de mesaje POST către server)
- algoritmi şi structuri implementate : Structuri json pentru datele trimise către server (temperatură, umiditate, senzor de ploaie, procentul

de risc pentru furtuna)

- surse şi funcţii implementate : Un fisier separat in python care are codul html si serverul flask care accepta mesaje de tip post de la esp32,

apoi le afiseaza in html si activeaza diferse semnale (notificari pe laptop, semnal audio vocal pe web, buton de redare a alertei daca vrem sa se repete, schimbarea culorii fontului). In main-ul din platformio citesc datele de la senzori, aprind si starea de alerta cu led-ul si buzzer-ul pentru riscul de peste 50% si il si calculez si apoi le trimit pe toate intr-un json catre server.

Rezultate Obţinute

Un server flask complet funcțional și o detecție a unei furtuni sau a unei apropieri de furtuni cu o acuratețe ridicată.

Concluzii

-Am realizat tot ce mi-am propus (ba chiar mai mult de atât)

-Am învățat cum să lucrez singur pe placă și breadboard

-Am înțeles cum se montează corect componentele pe breadboard

Overall, a fost o experiență foarte plăcută și interesantă.

Download

Arhiva care curpinde codul de server în Flask Python precum și proiectul din platformio:

proiect_pm.zip

Jurnal

01.05 - 05.05 — Am cumpărat toate piesele necesare și am făcut research-ul corespunzător pentru acestea pentru a avea tot ce îmi trebuie.

06.05 - 10.05 — Asamblare piese și conectare pe breadboard + mici teste să văd dacă totul funcționează

11.05 - 20.05 — Scriere cod, testare, conectare + design site web + calcul formulă procentaj furtună + adăugare funcționalități în plus(sunet, notificare)

Bibliografie/Resurse

https://www.youtube.com/watch?v=UuxBfKA3U5M&ab_channel=Lucca%27sLab

https://www.youtube.com/watch?v=1O1kDmfUG68&ab_channel=SriTuHobby

https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

Export to PDF