**MONITORIZAREA MONOXIDULUI DE CARBON CU AJUTORUL APLICAȚIEI COMONITORING**

**Candidat: Paul-Andrei Bora**

**Coordonator științific: Ș.l. dr. ing. Daniel Iercan**

Sesiunea: Iunie 2023

**CUPRINS**

[1 INTRODUCERE 3](#_Toc137745839)

[2 MONOXIDUL DE CARBON 4](#_Toc137745840)

[2.1 INFORMAȚII GENERALE DESPRE MONOXIDUL DE CARBON 4](#_Toc137745841)

[2.2 STUDII DESPRE INTOXICAREA CU MONOXIDUL DE CARBON 6](#_Toc137745842)

[2.3 METODE DE PREVENIRE A INTOXICAȚIEI CU MONOXID DE CARBON 9](#_Toc137745843)

[3 PROGRAME FOLOSITE ÎN IMPLEMENTAREA APLICAȚIEI 11](#_Toc137745844)

[3.1 ANDROID STUDIO 11](#_Toc137745845)

[3.2 ARDUINO IDE 13](#_Toc137745846)

[3.3 THINGSPEAK 14](#_Toc137745847)

[4 DIAGRAME ȘI SCHEME 16](#_Toc137745848)

[4.1 DIAGRAMA APLICAȚIEI 16](#_Toc137745849)

[4.2 SCHEMA MONTAJULUI ARDUINO 17](#_Toc137745850)

[5 COMPONENTE HARDWARE FOLOSITE 18](#_Toc137745851)

[5.1 SENZOR DE MONOXID DE CARBON MQ-7 18](#_Toc137745852)

[5.2 MODULUL ESP8266EX 19](#_Toc137745853)

[6 PROTOCOALE FOLOSITE DE ESP8266 23](#_Toc137745854)

[6.1 TRANSMISSION CONTROL PROTOCOL (TCP) 23](#_Toc137745855)

[6.2 USER DATAGRAM PROTOCOL (UDP) 23](#_Toc137745856)

[6.3 INTERNET PROTOCOL SUITE (TCP/IP) 23](#_Toc137745857)

[7 LEGĂTURA DINTRE JAVA ȘI ANDROID STUDIO 25](#_Toc137745858)

[8 REST API’S 27](#_Toc137745859)

[8.1 API ȘI REST API 27](#_Toc137745860)

[8.2 CUM FUNCȚIONEAZĂ REST API-URILE 27](#_Toc137745861)

[8.3 REST API-URI DEZVOLTATE DE THINGSPEAK 28](#_Toc137745862)

[9 INTERFAȚA CU UTILIZATORUL 29](#_Toc137745863)

[10 IMPLEMENTARE 34](#_Toc137745864)

[10.1 IMPLEMENTARE HARDWARE 34](#_Toc137745865)

[10.2 CONFIGURAREA CANALULUI DIN THINGSPEAK 36](#_Toc137745866)

[10.3 IMPLEMENTAREA SOFTWARE 39](#_Toc137745867)

[BIBLIOGRAFIE 57](#_Toc137745868)

# INTRODUCERE

Monoxidul de carbon este un gaz inodor, incolor și insipid care este degajat în aerul respirabil dintr-o multitudine de cauze, în special ardearea incompletă a combustibililor folosiți zilnic chiar și în locuința proprie. Din păcate, acest gaz este foarte toxic pentru om deoarece afectează direct sistemul cardiovascular ce duce la multe alte probleme de sănătate. Chiar și expunerea de scurtă durată, la o cantitate minoră de monoxid de carbon, poate cauza efecte adverse neplăcute și poate la consecințe pe termen lung. Pe piață există o diversitate de instrumente de măsurare a monoxidului de carbon care pot avertiza persoanele în cazul unei scurgeri, însă majoritatea nu dispun de o aplicație mobilă de monitorizare în timp real a concentrației de monoxid de carbon din aer. De aceea am decis să implementez o aplicație ce arată utilizatorului în permanență calitatea aerului și trimite email-uri de alarmă in cazul unor concentrații prea mari.

Aplicația permite setarea pragului pentru trimiterea de email-uri și afișează cantitatea de monoxid de carbon din aer. Reține un mic istoric cu email-urile care au fost trimise și totodată permite configurarea contactului unde se vor trimite notificările.

# MONOXIDUL DE CARBON

## INFORMAȚII GENERALE DESPRE MONOXIDUL DE CARBON

Monoxidul de carbon (CO) este un gaz comun, format din carbon și oxigen. Compoziția chimică a unei molecule de monoxid de carbon este formată dintr-un atom de carbon (C) și un atom de oxigen (O). Cei doi atomi sunt legați între ei printr-o legătură covalentă triplă. Oxigenul are o încărcare negativă datorită electronegativității sale crescute care atrage electronii mai mult decât carbonul (1).

(1)

Este cel mai simplu oxid de carbon.

Monoxidul de carbon este inodor, incolor și insipid și poate proveni din mai multe surse, fie naturale, fie induse și create de om. Cauzele naturale includ și incendiile de pădure, oxidarea hidrocarburilor nemetanice și oxidarea metanelor, erupții vulcanice și descărcări electrice. Dintre sursele antropice se pot enumera și arderea incompletă a combustibilor ce conțin atomi de carbon care sunt întâlniți și folosiți pe întregul glob, unii mai comuni decât alții, cum ar fi: cărbunele, benzina, kerosenul, gazele naturale, păcura, propanul și lemnul. Acțiunile produse de om ce pot rezulta în degajări de monoxid de carbon sunt cele mai comune în gospodăriile personale, la gătitul fără aerisire corespunzătoare (cuptoarele și aragazele emană monoxid de carbon în aerul respirabil din locuință), consumul de tutun în spațiile închise sau slab aerisite, dar și în agricultură, în curățarea pământului prin metoda tăierii și arderii.

O altă sursă, și probabil cea mai comună dintre toate, este conducerea unui vehicol. În traficul din orașele mari și foarte populate, concentrația de monoxid de carbon din aer este mult mai mare decât în orașele mici, comune sau sate.

Acest gaz este adesea folosit în industria chimică drept un intermediar chimic în diferite procese (ex., producția de metanol, acid acetic, anhidridă acetică) sau ca agent reducător în obținerea fontei. În afara laboratoarelor specializate, gazul este des întâlnit în aerul respirabil din cauza factorilor creatori enumerați anterior.

Monoxidul de carbon a câștigat titlul de „criminalul silențios” pentru că nu poate fi detectat de om fără să aibă în posesie un dispozitiv special de detectare a monoxidului de carbon din aer. Astfel că degajarea lui este simțită mult prea târziu, când saturația de monoxid de carbon crește în sânge și apar efectele adverse, lucru ce poate escalada repede în doar câteva minute.

Din moment ce nu poate fi detectat și este un gaz foarte toxic pune sănătatea în pericol. Expunerea și inhalarea gazului poate duce la scăderea oxigenului din organele vitale, în special din inimă și din creier. Odată pătruns în plămâni, moleculele de monoxid de carbon ajung în fluxul sangvin și înlocuiesc moleculele de oxigen transportate în hemoglobina din celulele roșii. Aproximativ 95% din monoxidul de carbon absorbit se combină cu hemoglobina, formând astfel carboxihemoglobina. Procentajul este crescut pentru că afinitatea hemoglobinei pentru monoxidul de carbon este de 200 de ori mai mare decât cea pentru oxigen (14 mmHg).

Efectele expunerii la monoxidul de carbon pot varia în funcție de timpul expunerii, de concentrație și de cantitatea asimilată. Efectele la expunerea de scurtă durată și inhalarea unei cantități minore pot fi neurocomportamentale (ex., de vorbire, de atenție, de procesare a diferitelor emoții, de dificultate la învățare, de confuzie), cardiovasculare, anemie, dureri la nivelul capului sau stomacului, stări de vomă și de amețeală, în vreme ce expunerea de lungă durată la o concentrație mare de monoxid de carbon poate duce la inconștiență, comă, convulsii, stop cardiorespirator și chiar deces. Efectele fatale sunt, de regulă, la 100 părți per milion (ppm), echivalentul a 99,886 de miligrame per litru sau a 114,561 miligrame per metru cub, când carboxihemoglobina ajunge la un procent de 50%(Tabelul 2.1.1).

Tabelul 2.1.1 – Simptomele intoxicației cu monoxid de carbon raportat

la nivelul de carboxihemoglobină din sânge

|  |  |
| --- | --- |
| Concentrația de carboxihemoglobină din sânge | Posibile simptome ale intoxicației cu monoxid de carbon |
| 5% | Niciuna |
| 10% | Niciuna |
| 15% | Dureri ușoare de cap |
| 20% | Migrene și dureri de stomac |
| 25% | Dureri de cap și amețeală |
| 30% | Stare de somnolență și confuzie |
| 35% - 40% | Stare de vomă, leșin, probleme de atenție |
| 40% - 50% | Confuzie, pierderea cunoștiinței, comă, posibil moarte |
| 50% - 70% | Convulsii, comă, stop cardiorespirator, moarte |
| > 70% | Moarte |

Concentrațiile de monoxid de carbon afectează corpul uman în mod diferit, iar în tabelul amintit anterior (Tabelul 2.1.1) sunt prezentate posibilele simptome generale, dar aceeași cantitate poate declanșa efecte adverse minore, cum ar fi starea de vomă sau dureri abdominale, la un pacient expus, dar și convulsii sau pierderea cunoștiinței la alt pacient expus. Cei mai afectați și predispuși la complicații sunt, de obicei, bătrânii, femeile însărcinate și feții, bebelușii, persoanele care suferă de boli cardiovasculare, respiratorii și cei care suferă de anemie. De asemenea, nefumătorii care stau în prezența altor persoane care fumează tutun sunt mult mai afectați pentru că fumătorii sunt cei care emit monoxidul de carbon prin expirare, iar la inhalare consumă oxigenul.

Efectele adverse ale monoxidului de carbon asupra corpului uman sunt similare cu simptomele unei răceli normale sau ale gripei (fără să fie menționată febra crescută), cu intoxicații alimentare sau boli comune. Astfel, ajung ignorate sau tratate necorespunzător, de aceea este benefică cunoașterea diferențelor și prevenirea și tratarea pacienților intoxicați într-un timp eficient.

## STUDII DESPRE INTOXICAREA CU MONOXIDUL DE CARBON

**A**. Un studiu creat și publicat într-un articol din revista „Lancet Planet Health” în anul 2021 a analizat efectele monoxidului de carbon asupra populației [2], sub numele de „Ambient carbon monoxide and daily mortality: a global time-series study in 337 cities” (traducere: Monoxidul de carbon și mortalitatea zilnică: un studiu global pe serii în 337 de orașe).

O echipă, formată din 38 de cercetători și profesori din orașele incluse în proiect, a colectat informații zilnice despre poluarea aerului, meteorologia și mortalitatea din 337 de orașe din 18 țări diferite, răspândite pe întregul glob, acoperind o perioadă de 37 de ani (din 1979 și până în 2016). Acest studiu internațional a fost cel mai mare studiu epidemiologic bazat pe mortalitatea zilnică rezultată din intoxicarea cu monoxid de carbon.

În urma informațiilor colectate (Tabelul 2.2.1), au ajuns la concluzia că la creșterea cu 0.001 părți per milion de monoxid de carbon din aerul respirabil, mortalitatea crește și ea cu 0.91%. Cu un total de 40 090407 de morți în orașele participante din țările menționate în Tabelul 2.2.1, specialiștii au putut estima corelația dintre decese și concentrația de monoxid de carbon. Media generală a monoxidului de carbon din aer a fost estimată la 2,3 miligrame per metru cub.

Unele orașe din Statele Unite ale Americii au prezentat cel mai mare nivel de monoxid de carbon în aer, în vreme ce orașe din Canada, România, Spania și Marea Britanie au prezentat cele mai mici nivele de monoxid de carbon.

Din orașele menționate, 28 au prezentat 7 miligrame per metru cub de monoxid de carbon, 24 de orașe fiind din Statele Unite ale Americii.

Tabelul 2.2.1 – Datele colectate despre mortalitate, aerul poluat și temperatură (informații preluate din „Table 1: Summary statistics of daily data on deaths, air pollutants, and temperature”, pagina e194, Vol 5 Aprilie 2021, revista Lancet Planet Health)

| **Țară** | **Nr. orașe** | **Perioadă** | **Morți** | **Monoxidul de carbon (în mg/ m3)** | | | **Temperatură medie (în grade Celsius)** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Prima percentilă** | **Medie** | **A 99-a percentilă** |
| Australia | 2 | 2000–09 | 279 410 | 0,00 | 0,38 (0,25–0,63) | 1,88 | 18,1 (14,5–22,2) |
| Canada | 24 | 2000–15 | 1 759 125 | 0,00 | 0,32 (0,23–0,48) | 1,39 | 8,1 (0,0–15,8) |
| Chile | 3 | 2004–14 | 282 232 | 0,01 | 0,57 (0,34–1,06) | 3,49 | 13,4 (10,3–17,1) |
| China | 3 | 2013–15 | 246 246 | 0,35 | 0,95 (0,73–1,24) | 4,16 | 19,9 (11,3–25,6) |
| Finland | 1 | 1994–2014 | 153 308 | 0,11 | 0,25 (0,19–0,35) | 0,95 | 5,9 (0,0–13,8) |
| Germany | 12 | 1993–2015 | 1 897 704 | 0,05 | 0,40 (0,27–0,63) | 1,79 | 10,4 (4,7–15,8) |
| Italy | 12 | 2013–15 | 212 494 | 0,14 | 0,57 (0,33–0,99) | 2,23 | 16,1 (10,6–21,7) |
| Japan | 6 | 1979–09 | 3 112 302 | 0,18 | 0,68 (0,48–0,99) | 2,63 | 15,6 (8,1–22,3) |
| Portugal | 2 | 1995–2012 | 559 147 | 0,02 | 0,27 (0,19–0,40) | 1,09 | 16,3 (12,8–19,9) |
| Romania | 7 | 2008–16 | 242 014 | 0,01 | 0,17 (0,08–0,39) | 2,72 | 12,3 (4,9–19,8) |
| South Korea | 7 | 1999–2015 | 1 661 559 | 0,27 | 0,61 (0,48–0,82) | 1,85 | 15,1 (5,9–22,0) |
| Spain | 45 | 2003–14 | 1 306 694 | 0,04 | 0,36 (0,25–0,51) | 1,25 | 15,7 (10,4–21,2) |
| Sweden | 1 | 1990–2010 | 194 239 | 0,31 | 0,92 (0,65–1,45) | 3,48 | 6,7 (1,1–13,6) |
| Switzerland | 4 | 1995–2013 | 150 003 | 0,20 | 0,50 (0,40–0,90) | 2,60 | 11,0 (4,9–16,8) |
| Taiwan | 3 | 1996–2014 | 1 113 099 | 0,26 | 0·70 (0·53–0·93) | 1,91 | 25,1 (20,4–28,0) |
| Thailand | 19 | 1999–2008 | 758 133 | 0,13 | 0,63 (0,42–0,92) | 1,99 | 28,0 (26,5–29,2) |
| UK | 30 | 1990–2016 | 3 543 977 | 0,07 | 0,33 (0,21–0,53) | 1,71 | 10,6 (6,7–14,8) |
| USA | 156 | 1985–2006 | 22 618 721 | 0,06 | 0,87 (0,56–1,32) | 3,93 | 15,2 (7,1–22,2) |

Calitatea standard a aerului consumabil din prezent, pe o durată maximă de 8 ore, este de 10 miligrame per metru cub și pe o durată de 24 de ore de aproximativ 7 miligrame per metru cub, cantități stabilite drept standard chiar din anul 2010 de către Organizația Mondială a Sănătății (OMS sau WHO – World Health Organization).

Studiile parametrilor au dus la o legătură directă dintre expunerea pe termen scurt la monoxidul de carbon în cantități moderate sau mici și markerii sangvini ai inflamațiilor și coagulării sângelui, incluzând și creșterea nivelului de proteine reactive de tip C, creșteerea fibrinogenului și a timpului protombinei.

Expunerea pe termen scurt la monoxidul de carbon poate duce la creșterea hipertensiunii arteriale la femei și afectarea funcțiilor pulmonare la copiii cu astm, din cauza reducerii capacității vitale de respirație.

Studiul lor se finalizează cu concluzia că expunerea pe termen scurt la monoxidul de carbon, chiar și într-o concentrație mai mică decât cea prevăzută de Organizația Mondială a Sănătății, rămâne un pericol pentru sănătatea oamenilor.

**B**. Obiectivul altui studiu realizat de Colegiul American al Medicilor de Urgență și publicat de Elsevier Inc [3] în 2022, este de a evalua riscul de epilepsie la pacienții care au fost intoxicați cu monoxid de carbon în trecut pentru că s-a constatat că porțiunile afectate de monoxidul de carbon adesea coincid cu cele afectate de epilepsie.

Au analizat 8,264 pacienți cu intoxicare de monoxid de carbon în trecut și 41,320 fără, corelându-i în funcție de vârstă, sex și anul internării, cu o rație de 1 la 5, între anii 2000 și 2010. Acești pacienți au fost urmăriți până ce au dezvoltat epilepsie, au murit sau până în data de 31 decembrie 2013 (finalul perioadei de studiu).

Folosindu-se de baza de date de cercetare a asigurării naționale de sănătate din Taiwan, au calculat indicele de risc și intervalul de încredere de 95%, în funcție de vârstă și sex (Tabelul 2.2.2) pentru a descoperi riscurile de epilepsie ulterioară.

Tabelul 2.2.2 – Asocierea epilepsiei ulterioare la pacienții cu intoxicare de monoxid de carbon

|  |  |  |
| --- | --- | --- |
| **Pacienți intoxicați cu monoxid de carbon** | **Indice de risc** | **Intervalul de încredere 95%** |
| Între 20 și 39 de ani | 11,06 | 7,17 – 17,08 |
| Între 40 și 69 de ani | 8,40 | 6,48 – 10,88 |
| Femei | 9,53 | 5,95 – 15,26 |
| Bărbați | 8,00 | 5,86 – 10,92 |

Concluzia studiului este că pacienții care au trecut prin cazuri de intoxicare cu monoxid de carbon au un risc mult mai ridicat de a dezvolta epilepsie decât pacienții care nu au fost intoxicați cu monoxid de carbon. Studiul mai arată și că persoanele mai tinere, curpinse între 20 și 39 de ani sunt mai expuse și predispuse la epilepsie, față de cei mai înaintați în vârstă.

## METODE DE PREVENIRE A INTOXICAȚIEI CU MONOXID DE CARBON

Expunerea la monoxidul de carbon este și va rămâne o problemă reală care pune în pericol starea de sănătate. Din cauza proprietăților ce fac gazul să fie „invizibil” pentru oameni, metodele de protejare și prevenire împotriva intoxicației cu monoxidul de carbon trebuie să fie cunoscute de toată lumea.

Posibile surse de monoxid de carbon prezente în locuința personală:

* Cuptoare
* Aragazuri pe gaz
* Boilere
* Cazane de apă
* Șemineuri
* Grătare interioare
* Generatoare portabile
* Lumânări parfumate
* Bețișoare parfumate
* Mașini
* Alte vehicule
* Consumul de tutun
* Încălzitoare de parafină
* Vecini (prin ventilație - pentru apartamente)

Pe lângă posibilele surse de monoxid de carbon din locuințe, mai există și altele:

* Bărci: motoarele bărcilor produc monoxid de carbon, iar bărcile care au cabine pot stoca gazul în interior
* Încălzitoarele folosite în corturi speciale (ex., pentru pescuitul pe gheață, pentru camping)
* Utilajele și echipamentele agricole ce funcționează pe bază de gaz sau alți combustibili

Unele dintre potențialele surse de monoxid de carbon sunt folosite zilnic în gospodărie sau în apropierea sa. Casa trebuie să fie un loc sigur pentru familie, în special dacă sub acoperișul ei locuiesc bebeluși, femei însărcinate, persoane în vârstă sau cu probleme de sănătate.

Din acest motiv, măsurile de prevenire a intoxicației cu monoxid de carbon pentru cunoscute și respectate de fiecare.

Iată o listă cu cele mai importante măsuri de prevenire:

* Aerisirea constantă a încăperilor din casă, a corturilor și a bărcilor cu cabine închise
* Instalarea unei hote deasupra aragazului și cuptorului și verificarea periodică a acesteia pentru a se asigura buna funcționare și ventilare
* Verificarea periodică a electrocasnicelor care funcționează pe bază de gaz sau alți combustibili
* Verificarea flăcărilor de la aragaze, cuptoare, șeminee (flacăra trebuie să fie albastră, dacă este portocalie înseamnă că monoxidul de carbon este degajat în aer)
* Deschiderea ferestrelor în cazul aprinderii lumânărilor parfumate (în special cele făcute din parafină), a bețișoarelor parfumate sau a arderii altor produse (lemn, plante) menite să împrospăteze aerul dintr-o cameră
* Verificarea de către un specialist a caloriferelor, sobelor sau a altor metode de încălzire a unei case
* Folosirea generatoarelor de căldură sau electricitate în afara spațiilor închise
* În lunile de iarnă, mașina nu trebuie pornită în garajele închise ermetic sau cele anexate la casă
* Instalarea unei alarme de monoxid de carbon

# PROGRAME FOLOSITE ÎN IMPLEMENTAREA APLICAȚIEI

## ANDROID STUDIO

În zilele noastre cea mai rapidă cale către infomație este internetul, iar cel mai comun și la îndemână dispozitiv cu acces la internet este telefonul. Prin acesta sunt oamenii sunt cel mai rapid notificați despre diferite articole, postari, sau chiar știri și date care le-ar putea salva viața la un moment dat. Din cauza acestor factori am ales ca aplicația folosită pentru a monitoriza cantitatea de monoxid de carbon dintr-o încăpere să fie o aplicație de tip mobile. Prin urmare unul dintre programele folosite la crearea aplicației este Android Studio.

Android Studio (Figura 3.1.1) este un IDE (Integrated Developement Kit) special folosit pentru aplicațiile mobile care rulează cu un sisteme de operare de tip Android. Acesta suportă diferite limbaje de programare precum Java sau C++.

A screenshot of a computer

Description automatically generated

Figura 3.1.1 – Interfață Android Studio

Android Studio folosește ca și sistem de build tipul Gradle, în care se pot adăuga diferite dependințe în funcție de nevoile programatorului.

Gradle este o unealtă automată pentru crearea aplicațiilor, acesta câștigându-și popularitatea prin susținerea aplicațiilor care sunt scrise în limbaje de programare precum Java sau C++. Totodată, Gradle permite construirea și testarea produselor software.

Dependințele (Figura 3.1.2) pe care le-am adăugat pentru a afișa diferite grafice sau pentru a trimite email-uri. Mai multe detalii se află în capitolul 8.1.1 Dependințe.

A screenshot of a computer

Description automatically generated

Figura 3.1.2 – Dependințele din fișierul „build.gradle”

O altă caracteristică pe care o Android Studio o prezintă este cea de emulare. Android Studio ne pune la dispoziție un întreg emulator (Figura 3.1.3) pentru a emula diferite dispozitive care au caracteristici specifice nevoilor.

A screenshot of a computer

Description automatically generated with medium confidence

Figura 3.1.3 – Emulator

Caracteristici precum tipul dispozitivului, rezoluția acestuia, dimensiunea ecranului, densitatea pixelilor sau imagini de sistem (Figura 3.1.4). Imaginile de sistem reprezintă o implementare Android, care este open source și care poate fi rulată pe mai multe sisteme de tip Android. Imaginea ajută la testarea și rezolvarea problemelor de compatibilitate și reprezintă un mediu în care se pot rula aplicațiile care sunt create în Android Studio.

A screenshot of a computer

Description automatically generated

Figura 3.1.4 – Setarea caracteristicilor

Android Studio este cel mai potrivit IDE pentru cerințele date și pentru diversitatea pe care o are în dezvoltarea aplicațiilor.

## ARDUINO IDE

Arduino IDE (Figura 3.2.1) este un editor de text pentru Arduino. În Arduino IDE se pot scrie schițele care sunt verificate și compilate. Dacă codul nu prezintă erori, schița este încărcată pe microcontrollerul conectat.

A screenshot of a computer

Description automatically generated with medium confidence

Figura 3.2.1 – Interfața Arduino IDE

Arduino IDE dispune și de un monitor serial în care se afișează toate datele transmise pe port la un anumit „baud rate”, ce reprezintă numărul de schimbări pe care le are un semnal într-o secundă.

Arduino IDE folosește ca limbaj de programare o variantă de C++, care are câteva metode și funcții adiționale. Datorită faptului că am ales o placă de dezvoltare Arduino UNO, Arduino IDE, este indispensabil pentru implementarea funcționalităților necesare.

## THINGSPEAK

ThingSpeak este un program software, open source, care permite conectarea dispozitivelor care au acces la internet. Acesta permite vizualizarea și analizarea datelor live primite de la diferiti senzori. ThingSpeak poate primi date de la orice dispozitiv conectat la internet prin folosirea unor „REST API”-uri. (detaliate la capitolul 8 REST API)

Un alt instrument foarte folositor, care este inclus în ThingSpeak, este MATLAB. În ThingSpeak se poate scrie, genera sau rula cod MATLAB, care ajută la vizualizarea și analiza datelor.

În Figura 3.3.1 sunt afișate datele care au fost înregistrate de câtre senzorul de monoxid de carbon. Acestea sunt transmise în timp real de la senzor și sunt salvate și expuse pe grafic.

A picture containing text, screenshot, line, plot

Description automatically generated

Figura 3.3.1 – Afișaj grafic în ThingSpeak

# DIAGRAME ȘI SCHEME

## DIAGRAMA APLICAȚIEI

Pentru a avea o vizualizare mai ușoară și pentru a vedea cum sunt transmise datele de la senzorul de monoxid de carbon către aplicația creată în Android Studio, am creat o diagramă (Figura 4.1.1) în care este expus modul și direcția în care sunt transmise datele. Funcționalitatea aplicației se bazează pe transmiterea valorilor înregistrate de către senzorul de monoxid de carbon, către aplicația mobilă.

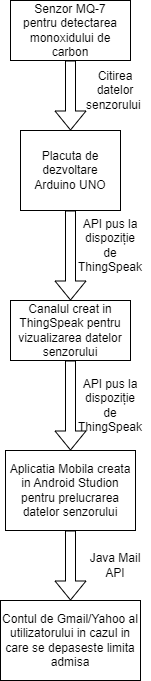


Figura 4.1.1 – Diagramă aplicație

Senzorul MQ-7 măsoară cantiteatea de monoxid de carbon din aerul încăperii, apoi acesta o transmite pe portul analogic al plăcuței, urmând ca plăcuța se conectează la internet printr-un modul ESPX8266 pentru a putea face un request către API-ul pus la dispoziție de către ThingSpeak. În ThingSpeak s-a creat un canal în prealabil și s-a generat o cheie care este necesară pentru folosirea API-ului. După ce se configurează modul ESPX8266 se trasmite valoarea senzorului către canalul de pe ThingSpeak unde se adaugă în grafic. Mai departe aplicația mobilă folosește alt API pentru a prelua datele de pe canalul din ThingSpeak și pentru a le prelucra. Aplicația verifică dacă valoarea măsurată de către senzor este mai mare decât o limită setată la lansarea aplicației sau o limită setată de către utilizator direct din interfață. Dacă valoarea măsurată de senzor este mai mare decât limita impusă de către utilizator, se va trimite un email către adresa de mail care este setată în fereastra de *log-in* a aplicației. Procesul se repetă la fiecare 3 minute sau la un intreval de timp stabilit de utilizator. Totodată, utilizatorul are la îndemână un istoric al mail-urilor și un grafic cu datele în timp real colectate de la senzor.

## SCHEMA MONTAJULUI ARDUINO

În Figura 4.2.1 este realizat montajul senzorului de detectare al monoxidului de carbon și modulul care permite conectarea plăcuței la internet, realizată în Tinkercad.

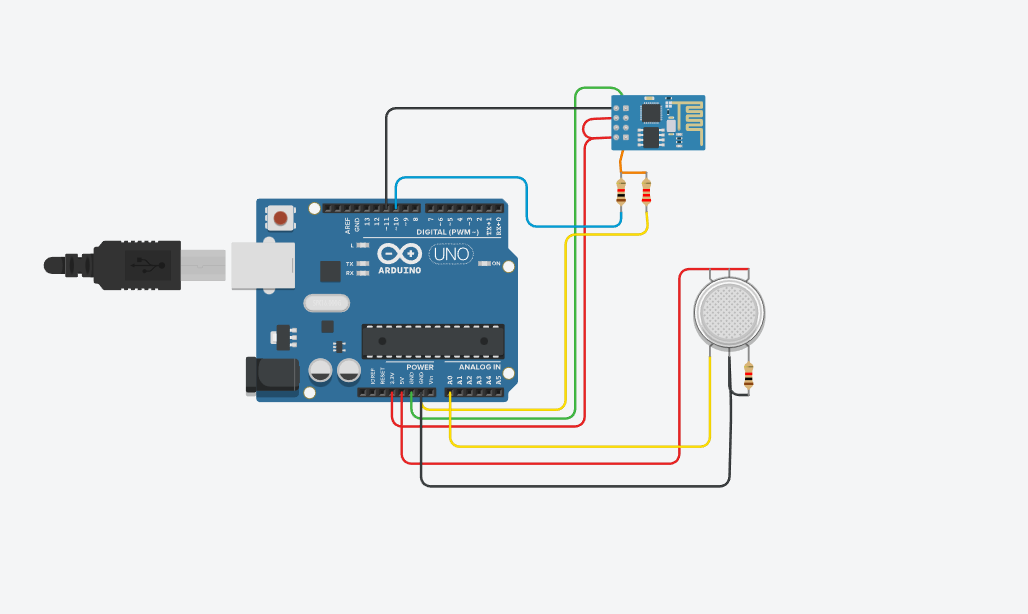


Figura 4.2.1 – Schema montajului Arduino în Tinkercad

# COMPONENTE HARDWARE FOLOSITE

## SENZOR DE MONOXID DE CARBON MQ-7

Senzorul de monoxid de carbon MQ-7 (Figura 5.1.1) este un senzor care are poate fi folosit la o multitudine de aplicații, de la aplicații care constă în monitorizarea locuințelor sau spațiilor în care pot fi degajate cantități de monoxid de carbon până la aplicații mai vaste de uz industrial. Este un senzor care nu are costuri mari de producție și care este ușor de folosit.



Figura 5.1.1 – Senzor monoxid de carbon

Senzorul funcționează pe baza unui filament de dioxid de staniu (SnO2). Dioxidul de staniu are o conductivitate mai mică atunci când în aer nu se află alte gaze. Conductivitatea crește pe măsură ce cantitatea de monoxid de carbon din aer crește, astfel putând fi măsurată. Senzorul poate măsura cantitați între 10ppm până la 10000 ppm. Senzorul funcționează cel mai bine după ce s-a încălzit deoarece gazele care se absorb la temperaturi mai mici de funcționare se disipă.

**Interfața:**

Senzorul dispune de 4 pini, după cum se poate vedea și în Tabelul 5.1.2.

Tabelul 5.1.2 – Interfață senzor MQ-7

|  |  |  |
| --- | --- | --- |
| **Numar Pin** | **Simbol** | **Descriere** |
| 1 | A0 | Iesire Analogica |
| 2 | D0 | Iesire Digitala |
| 3 | GND | Ground(0V) |
| 4 | VCC | Sursa 2.5-5V |

Senzorul dispune de multe avantaje, precum:

* sensibilitate ridicată deoarece poate măsură schimbări minore ale cantității de monoxid de carbon din aer
* are un interval de măsurare foarte mare, ideal și pentru aplicații de uz industrial
* are o durată de viață ridicată
* nu este costisitor din punct de vedere financiar

## MODULUL ESP8266EX

Modulul ESP8266EX (Figura 5.2.1) este o componentă cheie în aplicația de monitorizare a monoxidului de carbon, fără ea nu s-ar putea face trasmiterea de date între partea hardware și cea software. Modulul este un modul Wi-fi care permite conectarea la internet, acesta poate fi folosit singur sau poate fi folosit ca “slave” de către un microcontroller. În cazul de față, modulul ESP8266EX este folosit ca “slave” de către microcontroller.

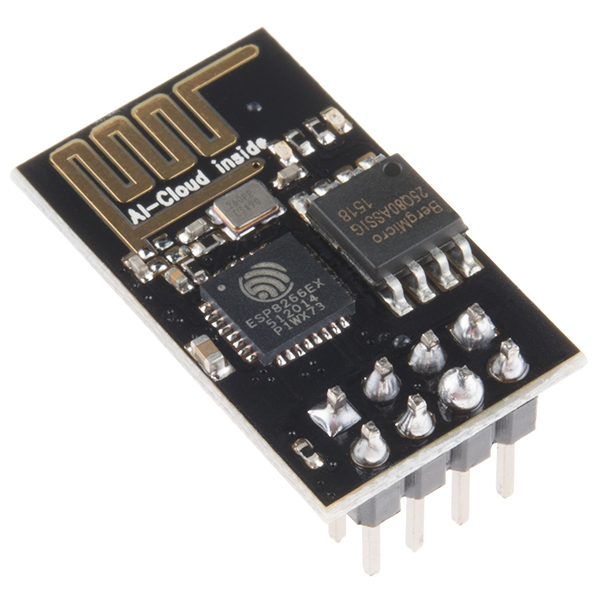


Figura 5.2.1 – Modulul ESP8266EX

**Interfața:**

Modulul dispune de 8 pini, după cum se poate vedea și în Tabelul 5.2.2.

Tabelul 5.2.2 – Interfață modul ESP8266EX

|  |  |  |
| --- | --- | --- |
| **Numar Pin** | **Simbol** | **Descriere** |
| 1 | VCC | Sursa : 3.3V |
| 2 | GND | Ground : 0V |
| 3 | Rx | Receptor- folosit la receptia datelor de la un alt dispozitiv |
| 4 | Tx | Transmițator- folosit la transmiterea datelor catre alte dispositive |
| 5 | Ch\_En | “Chip enable”- activare modul, de obicei conectat la 3.3V |
| 6 | GPIO | Pin Generic – poate fi programat |
| 7 | GPIO 2 | Pin Generic 2 |
| 8 | RST | Reset – folosit la resetare, de obicei conectat la GND(0V) |

Modulul ESP8266EX este o alegere foarte bună deoarece este compact și este foarte accesibil din punt de vedere financiar. Acesta poate folosi chiar și end-point-uri pentru dezvoltarea aplicaților de tip IoT (Internet of Things).

Pentru a comunica cu internetul, modulul are nevoie de o serie de comenzi de tip AT. Dacă totul a fost configurat, modulul o să aibă acces la internet și o să folosească protocolul de comunicare TCP/IP sau UDP.

Comenzile AT suportate de către modulul ESP8266EX sunt listate în Tabelul 5.2.3.

|  |  |  |
| --- | --- | --- |
| **Funcție** | **Comandă AT** | **Răspuns** |
| Funcționare | AT | OK |
| Restart | AT + RST | OK  ...  Ready |
| Versiune de Firmware | AT + GMR | <Verisiunea AT> informatii despre versiunea AT  <Versiunea SDK> informatii despre versiunea de SDK  <timp de compilare> timpul in care bin-ul a fost compilat  OK |
| Lista cu punctele de acces | AT + CWLAP | +CWLAP:<ecn>,<ssid>,<rssi>,  <mac>,<ch><freq offset>  OK |
| Detealii despre punctele de acces la care se poate conecta modulul | AT + CWJAP ? | + CWJAP: <ssid>,<bssid>,<channel>,<rssi>  OK |
| Conectarea modulului la un punct de acces | AT + CWJAP =”SSID”, “Password” | WIFI Connected  WIFI GOT IP  OK |
| Deconectarea de modului de la un punct de acces | AT + CWQAP | OK  WIFI Disconnected |
| Afisarea adresei IP | AT + CIFSR  (Prima data trebuie setat  AT + CWMODE = 3) | +CIFSR:APIP, <adresa IP>  +CIFSR:APMAC, <adresa mac>  +CIFSR:STAIP, <adresa IP>  +CIFSR:STAMAC, <adresa mac>  OK |
| Afisarea modului Wi-fi | AT + CWMODE ? | +CWMODE:<mod> |
| Setearea modului Wi-fi | AT + CWMODE = <mode>  1 = STA (station)  2 = AP (Access Pont)  3 = Ambele (STA & AP) | OK |
| Detalii despre UDP | AT + CIPMUX ? | +CIPMUX:<mod> |
| Setare UDP | AT + CIPMUX = <mode>  0 = Single Connection  1 = Multiple Connection | OK |
| Detalii despre TCP/IP | AT + CIPSTATUS | STATUS:<status>  Statusuri posibile:  2 : Got IP  3:Connected  4:Disconnected |
| Verificarea modului de transmitere TCP | AT + CIPMODE ? | +CIPMODE:<mod> |
| Setarea modului de trasmitere TCP | AT + CIPMODE = <mode>  0 = Normal mode  1 = Transparent mode | OK |
| Setarea conexiunii UDP | (CIPMUX = 0)  AT + CIPSTART = <type> , <address> , <port>  (CIPMUX = 1)  AT + CIPSTART = <id>, <type>, <address>, <port> | CONNECT  OK |
| Transmitere de date | (CIPMUX = 0)  AT + CIPSEND = <data length>  (CIPMUX = 1)  AT + CIPSEND = <id>, <data length> | OK  Recv<data length> bytes  SEND OK |
| Inchidere conexiune | AT + CIPCLOSE | CLOSED  OK |

# PROTOCOALE FOLOSITE DE ESP8266

## TRANSMISSION CONTROL PROTOCOL (TCP)

Este o metodă de comunicare între două sau mai multe dispozitive care pot accesa o rețea pe internet. TCP este unul dintre metodele de comunicare standard pe internet, acesta este conceput să trimite pachete într-o rețea asigurându-se totodată că informația a fost trimisă și primită cu succes. Este cel mai folosit protocol de transmitere de date în cadrul comunicațiilor în rețea deoarece verifică transmiterea informației de la un capăt la altul. TCP organizează și structurează datele pentru ca ele să poată fi transmise între server și client. În plus, garantează și pentru integritatea datelor care sunt trimise prin rețea. Înainte ca datele să fie trimise, TCP stabilește o conexiune între surse și asigură stabilitatea și integritatea conexiunii până la începerea comunicației. Mai apoi desparte cantitatea mare de informație în pachete mai mici și se asigură ca nimic să nu fie pierdut în timpul procesului. Ca rezultat al faptului că integritatea datelor este garantată de TCP, toate protocoalele de nivel înalt care au nevoie de transmitere de date îl folosesc, cum ar fii rețelele peer-to-peer (egal-la-egal), File Transfer Protocol (FTP), Secure Shell(SSH), si Telnet.

## USER DATAGRAM PROTOCOL (UDP)

UDP este un protocol de trasmitere de date pe internet, folosit în special la trasmisiile la care nu contează integritatea datelor trimise, ci viteza cu care ele au fost trimise. Viteza crescută rezultă din faptul că UDP nu stabilește o conexiune înainte ca datele să fie trimise, el permite ca datele să fie trimise la o viteză foarte mare, dar, în urma neglijării pachetelor trimise, unele ar putea fi pierdute în timpul transmiterii, creând oportunități pentru atacuri de tip DdoS (Distributed Denial of Service). UPD realizează transferul de date într-o manieră mai simplă, trimite pachetele direct la un dispozitiv din rețea fără a mai stabili o conexiune în prealabil, indicând doar ordinea în care s-au trimis pachetele, fără a mai verifica dacă ele au ajuns cu succes sau nu.

## INTERNET PROTOCOL SUITE (TCP/IP)

TCP/IP este un set de protocoale de comunicații care permite o trasmitere a datelor foarte exactă și corectă din punct de vedere al integrității datelor. El desparte datele în pachete pentru a evita retrimiterea integrală a datelor în cazul în care trimiterea eșuează. Fiecare pachet poate lua o rută diferită între cele două dispozitive, dacă ruta pe care a fost trimis pachetul anterior este indisponibilă. La final, după ce toate pachetele au fost trimise cu succes, ele se reasamblează automat formând mesajul integral. TCP/IP simplifică comunicarea, aceasta fiind împarțită în patru straturi fără a avea implicări hardware sau software. Pachetele trebuie să treacă prin toate cele patru straturi ale protocolului înainte de a ajunge la destinația finală, după ce pachetele au ajuns, TCP/IP parcurge invers cele patru straturi, reasalmblând mesajul final. TPC/IP, fiind un protocol care se bazează pe conexiunea între dispozitive, acesta folosește “three-way handshake” pentru a stabili conexiunea. Serverul primește un pachet pentru sincronizare, prin care știe că urmează să se conecteze un client. În al doilea pas, serverul trimite un răspuns clientului, prin care îl anunță că a primit pachetul, iar in final, clientul trimite încă un răspuns serverului pentru și pentru a stabili conexiunea.

**Cele 4 straturi care definesc protocolul TCP/IP:**

1. *Interfața de rețea.*

Interfața de rețea definește modul în care datele ar trebui trimise, se ocupă de actul fizic de recepție și transmitere de date și este responsabilă cu transmiterea datelor între aplicații sau dsipozitive în rețea. Acestea includ modul în care datele ar trebui semnalate de componetele hardware și de celelalte dispozitive din rețea.

1. *Internet*

Stratul Internet este responsabil cu trimiterea pachetelor și cu controlul mișcării lor de-a lungul rețelei, pentru a se asigura că pachetele ajung la destinație.

1. *Transport*

Stratul de transport implică producerea unei conexiuni solide și stabile între aplicatia sau dispozitivul care trimite date și destinația dorită. Acesta este nivelul unde data este desparțită în pachete și este numerotată pentru a se crea o secvență. Totodată stratul de transport, determină cantitatea de date care se trimite, unde trebuie trimisă și rata la care se trimit datele.

1. *Aplicație*

Stratul aplicație se referă la faptul că dispozitivele au nevoie de TCP/IP pentru a le ajuta la comunicare. Este stratul unde utilizatorii interacționează, cum ar fi sisteme de e-mail sau platforme de comunicare prin mesaje.

# LEGĂTURA DINTRE JAVA ȘI ANDROID STUDIO

După cum am precizat în capitolul 3 „PROGRAME FOLOSITE ÎN IMPLEMENTAREA APLICAȚIEI”, Android Studio este un IDE care suportă un număr foarte mare din librăriile care fac parte din Java 5.0, de aceea putem spune că Android Studio este bazat pe Java. Majoritatea librăriilor care nu sunt suportate de către Android Studio din Java, fie sunt înlocuite cu alte librării, fie nu sunt necesare. Una dintre diferențele majore care apare în Android studio este faptul că metoda main nu există, aceasta fiind înlocuită cu metodele onCreate, onPause sau onResume, metode care trebuie suprascrise de către dezvoltatori.

Pentru a întelege mai bine legătura dintre Java și Android Studio avem Figura 7.1.1. Putem observa o vedere în ansamblu a sistemului de Build pe care îl folosește Android Studio.

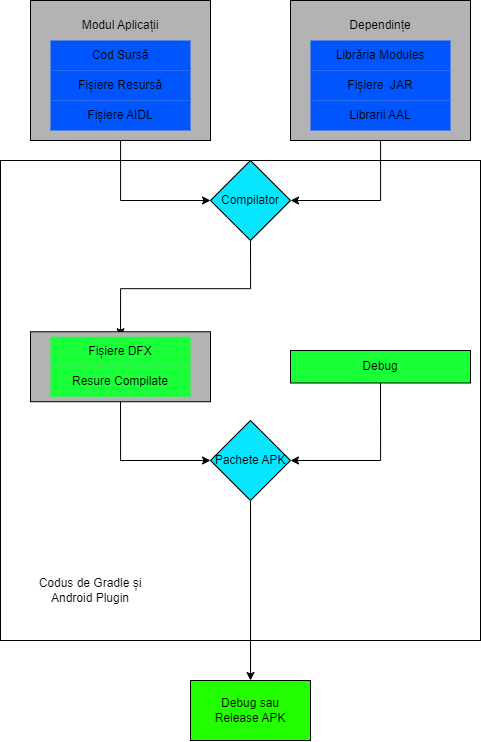


Figura 7.1.1 – Sistem de build din Android Studio

**Javac and dx**

Acesta a fost toolchain-ul lansat drept parte a Android SDK, codul java este luat și convertit în Java bytecode, prin compilatorul Java, javac, care include sursele generate de java annotation processor (ex. Lombok). După care, în final codul este tradus în dex bytecode de către dx. Datorită acestor lucruri, putem spune că Android Studio este bazat în mare măsura pe Java. Divergența apare în toolchain când codul este tradus în cod specific Andriod Studio. Dar oricum ar fii, rezultatul final este mai mult sau mai puțin la fel cu cel din Java.

**Ce este Java?**

Java este un limbaj de programare orientat pe obiecte și, totodată, este un software care rulează pe miliarde de dispozitive din întreaga lume, incluzând calculatoare, telefoane mobile, console de jocuri video și dispozitive medicale. Sintaxele din java sunt bazate pe sintaxele limbajelor de programare C și C++. Unul dintre cele mai mari avantaje este protabilitatea, deoarece dacă se scrie cod pentru software-ul unui calculator, codul este foarte ușor de adaptat pentru trecerea la un telefon mobil. Pentru a scrie programe în Java, avem nevoie de Java Developement Kit. La fel ca și în Android Studio, programele scrise în java sunt compilate și transformate în Java bytecode. Java este un limbaj orientat pe obiecte, în alte cuvinte, java folosește obiecte ca prima sursă de implementare. Limbajele orientate pe obiect au ca obiectiv implementarea de entități din viața de zi cu zi, precum moștenirea și polimorfismul.

**Conceptele programării orientate pe obiect**:

1. Clasele: sunt un set de obiecte care au atribute caracteristici și proprietăți comune.

2. Obiectele: sunt o entitate de bază a limbajelor OOP și reprezintă entități din viața reală.

3. Metodele: sunt o colecție de pași care execută și returnează un rezultat apelantului.

# REST API’S

## API ȘI REST API

Pentru a afla ce este un REST API, prima dată trebuie să înțelegem conceptul de API. Un API (application programming interface) este un set de definiții și protocoale ce ajută la construirea și care se pot integra in aplicații software. Api-urile mai pot fi vazute ca și o ințelegere intre un furnizor de informație și un consumator de informație. Dacă se dorește comunicarea cu un sistem pentru a primi informații sau pentru a apela o funcție, un API este cel care ajută și face posibilă comunicarea. În Figura 8.1.1 este prezentată modul în care comunica API-urile cu diferite dispozitive.

A white rectangular sign with black text

Description automatically generated with low confidence

Figura 8.1.1 – Cum funcționează API-urile

REST (representational state transfer) este un tip de arhitectură software care a fost creată pentru a face sigură operabilitatea între mai multe sisteme. Serviciile care sunt construite conform arhitecturii REST pot comunica între ele mult mai usor.

Cum am precizat, REST este un tip de arhitectură pentru programele software, iar REST API face referire la un API care face un apel de tip HTTP pentru a accesa și a folosi datele. Datele pot fi folosite pentru mai multe metode precum GET, PUT, POST și DELETE, fiecare are un rol diferit, GET citește datele, PUT face update, POST crează date, iar DELETE, după cum spune și numele, le șterge. REST API-urile sunt folosite foarte mult în partea de cloud deoarece ajută utilizatorii să se conecteze și să acceseze servicii de tip cloud, oferindu-le flexibilitate foarte mare.

## CUM FUNCȚIONEAZĂ REST API-URILE

REST API-urile folosesc metodele HTTP pentru a obține resurele necesare, aceste metode sunt definite de către protocolul RFC 2616, metode precum:

* GET
* PUT
* POST
* DELETE

Fiecara metodă are o sarcină specială. Totodată aceste metode sunt folosite și la identificare REST API-urilor, cand endpoint-urile sunt la fel.

* Metoda GET: folosită pentru a recupera o resursă
* Metoda PUT: folosită pentru a modifica o resursă
* Metoda POST: folosită pentru a crea o resură
* Metoda DELETE: folosită pentru a șterge o resursă

Când se apelează un REST API, apelul poate sa conțină un corp care poate fi în mai multe formate de text, cele mai cunoscute și folosite ar fi:

* Application/json
* Application/xml
* Application/x-wbe-xml

## REST API-URI DEZVOLTATE DE THINGSPEAK

API care permite scriere de date pe un canal:

<https://api.thingspeak.com/update>

Acesta va trebui sa fie însoțit neapărat de api\_key, cheie care este necesară pentru a indica canalul pe care se va scrie informația. Alte câmpuri care sunt opționale ar mai fi:

* field<X>, unde x este id-ul câmpului în care vrem să se scrie informația
* lat, latitudinea in grade, cu o valoare specifică între -90 și 90
* long, longitudinea în grade cu o valoare specifică între -180 și 180
* status, face actualizare la mesaje.

API care permite să se citească date de pe un canal:

[https://api.thingspeak.com/channles/<channel\_id>/feeds.<format](https://api.thingspeak.com/channles/%3cchannel_id%3e/feeds.%3cformat)>

La fel ca si API-ul va trebui sa fie însoțit neapărat de api\_key, care din nou este necesară pentru a indica canalul de pe care se va citi informația. Câmpuri care nu sunt necesare:

* result: numărul de entități care sunt extrase
* min: valoarea minimă inclusă în răspuns
* max: valoarea maximă inclusă n răspuns
* status: include statusul în răspuns și setează câmpul „status” pe adevărat.

# INTERFAȚA CU UTILIZATORUL

Interfața de logare (Figura 9.1.1): Utilizatorul trebuie să se conecteze cu un cont, email-ul care este setat la contul de utilizator este email-ul la care se vor trimite alertele când se va detecta o creștere semnificativă de monoxid de carbon.

A screenshot of a log in

Description automatically generated with low confidence

Figura 9.1.1 – Interfața de logare

Interfața de înregistrare (Figura 9.1.2): În această interfață, utilizatorul este nevoit să își creeze cont în cazul în care nu are deja un contul creat.

A screen shot of a login form

Description automatically generated with low confidence

Figura 9.1.2 – Interfața de înregistrare

Interfața principală (Figura 9.1.3): În această interfață sunt afișate datele înregistrate de către senzor, câmpul pentru setarea threshold-ului, oprirea notificărilor pe email și setarea timpului de evaluare al datelor citite de senzor.

A screenshot of a device

Description automatically generated with low confidence

Figura 9.1.3 – Interfața principală

Interfața de setări (Figura 9.1.4): Sunt afișate toate setările, se pot trimite email-uri pentru a verifica dacă este funcțională trimiterea, se poate seta altă adresă la care să se trimită email-uri, totodată se poate afișa un istoric pentru email-urile ce au fost trimise și se poate contacta developer-ul.

A screenshot of a phone

Description automatically generated with low confidence

Figura 9.1.4 – Interfața de stări

Interfața pentru setarea adresei de email (Figura 9.1.5): Aici se pot configura adresa de email, subiectul emailului, cât și mesajul care să fie trimis în caz că există creșteri de monoxid de carbon.

A picture containing text, screenshot, software, multimedia software

Description automatically generated

Figura 9.1.5 – Interfața pentru setarea adresei de email

Interfata istoricului (Figura 9.1.6). În interfața aceasta regăsim un mic istoric cu toate email-urile ce au fost trimise, data la care s-au trimis și threshold-ul care a fost atins.

A picture containing text, screenshot

Description automatically generated

Figura 9.1.6 – Interfața istoricului

Interfața de contact (Figura 9.1.7): Aici se pot trimite email-uri către email-ul de contact al developerului, în cazul în care se semnalează comportamente neprevăzute ale aplicației.

A picture containing text, screenshot, multimedia software, software

Description automatically generated

Figura 9.1.7 – Interfața de contact

Interfața cu graficul valorilor măsurate (Figura 9.1.8): Reprezentarea grafică în timp real ale valorilor măsurate de căatre senzorul de monoxid de carbon.

A picture containing text, screenshot, multimedia software

Description automatically generated

Figura 9.1.8 – Interfața cu graficul valorilor măsurate

În Figura 9.1.9 este prezentat modul în care se poate parcurge aplicația mobilă și în ce fel sunt conectate interfețele.

A picture containing screenshot

Description automatically generated

Figura 9.1.9 – Ghidul pentru interfața cu utilizatorul

# IMPLEMENTARE

## IMPLEMENTARE HARDWARE

Pentru a citi datele de la senzor am scris un program în Arduino IDE care preia datele de la senzorul de monixid de carbon MQ-7, acesta le citește pe portul analogic al plăcii de dezvoltare, după care le trimite către ThingSpeak.

1. Librăria SoftwareSerial.h

Librăria pe care am importat-o, SoftwareSerial.h, face comunicarea dintre modulului ESP8266 și placa de dezvoltare Arduino posibilă. Metodele din librăria importată sunt:

* + SoftwareSerial(rx, tx): metoda care ne permite sa creăm un obiect de tipul SoftwareSerial
  + begin(viteza(baud\_rate)): setarea vitezei de transfer
  + pritnln(“mesaj”): scriere mesajului, comanda aceasta permite configurarea modulului ESP

Am început prin importarea librăriei SoftwareSerial.h, aceasta ne permite sa configuram modulul ESP8266. Dupa care am creat un obiect de tip SoftwareSerial, la care am setat RX si TX pe porturile 10, respectiv 11. Acest obiect o să ne ajute mai târziu cu configurarea modulului, si cu trimiterea de date catre ThingSpeak.

1. Metoda void setup() (Figura 10.1.1)

A screenshot of a computer program

Description automatically generated with medium confidence

Figura 10.1.1

În metoda de setup, am setat modul pinului la care este conectat senzorul de monoxid de carbon ca fiind un pin de input, deoarece pe pinul acesta se va face citirea datelor senzorului.După care am setat ca transmiterea serială să se inceapa cu un baud rate, adica cați biti să se transmită pe secundă, de 9600, iar pentru modulul ESP8266 am setat baud rate-ul ca fiind 115200.

Dupa ce am facut setările de bază, am început configurarea modulului ESP8266. Am început prin a seta modul Wifi-ului, cu comanda AT+CWMODE = 1, aceasta comandă setează modul Wifi-ului pe Station, modul acesta ne permite sa conenctăm modulul ESP8266 la un punct de acces Wifi. Comanda AT + CWJAP conectează modulul la rețeaua care este menționata, primul parametru fiind SSID-ul rețelei la care vrem sa ne conectăm iar cel de al doilea parametru este parola cu care ne conectăm la rețea. Dupa fiecare comandă am setat un delay, pentru a aștepta răspunsurile in monitorul serial.

1. Medtoda void loop() (Figura 10.1.2)

În metoda loop am preluat datele de la senzorul MQ-7.Cu metoda analogRead(), care ia ca parametru numarul pinului la care este conectat senzorul, aceasta metoda citește valorile specifice pinului analogic. Dupa care se setează AT+CIPMUX pe 0, astfel se notifică modulul ESP2866 că va urma să realizam o conexiune de tip “Single Connection”. Cu urmatoarea commandă, AT+CIPSTART, se realizeaza o conexiune de tip TCP, la adresa “api.thingspeak.com”, pe portul 80. AT+CIPSEND ne indica lungimea șirului de date pe care vrem sa-l trimitem, in cazul nostru, lungimea șirului de date trimis este 51. Mai apoi, se face o cerere cu metoda HTTP, GET, catre API-ul de update dezvoltat de catre ThingSpeak, in care este precizata cheia canalului unde urmează sa fie scrise datele senzorului cât și valoarea citită de le senzorul de monoxid de carbon. Ultima comandă, este cea care inchide conexiunea intre modulu Wifi ESP2866, si canalul de ThingSpeak. Delay-ul setat dupa fiecare comandă este o măsură de precauție, și este necesară deoarece unele comenzi durează o perioada de timp până când se îndeplinesc, iar acel delay, reprezintă siguranța că toate comenzile au fost primite si realizate cu succes.

A screenshot of a computer program

Description automatically generated with low confidence

Figura 10.1.2 – Metoda void loop()

## CONFIGURAREA CANALULUI DIN THINGSPEAK

Pentru realizarea aplicației, este necesar un cont de utilizator pe platforma ThingSpeak, mai multe detalii despre platformă pot fii regasite la subcapitolul 3.3 THINGSPEAK, după ce utilizatorul este conectat, trebuie să se creeze un canal pentru a primi și prelua date.

Trebuie să mergem la „My channels” (Figura 10.2.1) și să apăsam pe butonul „New Channel”, acesta o să deschidă o pagină în care va trebui să realizăm setările canalului.

A picture containing text, website, web page, screenshot

Description automatically generated

Figura 10.2.1 – „My channels” in ThingSpeak

Apoi în urmatoarea pagină (Figura 10.2.2) va trebui să se seteze canalul, să fie setat numele, numele câmpurilor pe care vom avea nevoie să scrie sau să citim date și multe alte setări care nu sunt neapărat necesare.

A screenshot of a computer

Description automatically generated with medium confidence

Figura 10.2.2 – Pagina „My Channels”

După ce tot a fost setat și configurat, interfața ar trebui să arate ca în Figura 10.2.3. Următorul lucru care necesită abordat este salvarea cheii reprezentative canalului, ea poate fi gasită la „API Keys”.

A screenshot of a computer

Description automatically generated

Figura 10.2.3 – Setările canalului

În pagina „API Keys” (Figura 10.2.4) se găsesc diferite informații despre cheile care ajută la scrierea si citirea datelor de pe canal, cheile sunt diferite, una este pentru citirea datelor, iar cealaltă este pentru scrierea lor. Ba mai mult, sunt câteva exemple cu API-uri, pentru a fi mai ușor de ințeles.

A screenshot of a computer

Description automatically generated with medium confidence

Figura 10.2.4 – Pagina „API Keys”

## IMPLEMENTAREA SOFTWARE

**Clasa DataTableDetails** (Figura 10.3.1)

Clasă care implementează interfața „Parcelable” și care are ca atribute următoarele:

* private String mContact
* private String mThreshold
* private String mDate

Aceste atribute vor fi afișate mai târziu în istoricul de e-mail-uri trimise. Clasa implementează interfața „Parcelable” deoarece este necesar ca instanțele clasei să poată fi trimise prin Intent, dintr-o activitate în alta. Tot în această clasă regăsim și un constructor, care ne permite sa creăm obiecte. Din moment ce clasa implementează interfața Parcelable, este nevoie și de implementarea metodelor care se regăsesc în interfață. Se poate observa metoda „writeToParcel” care are adnotarea „@Override”. Metoda care are ca parametrii un obiect de tip „Parcel” și paramatrul „flags” de tip intreg, și care transferă datele în obiectul de tip „Parcel”.

A screenshot of a computer program

Description automatically generated with medium confidence

Figura 10.3.1 – Clasa „DataTableDetails”

Alte metode care sunt implementate in clasa DataTableDetails ar fi getter-ele și setter-ele (Figura 10.3.2) care sunt folosite pentru setarea și extragerea informațiilor din obiectele create. Clasa „Creator” este o sub-interfață, care conține metodele „createFromParcel” și „newArray”. Aceste două metode trebuie implementate în clasa „DataTableDetails”.

A screen shot of a computer program

Description automatically generated with low confidence

Figura 10.3.2 – Getter-ele si Setter-ele calsei „DataTableDetails”

**Clasa „EmailDto”** (Figura 10.3.3) are ca atribute urmatoarele câmpuri:

* private String email
* private String subject
* private String message

Clasa vine în ajutorul gestionării utilizatorilor și salvării datelor lor. Fiecare instață a clasei conține o adresă de e-mail, subiectul e-mail-ului și mesajul care va fi trimis. În clasă sunt implemementați constructorii, unul cu toate atributele clasei, iar celălalt fără nici un atribut. Tot în clasa, pentru accesarea și scrierea de date în obiectele create, sunt implementate și metodele de „get” și „set”.

A screenshot of a computer program

Description automatically generated with medium confidence

Figura 10.3.3 – Calsa „EmailDto”

**Clasa „DBUtils”** (Figura 10.3.4)

Clasă ce ne ajută la definirea și implementarea metodelor pentru baza de date în care sunt salvate datele utilizatorilor.

A screen shot of a computer code

Description automatically generated with low confidence

Figura 10.3.4 – Clasa „DBUtils”

În clasă este declarată o constantă care reține numele bazei de date. Clasa „DBUtils” extinde clasa „SQLiteOpenHelper”, deci clasa „SQLiteOpenHelper” este clasa părinte pentru clasa „DBUtils”. De aceea în constructorul clasei „DBUtils” este cuvântul cheie „super”, acest cuvânt cheie, apelează constructorul clasei părinte, „SQLiteOpenHelper”.

**Metodele suprascrise din clasa DBUtils** (Figura 10.3.5)

Metoda „onCreate(SQLiteDatabase myDb)” face ceea ce îi sugerează numele, adică crează o bază de date locală printr-un query de tip SQL. Baza de date este creată cu câmpurile „email” de tip TEXT și care este și cheia primară a bazei de date și în care este salvată adresa de e-mail și „password” tot de tip TEXT care reține parola utilizatorului și metoda „onUpgrade(SQLiteDatabase myDb)” care șterge baza de date.

A screenshot of a computer program

Description automatically generated with low confidence

Figura 10.3.5 – Metodele suprascrise din clasa „DBUtils”

**Metoda „insertDataIntoDb(String email, String password)”** (Figura 10.3.6)

Metodă creată pentru a insera utilizatori în baza de date. În metodă se deschide conexiunea cu baza de date prin metoda „getWritableDatabase()”, iar prin cuvantul cheie „this” se face referire la obiectul curent al clasei. După care se crează un obiect „ContentValues” în care se vor stoca datele utilizatorului. La final se vor insera datele utilizatorului în baza de date și se va verifica dacă inserarea a fost realizată cu succes.

**Metoda „checkIfUserExists(String email)”** (Figura 10.3.6)

Elementele din baza de date trebuie să fie unice, pentru ca un utilizator sa poată dispune doar de un cont pe o singură adresă de mail. Metoda „checkIfUserExists” execută un query în baza de date cu care selectează toate obiectele care au email-ul egal cu cel transmis ca parametru și salvează rezultatul într-un obiect de tip Cursor, obiect care este returnat de metoda „rawQuery”. După ce query-ul a fost executat, se verifică dacă numărul de elemente din cursor este mai mare decât 0, returnând adevărat dacă este mai mare și semnalând faptul că utilizatorul a fost deja creat.

**Metoda „loginCheck(String email, String password)”** (Figura 10.3.6)

Metodă care are ca parametrii email-ul utilizatorului și parola acestuia, verifică dacă utilizatorul are cont deja creat în baza de date sau dacă parola acestuia se potrivește cu emai-ul. Și în această metodă se execută un query de tip SQL în baza de date care selectează toate obiectele care au email-ul și parola la fel ca cele transmise ca parametrii, apoi se verifică câte obiecte ce îndeplinesc aceste condiții sunt în cursor. Dacă utilizatorul are un cont creat, rezultatul metodei „cursor.getCount()” va fi 1 și metoda va returna adevărat.

A screenshot of a computer program

Description automatically generated with low confidence

Figura 10.3.6 – Metodele ajutătoare pentru introducerea de conturi în baza de date

**Clasa „MailUtils” (**Figura 10.3.7) S-au setat două constante, una care reprezintă adresa de email de pe care se vor trimite email-uri către utilizator, iar cealaltă este parola pentru aplicații care este creată în contul de Gmail, ambele fiind necesare pentru a trimite email-uri din aplicație.

A screen shot of a computer

Description automatically generated with medium confidence

Figura (10.3.7) – Clasa „MailUtils”

**Clasa „JavaMailApi”** (Figura 10.3.8):

Țelul clasei JavaMailApi este acele de a trimite email-uri. Atributele acestei clase sunt:

* private Context mContext
* private Session mSession
* private String mEmail
* private String mSubject
* private String mMessage

A screen shot of a computer program

Description automatically generated with low confidence

Figura 10.3.8 – Clasa „JavaMailApi”

În ea este declarat și constructorul care are are ca parametrii toate atributele clasei. Aceasta clasa extinde clasa „AsyncTask” deoarece este nevoie de suprascrierea metodelor care fac parte din această clasă.

**Metoda „onPreExecute()”** (Figura 10.3.9)

Face parte din clasa „AsyncTask” și este suprascrisă în clasa „JavaMailApi”. Metoda deschide un dialog înainte de a se trimite email, si notifică utilizatorul că va urma să se trimită un email.

**Metoda „onPostExecute(Void aVoid)”**(Figura 10.3.9)

Notifică utilizatorul când un email sa trimis cu succes către adresa setată.

A screen shot of a computer

Description automatically generated with low confidence

Figura 10.3.9 – Metodele suprascrise din clasa „JavaMailApi”

**Metoda „doItInBackground(Void ... params)”** (Figura 10.3.10)

În această metodă se trimite mesajul implicit către utilizator. În prima parte a metodei se setează proprietățile și se configurează pentru a putea trimite email-uri pe Gmail. Prima dată se setează hostul, acesta fiind „smtp.gmail.com”, după care se indică port-ul și se trece câmpul pentru autentificare pe „true”. Mai apoi se crează o sesiune nouă cu credențialele setate in MailUtils. După toate acestea se crează un obiect de tip „MimeMessage” cu sesiunea curentă și se configurează cu adresa de email de pe care se trimit email-uri, adresa de email care primește email-uri, cu subiectul si mesajul setat. Cu metoda statică „send(MimeMessage mm)” se trimite email-ul. În caz că ceva nu merge bine, sau se aruncă vreo excepție, aceasta este prinsă și este tratată.

A screen shot of a computer screen

Description automatically generated with low confidence

Figura 10.3.10 – Metodele suprascrise din clasa „JavaMailApi”

**Activitatea „LogIn”** (Figura 10.3.11)

În activitatea LogIn s-a creat un buton care preia datele din casetele de text în care utilizatorul a adăugat adresa de e-mail și parola cu care s-a înregistrat, după care se salvează în două variabile și se verifică dacă ambele casete de text au fost completate, în caz contrar utilizatorul va primi un mesaj în care este notificat că trebuie să completeze ambele câmpuri, după care, se verifică cu metoda „loginCheck” din clasa „DBUtils” dacă utilizatorul există în baza de date, în cazul în care acesta nu există, se afișează mesajul corespunzător, cum că datele introduse nu se află în baza de date sau că nu se potrivesc. Dacă datele sunt corecte, se crează un obiect „intent” care se ocupă cu deschiderea unei noi activități. În intent se adaugă un câmp extra cu adresa de e-mail introdusă în caseta de text, pentru a putea fi folosită la trimiterea de mesaje. Tot în activitatea de „LogIn” se află un buton care deschide activitatea de „Register”.

A screen shot of a computer program

Description automatically generated with low confidence

Figura 10.3.11 – Activitatea „LogIn”

**Activitatea „Register”** (Figura 10.3.12)

După completarea câmpurilor, cu adresa de email, parola și confirmarea parolei, utilizatorul va trebui să apese pe butonul de înregistrare, care va apela metoda „onClick” și va salva datele utilizatorului. În aceeași metodă se verifică dacă toate câmpurile au fost completate și dacă ambele parole coincid, în caz contrat se afișează un mesaj corespunzător pe ecran. Tot aici se verifică daca utilizatorul există deja, cu metodele implementate în clasa „DBUtils”, dacă utilizatorul nu există se inserează în baza de date email-ul și parola acestuia. În cele din urmă se redeschide activitatea de logare, pentru ca utilizatorul să se poată înregistra cu noul cont creat.

A screenshot of a computer screen

Description automatically generated with low confidence

Figura 10.3.12 – Activitatea „Register”

**Activitatea „SenzorValueActivity” (**Figura 10.3.13)

Activitatea „SenzorValueActivity” este activitatea în care sunt definite metodele pentru citirea datelor de pe ThingSpeak și prelucrarea lor.

A screenshot of a computer program

Description automatically generated with medium confidence

Figura 10.3.13 – Activitatea „SenzorValueActivity”

Aici sunt definite metodele „onClick” ale butoanelor, butonul de reîmprospătare apelează metoda „getRequest(url)”, iar în celelalte metode se deschide activitatea care conține setările și cea care conține graficul cu datele în timp real. La final se setează atributele obiectului „emailDto”, email-ul care provine din activitatea „LogIn”, prealuat din intent și celelalte atribute care sunt declarate drept constante în activitatea curentă.

**Metoda „afterTextChange”** (Figura 10.3.14) execută codul după ce se modifică textul în câmpul din activitatea principală unde se actulizează „threshold”-ul, actualizând și mesajul email-ului, în caz că se întamplă ceva neprevăzut, excepția este tratată și este afișat un mesaj care să notifice utilizatorul.

A picture containing text, screenshot, software, multimedia software

Description automatically generated

Figura 10.3.14 – Metoda „afterTextChange”

**Metoda „onCheckedChanged”** (Figura 10.3.15)

Metodă care determină dacă se trimit sau nu email-uri, acesta verifică dacă butonul de tip „switch” este bifat sau nu, și seteză câmpul „isSendingEmails” în funcție de valoarea butonului.

A screen shot of a computer

Description automatically generated with low confidence

Figura 10.3.15 – Metoda „onCheckedChanged”

Codul din Figura 10.3.16 apelează „run()” o dată la o perioadă de timp, în cazul de față, se apelează la valoarea setată de către utilizator din interfață sau la o valoare standard care este setată.

A picture containing text, screenshot, multimedia software, software

Description automatically generated

Figura 10.3.16 – Apelarea metodei „run”

**Metoda „checIfThresoldIsReached** (Figura 10.3.17) verifică dacă valoarea citită de senzor este mai mare decât pragul setat și dacă butonul care oprește trimiterea de email-uri nu este setat pe adevărat, dacă condiția este adevărată, se trimite email și se adaugă într-un obiect de tip „DataTableDetails” email-ul la care s-a trimis mesaj, data la care s-a trimis și valoarea pragului care a fost depașită pentru a se adăuga mai apoi în istoric.

A screen shot of a computer screen

Description automatically generated with low confidence

Figura 10.3.17 – Metoda „checIfThresoldIsReached”

**Metoda „checkIfSendEmails”** (Figura 10.3.18)

Verifică dacă valoarea senzorului depășește pragul și totdată verifică starea în care se află butonul pentru setarea trimiterii de email-uri.

**Metoda „sendMail”** (Figura 10.3.18) face exact ceea ce îi sugerează numele, trimite email-uri către utilizatorul care este trimis ca parametru, cu mesajul și subiectul respectiv. În această metodă este definit un obiec de tip „JavaMailApi” în care se setează contactul, subiectul și mesajul, după care se apelează metoda „execute()” care declanșează trimiterea email-ului.

A picture containing text, screenshot, software, multimedia software

Description automatically generated

Figura 10.3.18 – Metoda „checIfThresoldIsReached” și Metoda „sendMail”

Metodele din Figura 10.3.19 sunt metode apelate cand se apasă butoanele, acestea crează un intent la care se adaugă câmpurile necesare în celelate activitați, apoi se apelează metoda de strat a activității și se oprește activitatea curentă.

A screen shot of a computer screen

Description automatically generated with low confidence

Figura 10.3.19 – Metode de deschidere a unor noi activități

**Metoda „getRequest”** ( Figura 10.3.20) crează o coadă în care se adaugă un apel cu contextul activitații curente. Se crează un obiect „StringRequest” și se configurează apelul către ThingSpeak, metoda folosită este „GET”, metodă care ne permite sa retragem date din canalul creat și se specifică și url-ul la care să se facă apel.

**Metoda „onResponse”** (Figura 10.3.20) preia ca parametru răspunsul apelului către url și îl prelucrează. Se crează un obiect de tip „JSONObject” și se adaugă răspunsul apelului, după care se preia toate datele din obiectul „arrayOfData” sub forma de „JSONArray”. Se ia ultimul obiect din listă și se transformă din „String” în obiect de tip „JSONObject” pentru a putea fi preluată data din câmpul „field1”, această valoare fiind de fapt valoarea senzorului.

A screen shot of a computer program

Description automatically generated with low confidence

Figura 10.3.20 – Metodele „getRequest” și „onResponse”

În URL-ul din Figura 10.3.21 se poate observa că are mai mulți parametrii, printre care și cheia canalului de pe care se vor retrage date.

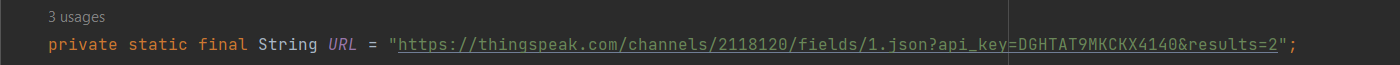


Figura 10.3.21 – Metodele „getRequest” și „onResponse”

Pentru a înțelege mai bine cum funcționează transferul de date între ThingSpeak si aplicația din Android Studio s-a simulat un apel din PostMan către API-ul pus la dispoziție. Se poate vedea un exemplu al unui răspunsul primit de la ThingSpeak în Figura 10.3.22. Se poate observa campul „feeds” care este un vector de obiecte care conține câmpurile „created\_at”, „entry\_id” și „field1” care de fapt este valoarea de monoxid masurată de senzor.

A screenshot of a computer code

Description automatically generated with low confidence

Figura 10.3.22 – Corpul în format JSON al răspunsului din ThingSpeak

**Activitatea „ChartOftDataActivity”** (Figura 10.3.23)

Pentru a crea un grafic în timp real, acesta se folosește de aceeași metodă care se reapealează la un interval de timp setat. Se crează un „lineDataSet” în care se adaugă valoarea citită de la senzor după care se setează în obiectul „mLineChart”, acesta afișând datele într-un grafic.

A picture containing text, screenshot, software, multimedia software

Description automatically generated

Figura 10.3.23 – Crearea Graficului în timp real

**Activitatea „HistoryActivity”** (Figura 10.3.24)

Pentru afișarea tuturor datelor în istoric, datele sunt preluate de la celelalte activitați și sunt parcurse, iar pentru fiecare obiect se setează o nouă linie în tabelul „tableRowForData”, și se adaugă contactul, valoarea pragului și data la care s-a trimis email-ul.

A screen shot of a computer code

Description automatically generated with low confidence

Figura 10.3.24 – Crearea istoricului

# BIBLIOGRAFIE

[1] David G. Penney, ‚,Carbon Monoxide Poisoning’’, CRC Press Taylor & Francis Group, vol. 3, pag. 6-140, pag. 203-305, 2008

[2] Elsevier Ltd, ,,Ambient carbon monoxide and daily mortality: a global time-series study in 337 cities’’, Lancet Planet Health, pag. 191-99, aprilie 2021

[3] Donald M. Yealy, MD, ,,Effects of Carbon Monoxide Poisoning on Epilepsy Development: A Nationwide Population-base Cohort Study’’, Annals of Emergy Medicine, American College of Emergency Physician, vol. 77, februarie 2023

[4] \*\*\*, <https://www.baeldung.com/java-exceptions>

[5] \*\*\*, <https://medium.com/@makkenasrinivasarao1/line-chart-implementation-with-mpandroidchart-af3dd11804a7>

[6] \*\*\*, <https://www.instructables.com/Getting-Started-With-the-ESP8266-ESP-01/>

[7] \*\*\*, <https://www.instructables.com/Learn-How-to-Setup-the-Wifi-Module-ESP8266-by-Usin/>

[8] \*\*\*, <https://www.mathworks.com/help/thingspeak/rest-api.html>

[9] \*\*\*, <https://javaee.github.io/javamail/docs/api/>

[10] \*\*\*, <https://developer.android.com/reference/android/content/Intent>

[11] \*\*\*, <https://cdn.sparkfun.com/datasheets/Sensors/Biometric/MQ-7%20Ver1.3%20-%20Manual.pdf>

[12] \*\*\*, <https://www.electronicwings.com/sensors-modules/esp8266-wifi-module>

[13] \*\*\*, <https://www.fortinet.com/resources/cyberglossary/tcp-ip>

[14] \*\*\*, <https://www.ibm.com/topics/java>

[15] \*\*\*, <https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/>

[16] \*\*\*, <https://weeklycoding.com/mpandroidchart/>

[17] \*\*\*, <https://github.com/Musfick/JavaMailAPIDemo>

[18] \*\*\*, <https://www.w3schools.com/java/java_intro.asp>

[19] \*\*\*, <https://www.w3schools.com/java/java_classes.asp>

[20] \*\*\*, <https://docs.gradle.org/current/userguide/what_is_gradle.html>

[21] \*\*\*, <https://docs.gradle.org/current/userguide/declaring_dependencies.html>

[22] \*\*\*, <https://www.geeksforgeeks.org/methods-in-java/>

[23] \*\*\*, <https://developer.android.com/reference/android/os/Parcelable>

[24] \*\*\*, <https://github.com/Gruzer/simple-gauge-android>

[25] \*\*\*, <https://docs.arduino.cc/learn/built-in-libraries/software-serial#begin>

[26] \*\*\*, <https://docs.espressif.com/projects/esp-at/en/latest/esp32/AT_Command_Set/Wi-Fi_AT_Commands.html>