Sistem de detecție a scurgerilor de gaze

Introducere

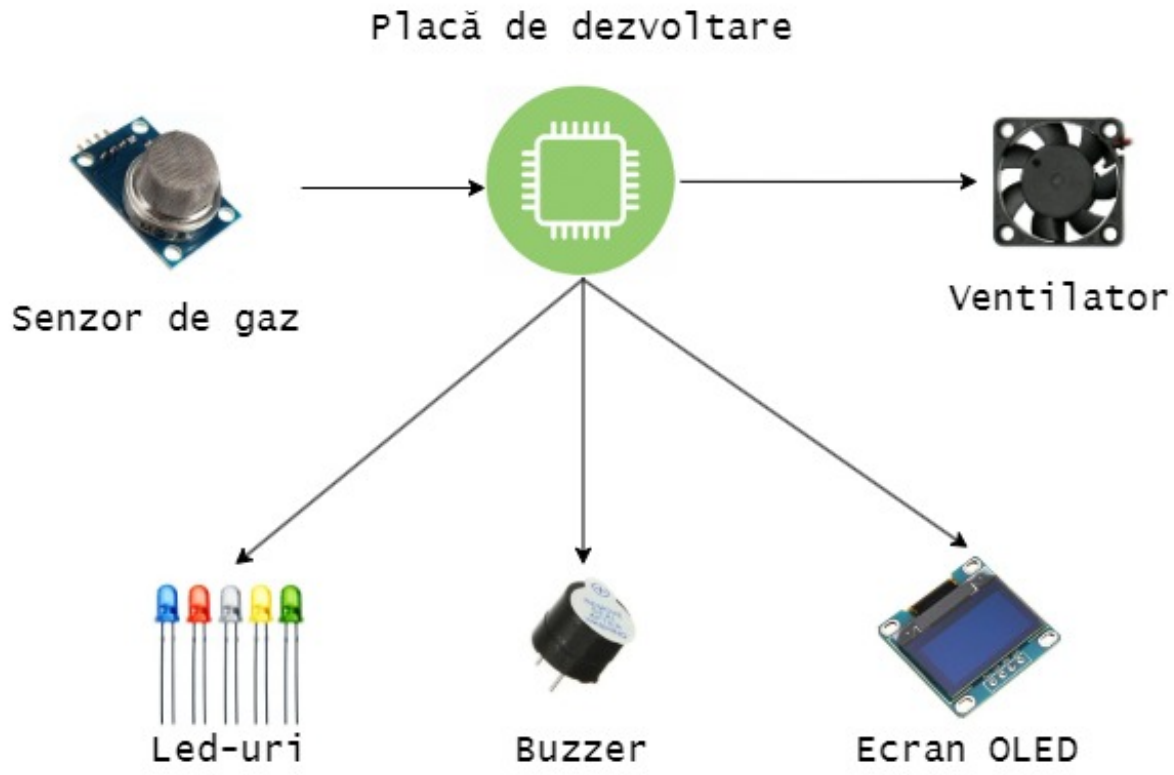
- **Nume: Cepoiu Ioana-Andreea**
- **Grupă: 331CB**
- **Îndrumător: Ionuț Oțelea**

- Proiectul propune crearea unui detector de gaze (GPL, izobutan, propan, metan, alcool, hidrogen, fum) ca metodă de precauție în cazul incendiilor sau intoxicațiilor.
- Am ales acest proiect pentru a mă familiariza cu conceptele de PM, și tot odată ca să înțeleg cum poate fi creat un device important pentru siguranța unei locuințe.

Descriere generală

Sistemul dispune de un senzor de gaze al cărui output este analizat constant de către microcontroller. Acesta este dotat cu un buzzer pentru a semnala scurgerile de gaze sau fumul. De asemenea, sistemul dispune de 4 led-uri și un ecran folosite pentru a descrie concentrația exactă a gazului în încăpere. Cu cât concentrația gazului din încăpere crește, cu atât se vor aprinde mai multe led-uri. Pentru a evacua gazul/fumul din încăpere, detectorul dispune și de un ventilator.

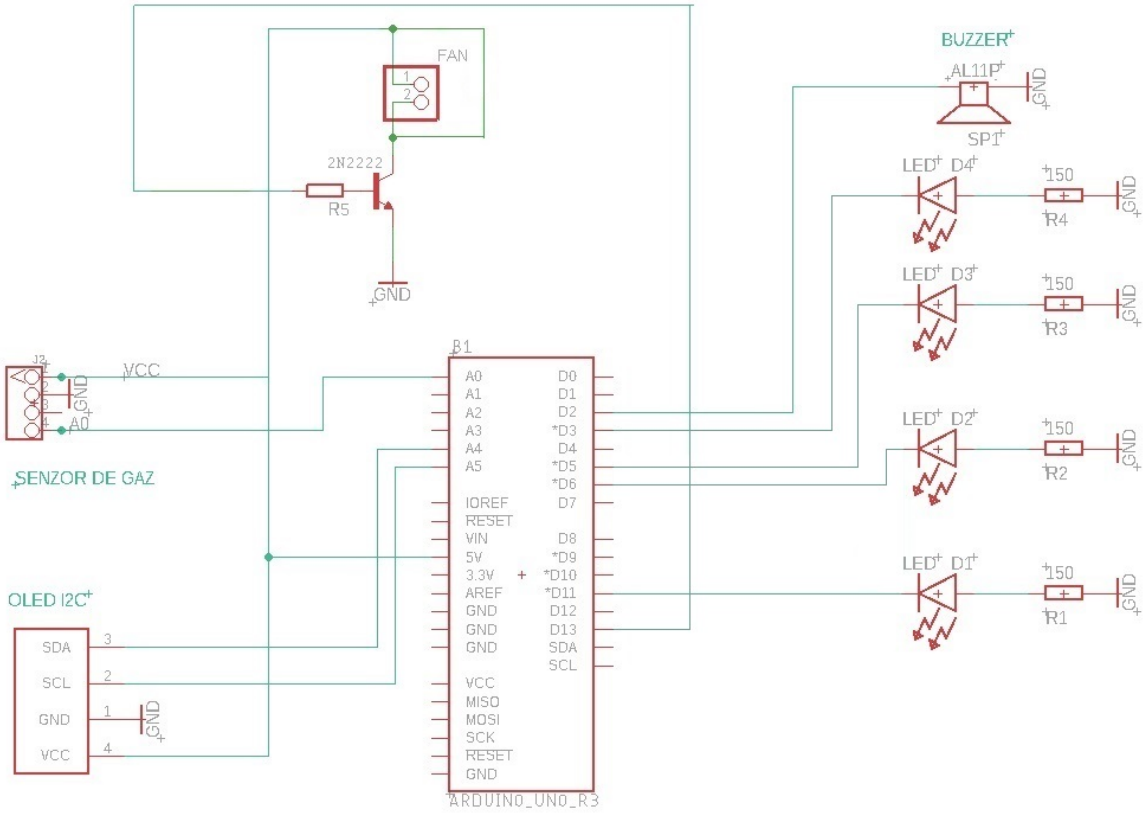
Inițial, ventilatorul va fi stins, iar led-urile vor fi aprinse doar partial in functie de concentratia citita de senzor. Cu cat senzorul detecteaza mai mult gaz/fum, cu atat mai mult se vor aprinde led-urile. Daca în încăpere procentul de gaz este mai mare decat al aerului, toate led-urile se vor aprinde, buzzerul va începe să sune, iar ventilatorul va fi activat pentru a evacua gazul sau fumul din încăpere. Chiar daca valoarea scade sub threshold, ventilatorul va continua sa functioneze pana cand procentul scade iar la valori normale.



Hardware Design

Pentru crearea proiectului folosesc:

- Arduino UNO R3
- senzor gaz MQ-2
- buzzer activ 5v
- led-uri de 20mA
- rezistențe (4 x 100 Ohm)
- ventilator 5V Raspberry Pi 3030
- tranzistor 2N2222 (astept comanda)
- display OLED 0.96” I2C IIC
- 2 breadboard-uri
- sursă de alimentare (tensiune de iesire 5V)



Conectare:

Ecran OLED - Arduino UNO R3:

- **GND - GND**
- **VCC - 5V**
- **SDA (serial data pin) - A4** (SDA pentru comunicarea prin I2C)
- **SCL (clock pin) - A5** (SCL pentru I2C)

Buzzer - Arduino UNO R3:

- **VCC - D2 (pin digital)**
- **GND - GND**

Ventilator - Arduino UNO R3:

- **VCC - D13 (pin digital)**
- **GND - GND**

Voi folosi si un tranzistor NPN pentru a ma putea asigura ca pot alimenta ventilatorul (acesta consuma 0.1-02A, iar pinul digital de Arduino poate oferi doar maxim 40mA pe pinul digital). La baza tranzistorului se foloseste o rezistenta de 470.

Senzor de gaz - Arduino UNO R3:

- **VCC - 5V**
- **GND - GND**
- **D0 - LIBER**

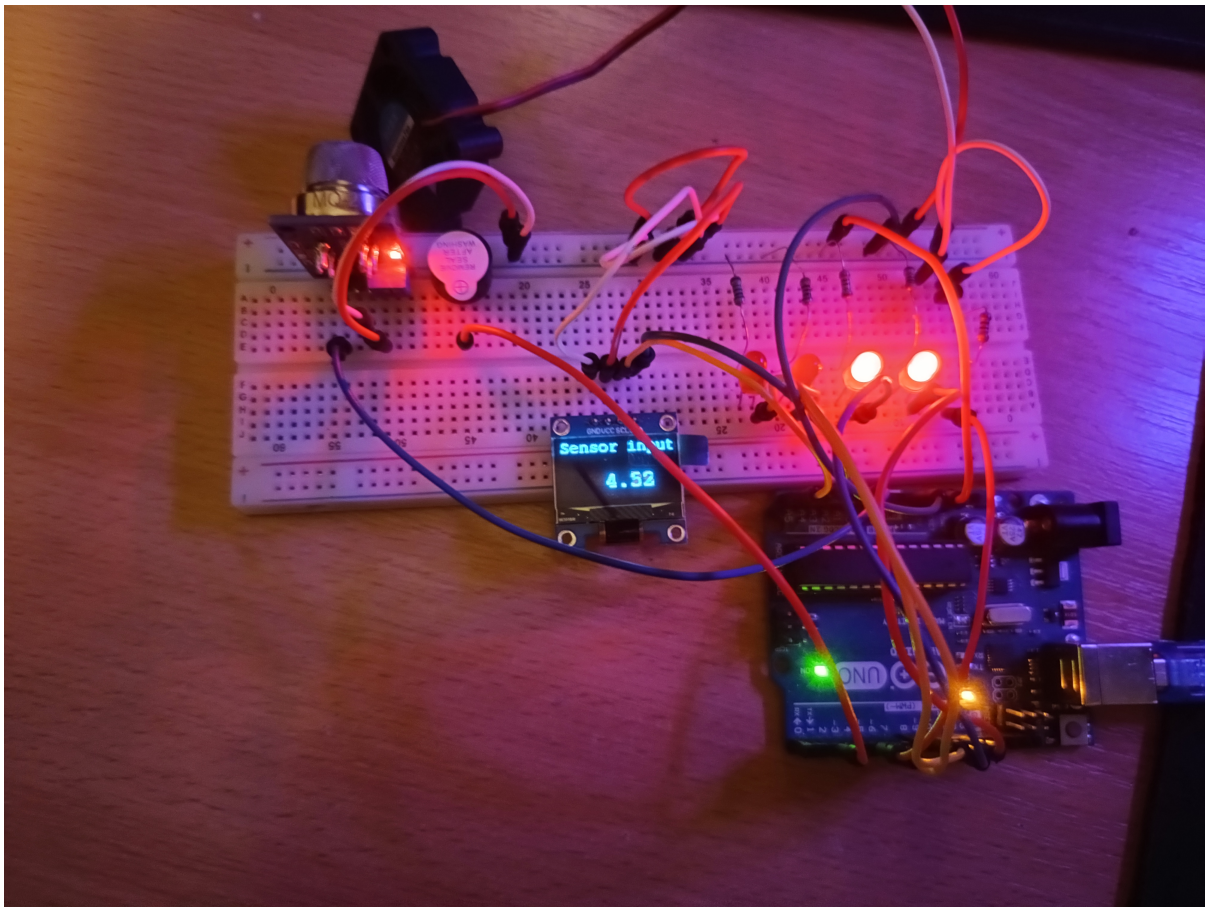
- **A0 - A0 (pin analogic)**

Am conectat pinul analog si nu si cel digital al senzorului deoarece sistemul trebuie sa masoare concentratia in ppm a gazului in incapere, si nu pot folosi pinul digital pentru masuratori mai exacte.

Led-uri - ESP32:

- **catod Led1: D11**
- **catod Led1: D6**
- **catod Led1: D5**
- **catod Led1: D3**

Am ales acesti pini deoarece au functionalitate PWN, iar sistemul trebuie sa controleze separat aprinderea fiecarui LED in functie de concentratia gazului. Pentru protejarea LED-urilor am folosit Rezistente de 150.



Se poate observa ca toate componentele sunt functionale. Pe ecranul OLED afisez concentratia (in ppm) citita de la senzorul de gaz, calibrata.

Software Design

Descrierea codului aplicaţiei (firmware):

- **Mediu de dezvoltare:** Arduino IDE

- **Librării:** MQUnifiedsensor (pentru sensorul MQ2), Adafruit GFX si SSD1306 (display), TimerOne

Functii implementate:

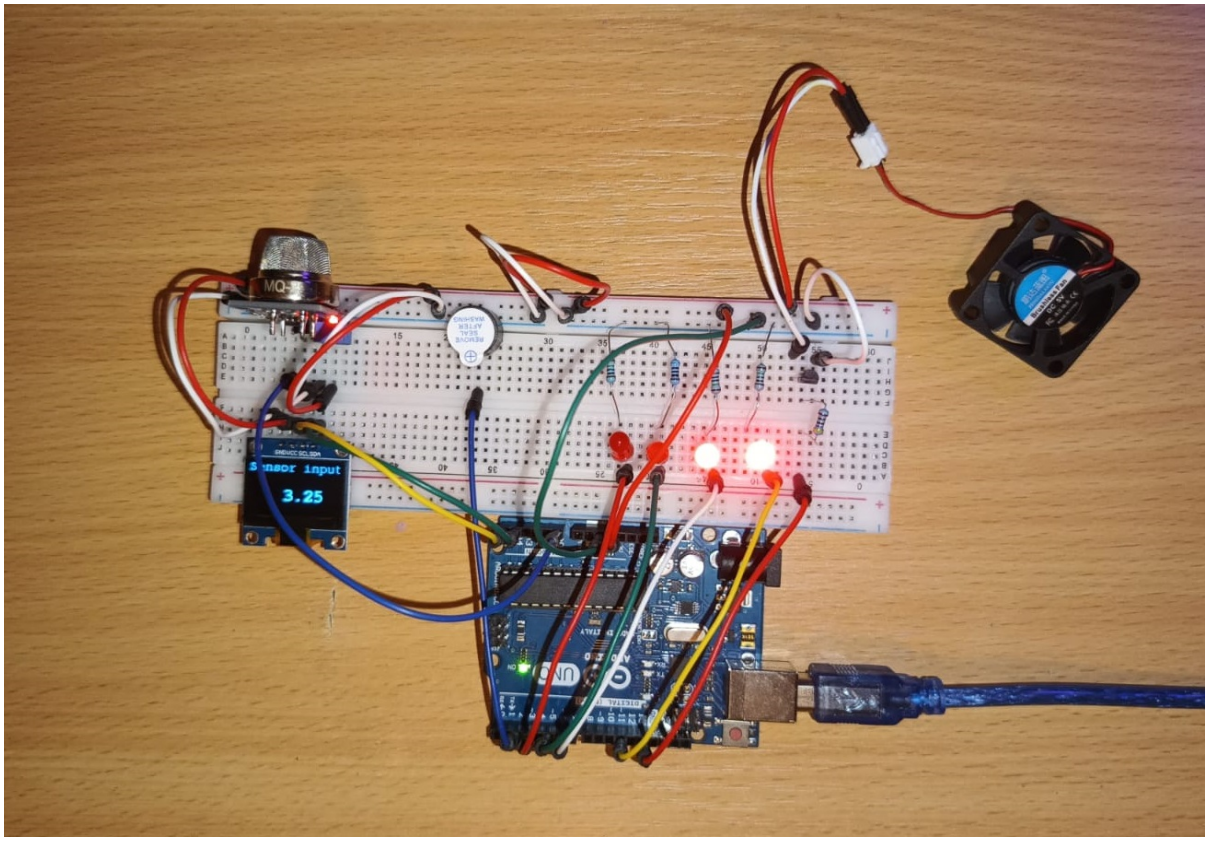
- **float mapfloat(float x, float in_min, float in_max, float out_min, float out_max)** - functie folosita pentru maparea gradelor de concentratie masurate de senzor. Astfel, gradele sunt reprezentate de intervalele (0, 2.5), (2.5, 5), (5, 7.5), (7.5, 10), iar acestea vor fi mapate pe intervalul (0, 255) pentru ajustarea luminozitatii fiecarui LED.
- **void LEDs_Brightness()** - setarea luminozitatii pentru toate cele 4 LED-uri in functie de outputul functiei mapfloat
- **void check_sensor()** - functie apelata de Timer1 la fiecare 500ms in care se citesc datele de la senzorul de gaz si verifica daca trebuie sa sune alarma. De asemenea, aici se apeleaza functia LEDs_Brightness() pentru a schimba luminozitatea.
- **void setup()** - display, pins, timer setup
- **void set_display()** - functie care plaseaza datele pe ecran
- **void loop()** - functie care aplica modificarile pe ecran la fiecare 500ms.

Librariile au fost foarte utile:

- **MQUnifiedsensor** ofera functii speciale pentru a verifica serial valorile date de sensor, precum si pentru calibrarea sensorului.
- **Adafruit SSD1306** are functii straightforward pentru modificarea afisajului pe ecran.
- **TimerOne** a fost de asemenea foarte usoara de folosit pentru a putea efectua citirile si schimbarile PWM ale led-urilor in mod regulat.

Rezultate Obţinute

Am reusit sa obtin tot ce mi-am propus. Mai jos voi atasa un video pentru a arata cum functioneaza detectorul meu



Practic, in sensorul meu detecteaza la fiecare 500ms concentratia de gaz din jurul sau. Daca inputul de la sensor creste de aprox. 10, cum se observa si pe ecran, incepe alarma si ventilatorul este aprins. Daca sursa de gaz este inlaturata, atunci ventilatorul continua sa mearga pana cand concentratia scade sub un threshold mai safe (aprox 5).

Concluzii

Proiectul meu este destul de simplist, munca mea a fost usurata foarte mult de librariile pe care le-am putut gasi pentru piesele mele. Am intampinat cateva probleme in momentul in care trebuia sa-mi aleg componentele. De exemplu, initial voiam sa folosesc un ESP32 in locul Arduino-ului, insa am constatat ca pinii digitali dau doar 3.3V, in timp ce componentele mele aveau nevoie de 5V (sensorul si ventilatorul, nu am putut gasi un ventilator care sa functioneze cu 3.3V).

De asemenea, la ventilator am facut o greseala destul de mare initial: calculasem gresit curentul necesar pentru aprinderea lui. In timp ce el avea nevoie de 0.1-0.2A, pinii digitali nu aveau direct cum sa-i ofere decat vreo 20-40 mA (desi doar 20mA sunt recomandati). Asa ca a trebuit sa-mi comand si un tranzistor pentru a ma asigura ca pot folosi ventilatorul. O idee buna ar fi fost sa folosesc si o dioda pentru a proteja circuitul de tensiunea negativa care apare atunci cand opresc ventilatorul.

Alta mica dificultate pe care am intampinat-o a fost la partea de timere. Libraria pe care am ales-o foloseste timerul 1, care afecteaza pinii digitali PWM 9 si 10, pe care eu initial ii foloseam pentru controlul LED-urilor. In momentul in care foloseam timerul, observam un efect de flickering, si a trebuit sa schimb pinii digitali folositi (din fericire mai aveam 2 pini digitali PWM disponibili).

Cat despre calibrarea senzorului, acesta are nevoie de cateva minute pana sa ofere citiri stabile (5-10 min).

Download

Arhiva proiectului meu: gas_detect.zip

Aceasta arhiva contine partea de software si schemele folosite pentru cablaj.

Resurse

Resurse hardware:

- https://www.circuito.io/blog/arduino-uno-pinout/
- https://docs.arduino.cc/built-in-examples/basics/Fade/
- https://lastminuteengineers.com/mq2-gas-senser-arduino-tutorial/
- https://www.sigmanortec.ro pentru piese

Resurse software:

- https://github.com/miguel5612/MQSensorsLib/tree/master MQ2 library
- https://github.com/adafruit/Adafruit_SSD1306
- https://github.com/adafruit/Adafruit-GFX-Library
- https://github.com/adafruit/Adafruit_BusIO
- https://forum.arduino.cc/t/use-of-timerone/438629/8
- https://www.youtube.com/watch?v=SXZkX3cJqDs (tutorial timere)

Jurnal

- 04.05.2024 - creere pagină de prezentare a proiectului, adăugare descriere, schemă bloc, listă de piese, schemă în Thinkercad
- 16.05.2024 - alegere pini de conectare, asamblare hardware
- 22.05.2024 - finish software, just finishing touches now

Export to PDF