

**Inginerie software si aplicatii in comunicatii de date**

**Profesor coordonator:**

**Ciobanu Cătălin**

**Participanți:**

**Gheorghe Ștefan – Responsabil documentație**

**Mînecuță Răzvan - Manager**

**Mureșan Vicențiu - Testare**

**Vlăsceanu Nicolae Ovidiu - Development**

**BRAȘOV, 2023**

Cuprins

[1 Optmizarea Supravegherii Pacientilor care sufera problema la nivelul inimii sau al plaminilor 2](#_Toc1882236323)

[1.1 Generalitati 3](#_Toc52996142)

[1.2 Obiective 3](#_Toc1407485259)

[1.3 SUMAR 4](#_Toc368961736)

[2 Componentele folosite pentru realizarea proiectului 4](#_Toc1360797189)

[2.1 Modul senzor puls cardiac GY-MAX30100 4](#_Toc1598540339)

[2.2 ESP32-WROOM-32 6](#_Toc746919645)

[2.3 Breadboard 7](#_Toc388858773)

[3 Arhitectura proiectului 8](#_Toc1814088763)

[3 Mini Tutorial pentru folosire 8](#_Toc30698204)

[4 Tehnologii folosite pentru development 8](#_Toc765930036)

[4.1 Arduino IDE 8](#_Toc41921674)

[4.2 React.js 9](#_Toc1277451234)

[4.3 Firebase 10](#_Toc1261312842)

[4.4 Wifi.h 11](#_Toc260050831)

[5 Repository GitHub 12](#_Toc910046621)

[5.1 Generalitati 12](#_Toc1106456650)

[5.2 Avantaje 12](#_Toc1718919463)

[6 Testare 13](#_Toc858417728)

[7 Management 13](#_Toc698497226)

[8 Lectii invatate 13](#_Toc271407184)

[9 Concluzii 13](#_Toc2049973537)

[10 Appendix 13](#_Toc1256894141)

[11 Codul sursa 13](#_Toc1179400644)

[12 Firmware 13](#_Toc905087419)

[13 Bibliografie 14](#_Toc518982232)

# Optmizarea Supravegherii Pacientilor care sufera problema la nivelul inimii sau al plaminilor

## Generalitati

Ne aflam in 2023, proaspat iesiti din pandemia COVID-19, inceputa pe data de 17 Noiembrie 2019, in Wuhan, China, care a omorit 6,829,605 de oameni din jurul lumii. Chiar si pana in ziua de astazi lumea care a supravietuit acestei boli a ramas cu probleme pulmonare si de inima. Astfel, a devenit din ce in ce mai importanta monitorizarea starii de sanatate si mentinerea unui stil de viata sanatos. Un proiect care poate ajuta la aceste scopuri este Health Monitor.

Health Monitor este un proiect care utilizeaza un modul Arduino, un ESP32 si un pulse oximetru pentru a monitoriza nivelul de oxigen din sange (SpO2) si ritmul cardiac al utilizatorului. Acest proiect poate fi construit pentru a monitoriza starea de sanatate a utilizatorilor si pentru a oferi alerte atunci cand valorile de SpO2 sau pulsul scade sub anumite praguri critice.

Pulse oximetrul este un dispozitiv medical non-invaziv care poate fi atasat la degetul utilizatorului si care masoara nivelul de oxigen din sange, numit SpO2, precum si ritmul cardiac al acestuia. Modulul Arduino poate fi programat sa citeasca valorile de SpO2 si pulsul utilizatorului prin intermediul senzorului din pulse oximetru si sa le trimita catre modulul ESP32. Acesta poate fi apoi conectat la un dispozitiv mobil sau la o platforma cloud pentru a fi monitorizate si inregistrate.

In plus, proiectul Health Monitor poate fi extins prin adaugarea altor senzori sau module, cum ar fi senzorii de temperatura sau de umiditate. Acest lucru poate ajuta la monitorizarea altor aspecte ale sanatatii, cum ar fi nivelul de hidratare sau temperatura corpului.

Proiectul poate fi util pentru persoanele care doresc sa isi monitorizeze starea de sanatate sau pentru cei care au afectiuni medicale. De asemenea, poate fi folosit pentru a monitoriza sportivii si pentru a preveni aparitia unor probleme de sanatate in timpul antrenamentelor.

In concluzie, proiectul Health Monitor poate fi un instrument util pentru monitorizarea starii de sanatate si mentinerea unui stil de viata sanatos, mai ales in contextul actual, proaspat iesiti din pandemia de COVID-19.

## Obiective

Obiectivele pe care noi le propunem prin realizarea acestui proiect sunt urmatoarele:

-evidentierea necesitarii monitorizarii pacientilor cu probleme pulmonare sau de inima, sau a sportivilor in timpul antrenamentelor

-evidentierea si crearea unor statistici/grafice pe termen scurt si indelungat ale fiecarui pacient, date care sunt stocate in cloud

-implementarea unui software a sistemului pentru identificarea fiecarui pacient cu ajutorul datelor personale

-solutie hardware in timp real si software pentru trimiterea datelor despre pulsul pacientului, in cazul in care pulsul scade/creste sub valoarea unui prag critic sa fie anuntat atat pacientul cat si doctorul care se ocupa de acesta printr-un SMS

-implementarea unor grafice zilnice pentru fiecare pacient din ora in ora pentru a putea fi analizate de catre un doctor specializat

## SUMAR

# Componentele folosite pentru realizarea proiectului

## Modul senzor puls cardiac GY-MAX30100

Senzorul GY-MAX30100 este un senzor de oximetrie de tip pulse oximeter, care poate fi utilizat pentru a masura nivelul de oxigen din sange (SpO2) si ritmul cardiac. Acest senzor poate fi integrat in diverse aplicatii, cum ar fi dispozitive medicale portabile, bratari fitness sau ceasuri inteligente.

Senzorul GY-MAX30100 utilizeaza tehnologia reflectometrica pentru a masura nivelul de oxigen din sange. Acesta emite o lumina infrarosie si rosie prin intermediul unui LED si masoara cantitatea de lumina reflectata de la piele. In functie de nivelul de oxigen din sange, hemoglobina din sange absoarbe mai mult sau mai putina lumina infrarosie, ceea ce poate fi detectat de senzor. Senzorul poate detecta, de asemenea, fluctuatiile in intensitatea luminii reflectate, care sunt generate de pulsul cardiac al utilizatorului.

Senzorul GY-MAX30100 are o interfata I2C pentru comunicarea cu un microcontroller, cum ar fi Arduino. Acesta poate fi setat pentru a emite date de SpO2 si puls printr-un singur registru de control si poate fi configurat pentru a efectua masuratori continue sau pentru a masura doar la cerere. Senzorul este, de asemenea, dotat cu un filtru digital pentru a elimina zgomotul si interferentele.

Specificatii:

* Tensiune alimentare: 3 – 5 VDC
* Aplicatii: Pulse oximeter (SpO2), Heart rate monitor
* Sticla acoiperire senzor optic
* Pini: GND, SCL, VIN, SDA, INT
* Curent led si rata refresh programabile pentru optimizarea consumului
* Consum current in mod oprit: 0.7 μA Typ
* Consum monitorizare puls: <1mW
* Output de date rapid
* Dimensiuni mm: 12.7 x 12.7mm
* Rezistenta la interferente date de miscare
* Temperatura operare: -40 C - +85 C

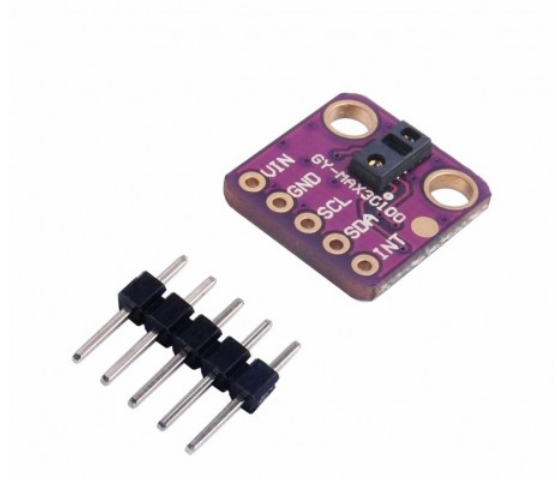


Figura 2.Imagine pulsoximetru GY-MAX30100 fata

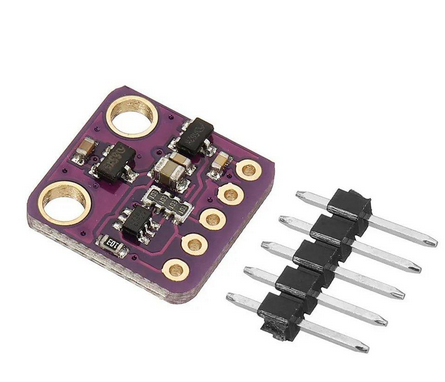


Figura 3.Imagine pulsoximetru GY-MAX30100 spate

## ESP32-WROOM-32

ESP32-WROOM-32 este un modul de dezvoltare Wi-Fi și Bluetooth dual-core bazat pe chipsetul ESP32 al companiei Espressif. Acesta este unul dintre cele mai populare module din gama ESP32, datorita puterii de procesare, a conectivitatii Wi-Fi si Bluetooth si a versatilitatii sale.

Acest modul poate fi programat cu usurinta cu ajutorul unui mediu de dezvoltare integrat (IDE), cum ar fi Arduino IDE sau platforma de dezvoltare ESP-IDF (Espressif IoT Development Framework). Modulul este dotat cu o serie de pin-uri GPIO care permit conectarea la o varietate de componente si senzori, cum ar fi senzori de temperatura, accelerometre, LED-uri si multe altele.

ESP32-WROOM-32 este utilizat in diverse proiecte si aplicatii IoT, cum ar fi sistem de supraveghere si control, automatizarea caselor inteligente, bratari de fitness sau dispozitive medicale portabile. Acesta este apreciat pentru performantele sale bune, conectivitatea puternica si versatilitatea in dezvoltarea de solutii IoT.

Specificatii:

* Procesor: Dual-core Tensilica LX6 cu frecventa de pana la 240MHz
* Memorie flash: 4MB
* Memorie SRAM: 520KB
* Modulul suporta conexiune WI-Fi 802.11 b/g/n
* Suporta Bluetooth
* Tensiune de functionare: 3.3V

Pentru acest proiect noi am folosit urmatorele conexiuni ale pinilor:

* + Alimentarile ambelor componente se face prin intermediul modului ESP32 de la orice sursa de curent prin Mini USB
  + SCL (Serial Clock) într-un puls oximetru este utilizat în comunicarea serială între microcontroler și senzorul de oximetrie. Acesta este un semnal de sincronizare care determină momentul trimiterii și recepționării datelor de la și către senzor.
  + SDA (Serial Data) într-un puls oximetru este utilizat în comunicarea serială între microcontroler și senzorul de oximetrie. Acesta este un semnal care transmite datele de la și către senzor.
  + GND este folosit pentru a conecta senzorul la pământul comun al circuitului, astfel încât măsurătorile să fie făcute relativ la aceeași referință de tensiune

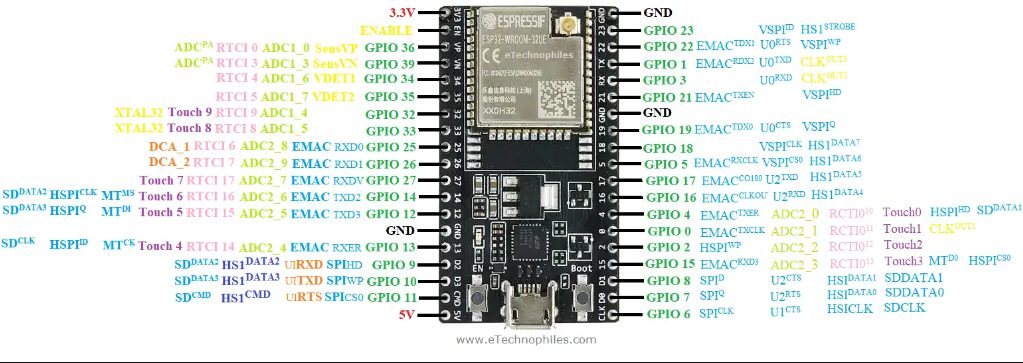


Figura 4.Functionalitati pini ESP32-WROOM-32s



Figura 5.Imagine ESP32-WROOM-32S

## Breadboard

Breadboard-ul este o componenta electrica cu ajutorul careia utilizatorul poate distribuii sarcina electrica intr-un mod eficient, conexiunile fiind pre-cablate sub fiecare gaura. Aceste conexiuni permit utilizatorilor să realizeze conexiuni rapide și să efectueze modificări la circuit fără a fi necesară deplasarea componentelor.

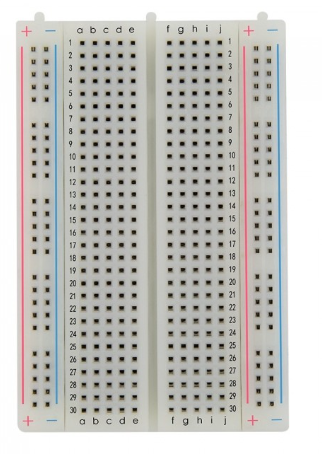


Figura 6.Breadboard 400

# Arhitectura proiectului

## Diagrama electrica

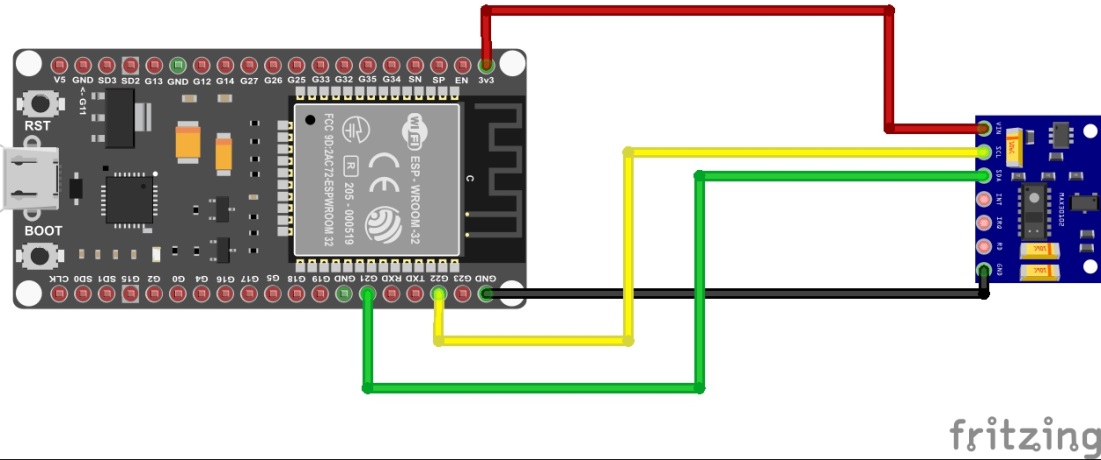


Figura 7. Diagrama electrica

## Schema electrical

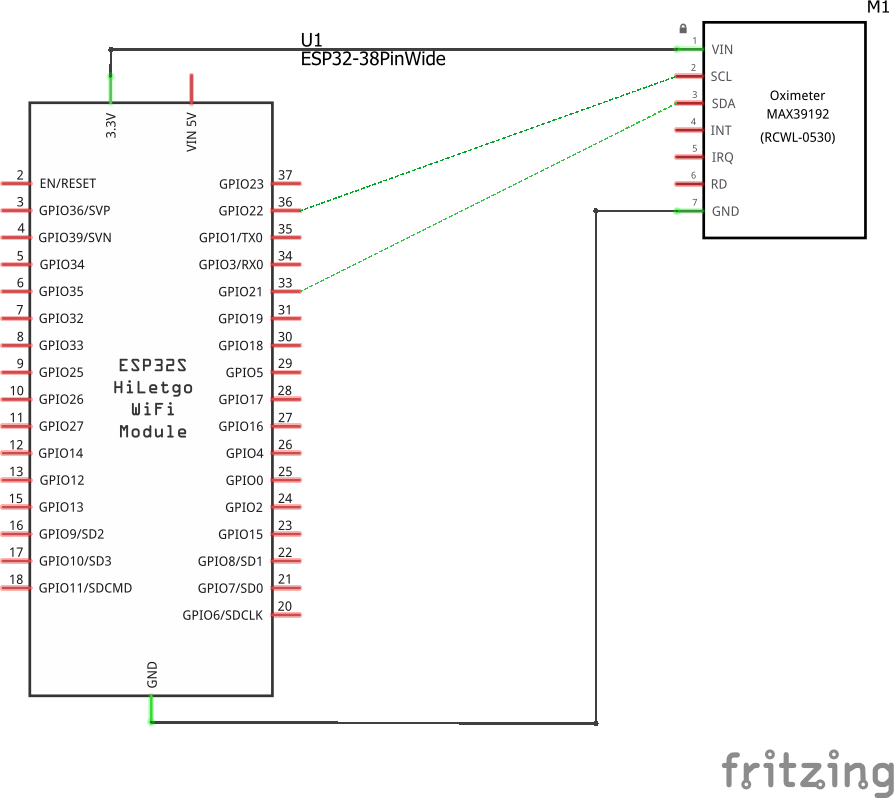


Figura 7. Schema electrica

# Mini Tutorial pentru folosire

# ****Tehnologii folosite pentru development****

## Arduino IDE

ArduinoIDE este un mediu de dezvoltare integrat (IDE) gratuit și open-source, folosit pentru programarea și dezvoltarea platformelor de prototipare hardware bazate pe placa Arduino. Acesta oferă o interfață simplă și intuitivă pentru programarea și depanarea codului, utilizând o limbaj de programare ușor de învățat și de folosit, numit Wiring. Cu ajutorul ArduinoIDE, utilizatorii pot crea proiecte interactivi și experimentale care implică controlul diverselor senzori și actuatori, cum ar fi LED-uri, motoare, senzori de temperatură și umiditate, senzori de lumină, accelerometre și multe altele. Aceste proiecte pot fi conectate la o varietate de platforme de calcul, inclusiv calculatoare, tablete și smartphone-uri, pentru a oferi interacțiunea utilizatorului.

ArduinoIDE oferă, de asemenea, o bibliotecă extinsă de cod predefinit pentru a face mai ușoară dezvoltarea proiectelor complexe. Utilizatorii pot descărca și instala noi biblioteci și exemple de cod din comunitatea Arduino, care oferă o mulțime de resurse și suport pentru începători și avansați.

Un alt avantaj al ArduinoIDE este că este compatibil cu mai multe plăci Arduino, inclusiv Arduino Uno, Arduino Nano și Arduino Mega, printre altele. Acest lucru oferă utilizatorilor posibilitatea de a lucra cu plăci diferite, în funcție de cerințele proiectului lor.

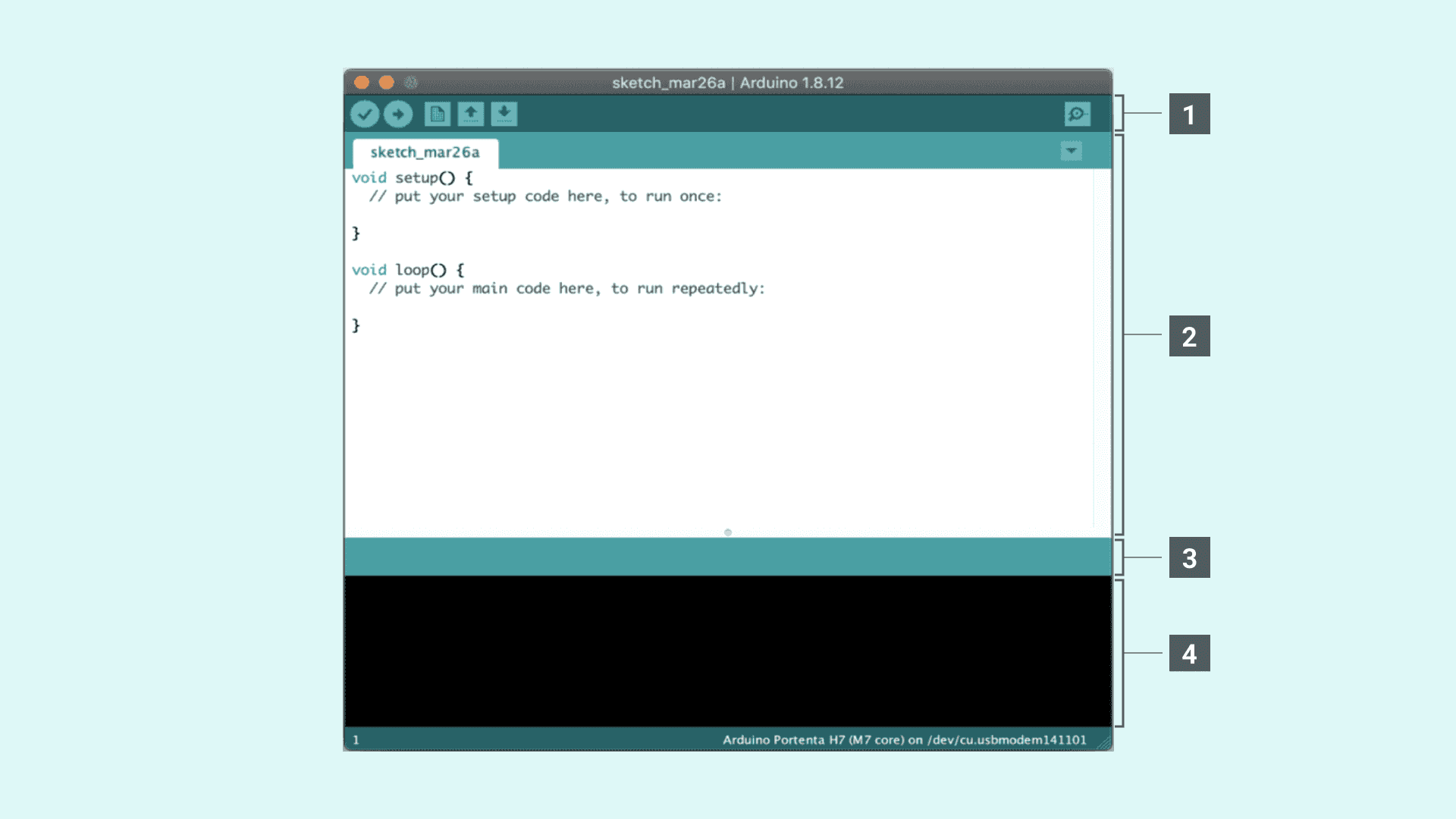


Figura 8.Arduino IDE Software

## React.js

React.js este o bibliotecă open-source pentru construirea interfețelor utilizator în aplicații web, dezvoltată de Facebook. Aceasta este o bibliotecă JavaScript care utilizează un model de programare declarativ pentru a construi interfețe utilizator complexe din componente simple și reutilizabile.

React.js este de obicei utilizat împreună cu alte tehnologii web, cum ar fi Redux, React Router, TypeScript și multe altele pentru a oferi funcționalități extinse pentru dezvoltatorii de aplicații web.

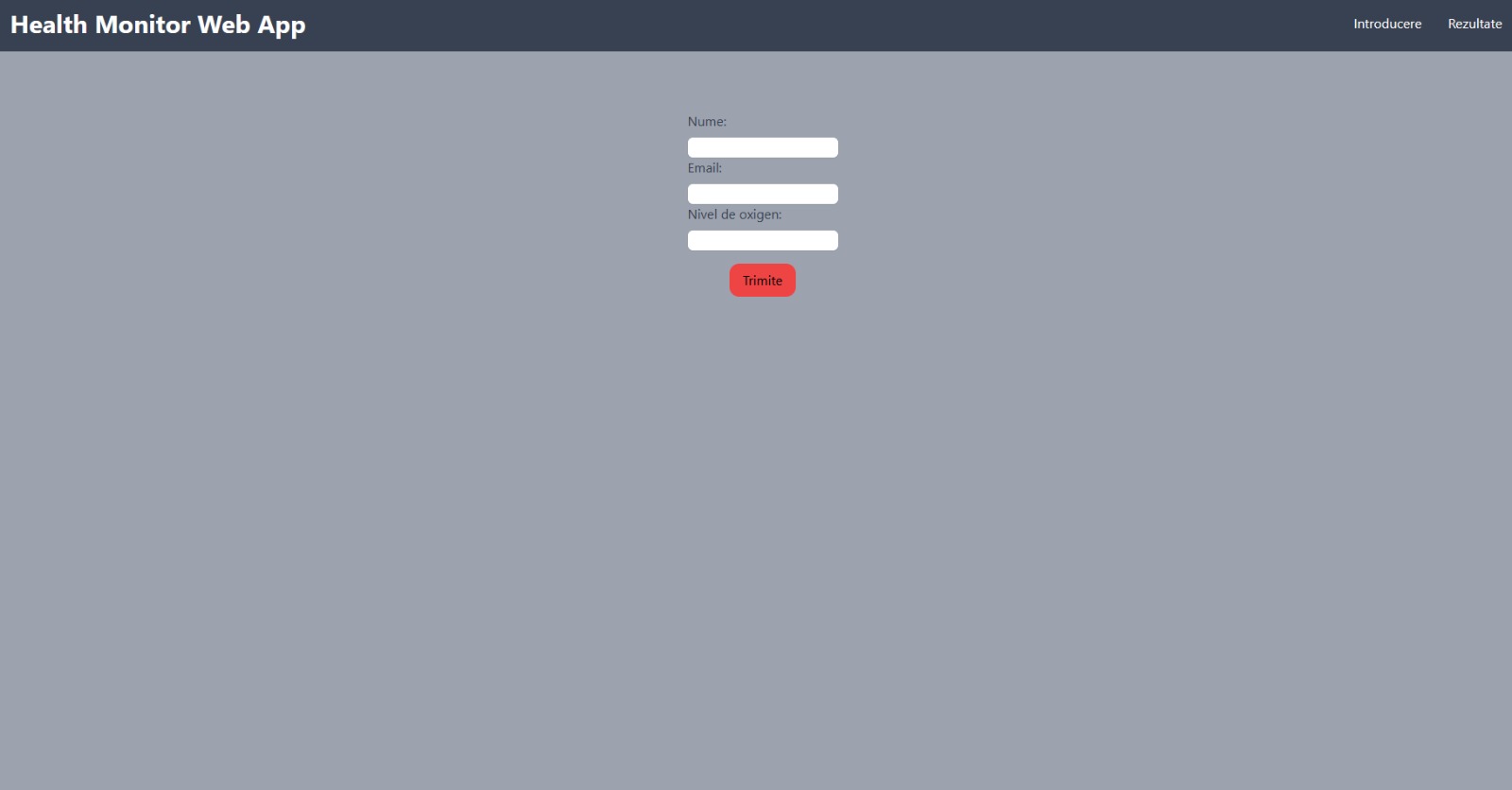
Cu ajutorul librariei React.js, am putut creea site-ul web al acestui proiect prin care putem cauta rezultatele pacientilor dupa numele acestora ,pentru a vizualiza nivelul de oxiegen al acestora si data cand a fost inregistrat rezultatul.

Figura 9.Interfata Web creeata cu React.js

Am ales React.js pentru a creea interfata web al acestui site deoarece, acesta utilizeaza un model de actualizare optimizat numit Virtual DOM, care ofera o viteza si performanta actualizarilor de interfata a utilizatorilor fata de alte framework-uri. Un alt avantaj al lui React.js sunt componentele reutilizabile, lucru care face ca dezvoltarea si intretinerea aplicatiilor sa fie mai usoara si mai rapida. Iar ultimul aspect pe care le-am luat in considerare atunci cand am ales React.js este intregrarea cu alte tehnologii. React.js poate fi integrat cu alte tehnologii JavaScript, chiar si non-JavaScript, cum ar fi TypeScript,Redux, iar daca proiectul este unul mai complex si implica o multitudine de tehnologii, aceasta librarie este o alegere buna.

În general, React.js este o alegere bună pentru dezvoltatorii care doresc să construiască interfețe utilizator rapide și performante, care sunt ușor de înțeles și de întreținut, și care pot fi integrate cu alte tehnologii pentru a construi aplicații complexe.

## Firebase

Firebase este o platformă de dezvoltare a aplicațiilor mobile și web, oferită de Google. Platforma oferă o gamă largă de servicii, inclusiv autentificare, bază de date în timp real, stocare de fișiere, notificări push și multe altele.

Firebase este cunoscută pentru ușurința cu care poate fi integrată în aplicații și pentru faptul că necesită puțină experiență în administrarea serverelor. Acest lucru face ca dezvoltarea aplicațiilor să fie mai rapidă și mai eficientă.

Firebase oferă API-uri pentru a accesa baza de date de la distanță. Acest lucru înseamnă că poți scrie cod care să acceseze baza de date Firebase, indiferent de locația ta fizică. Există API-uri disponibile pentru mai multe limbaje de programare, inclusiv JavaScript, Java, Objective-C și Swift.

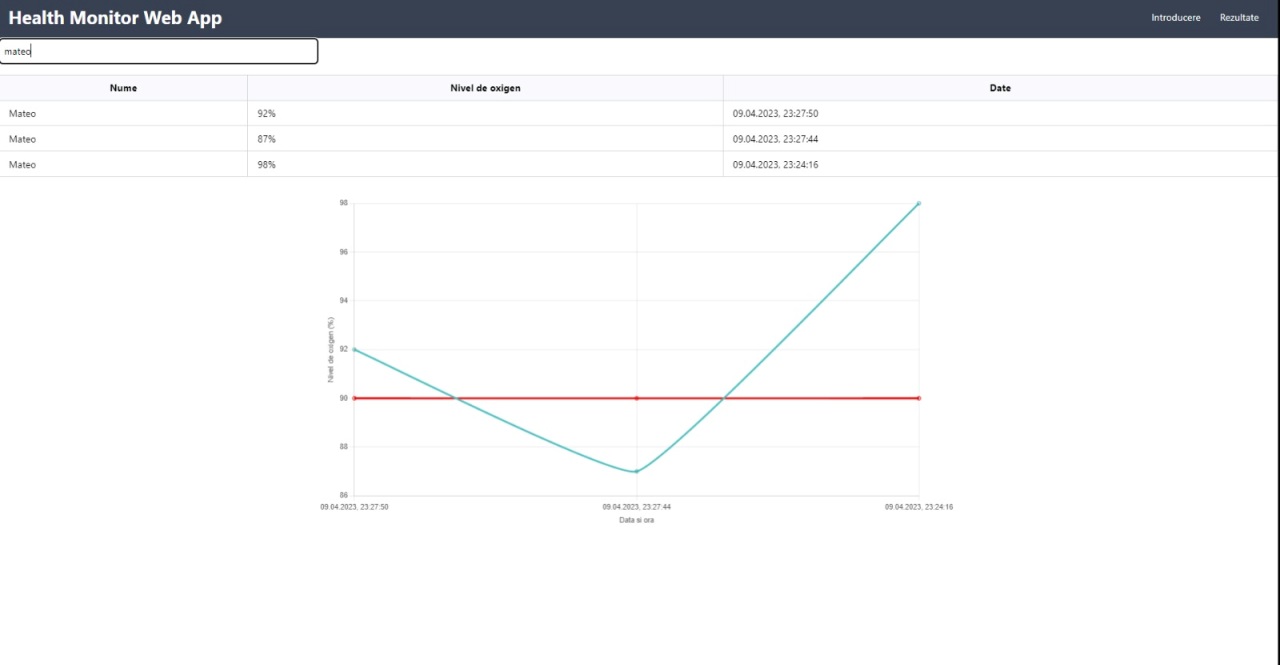


Figura 10.Interfata bazei de date

In momentul in care se introduce in bara se cautare a site-ului web, se vor afisa toti pacientii cu numele introdus, alaturi de nivelul de oxigen al acestora si data cand a fost inregistrata aceasta informatie.Pe baza acestor rezultate de va genera un grafic cu nivelul de oxigen si ora.

## Wifi.h

WiFi.h este o bibliotecă din Arduino IDE utilizată pentru a permite interacțiunea cu modulele WiFi, cum ar fi ESP32 cel folosit de catre noi la acest proiect. Această bibliotecă conține funcții pentru inițializarea și configurarea conexiunii WiFi, precum și pentru trimiterea și primirea de date prin această conexiune.

Biblioteca WiFi.h este foarte utilă pentru dezvoltarea de aplicații care necesită comunicare fără fir, cum ar fi aplicațiile IoT (Internet of Things) sau controlul dispozitivelor prin intermediul unei rețele WiFi. Aceasta permite ca dispozitivele să comunice între ele sau să acceseze resursele de pe internet prin intermediul unei rețele WiFi, permițând astfel crearea unor aplicații complexe și conectate.

Pentru a folosi biblioteca WiFi.h în Arduino IDE, trebuie să se includă biblioteca prin intermediul comenzii "#include <WiFi.h>" în codul sursă al proiectului.



Figura 11.Introducerea librariei Wifi.h in proiect

# Repository GitHub

## Generalitati

GitHub este o platformă de dezvoltare software bazată pe sistemul de control al versiunilor Git. Aceasta permite dezvoltatorilor să colaboreze la proiecte și să își împărtășească codul sursă.

Un repository Git este un spațiu de stocare în care se găsește întregul istoric al unui proiect de dezvoltare software. Acesta este format dintr-o serie de fișiere și directoare care conțin codul sursă, documentația și alte fișiere relevante pentru proiectul respectiv.

Repository-urile Git sunt gestionate prin intermediul comenzilor Git, care permit utilizatorilor să monitorizeze, să modifice și să distribuie conținutul dintr-un repository.

Un repository Git poate fi găzduit pe o varietate de platforme, inclusiv GitHub, Bitbucket și GitLab. Aceste platforme oferă un set de instrumente și funcționalități care permit dezvoltatorilor să colaboreze în mod eficient și să împărtășească codul sursă cu alți dezvoltatori sau cu comunități mai largi.

## Avantaje

Pentru a partaja proiectul si dezvoltarea progresiva a acestuia am ales sa folosim GitHub datorita mai multor avantaje pe care acesta il ofera.

1. GitHub ofera un sistem de control al versiunilor tuturor proiectelor, lucru care permite dezvoltatorilor implicati in proiect sa urmareasca fiecare modificare facuta asupra acestuia si in cazul in care sunt probleme dupa o anumita modificare, sa se poata reveni la versiunea anterioara al proiectului.
2. Comunitatea GitHub este o comunitate plina de dezvoltatori, lucru care face mai usoara munca, deoarece acestia pot sa primeasca feedback la codul lor sursa,precum si sa colaboreze intre ei pentru a duce un proiect la bun sfarsit.
3. Accesibilitatea este un alt punct forte al celor de la GitHub, deoarece oricine are acces la internet poate accesa repository-ul pe care il doresta.
4. Majoritate dezvoltatorilor folosesc GitHub pentru a creea un portofoliu unde pot prezenta toate proiectele la care au lucrat in colaborare cu alti dezvoltatori sau chiar singur, adaugand si codul sursa si contributiile aduce la proiect.

# Testare

# Management

## Importanta Management-ului in realizarea unui proiect

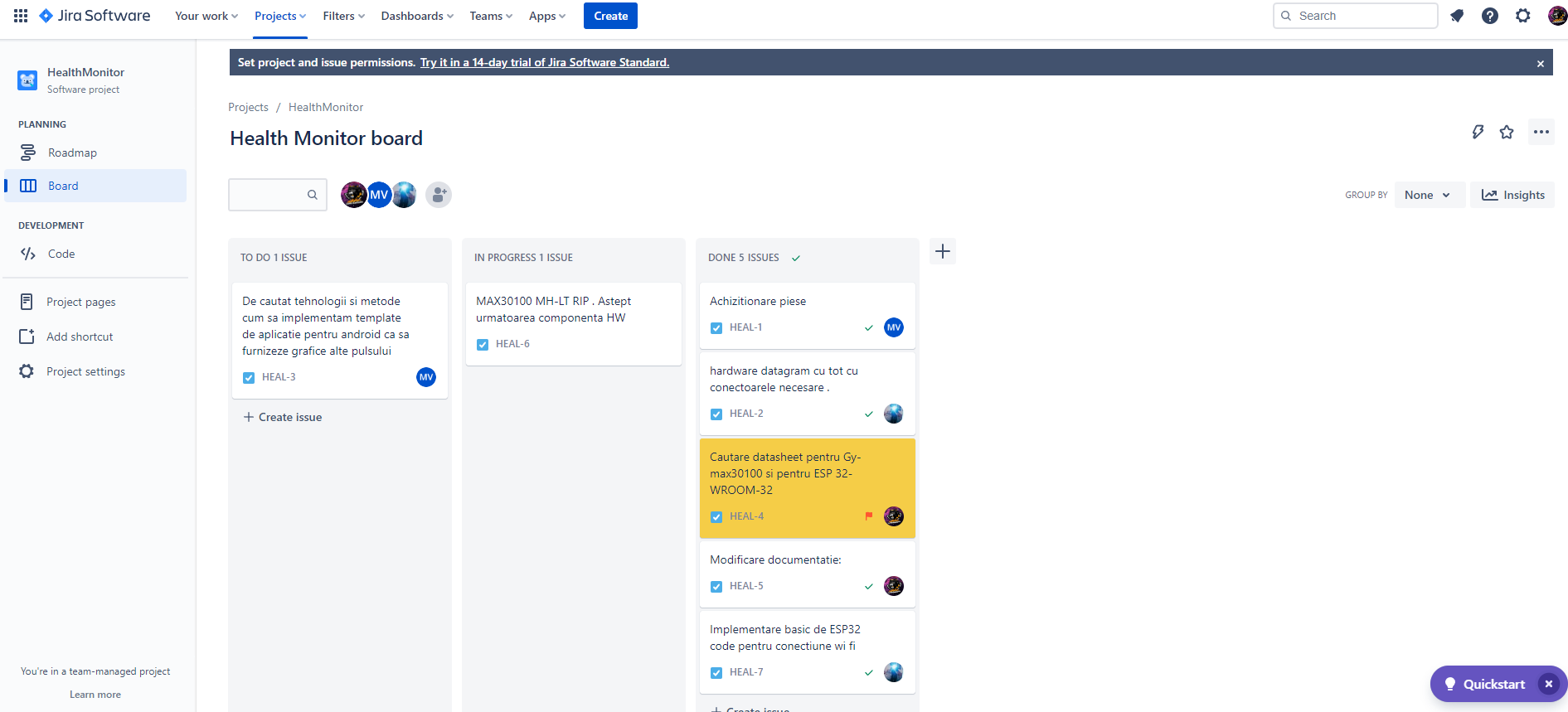
Managementul unui proiect este esențial pentru a asigura că acesta este finalizat în timp util, cu resursele și costurile alocate. Un bun management de proiect implică stabilirea unui plan detaliat pentru proiect, care să includă obiectivele și scopul, resursele necesare, termene limită și un plan de acțiune pentru a ajunge la aceste obiective. Managementul eficient al proiectului implică, de asemenea, monitorizarea și urmărirea progresului, identificarea și abordarea potențialelor probleme și riscuri, și comunicarea clară și eficientă cu membrii echipei și cu toate părțile interesate.

## Jira Management Software

Jira este o platforma de management pentru proiecte, utilizata pe scala larga la multe intreprinderi pentru a gestiona proiectele de tip software si nu numai.Jira permite utilizatorilor sa creeze sarcini de lucru, sa monitorizeze progresul si sa raporteze probleme in cazul in care apar in urma rezolvarii unei sarcini.

Jira are mai multe functii de baza precum:

1. Crearea de sarcini si probleme, unde utilizatorii pot adauga saricni, cerinte pentru proiect, inclusiv informatii detaliate asupra sarcinii, cat si termene limita si prioritati.
2. Administratorii proiectului pot atribui utilizatorilor sarcini catre membrii echipei.
3. Este posibila monotorizarea progresului sarcinilor si problemelor cu un sistem integrat de rapoarte si grafice.
4. Cea mai importanta functie de baza a programului Jira este colaborarea si comunicarea untilizatorior cu membrii echipei prin intermediul comentariilor si actualizarilor.



*Figura 12. Sarcinile alocate pe Jira*

In imaginea de mai sus se poate oberva cateva din sarcinile atribuite fiecarui utilizator din cadrul acestui proiect, stadiul in care se afla sarcina si importanta acestora.

# Lectii invatate

Pe parcursul dezvoltarii proiectului Health Monitor s-au folosit o multitudine de tehnologii, care au dus la acumularea unor noi informatii.

1. Conectarea si implementarea codului modulului ESP32 in Arduino IDE.
2. Realizarea conexiunii modului ESP32 la Wifi.
3. Colectarea datelor de la PulsOximetru si transmiterea acestora catre o baza de date.
4. Crearea interfetei pentru adaugarea si vizualizarea rezultatelor fiecarui client.
5. Intelegerea importantei bunei organizari in cadrul unui proiect unde participa mai multe persoane, fiecare avand un rol diferit cu ajutorul platformei Jira .
6. Acomodarea utilizarii site-ului GitHub pentru partajarea informatiilor si vizualizare noilor modificari facute de catre ceilalti utilizatori.

# Concluzii

Consideram ca, proiectul “Health Monitor” are aplicabilitate mare in viata moderna, fiind o aplicatie cu avantaje in realizarea unei vieti mai bune pentru persoanele care sufera de probleme respiratorii sau cardiace.

Dupa realizarea acestui proiect, putem asigura un trai mai bun persoanelor bolnave si o siguranta sporita asupra traiului, fara a traii cu frica, ca in orice moment poti avea o scadere sau crestere drastica a pulsului cu ajutorul graficelor si nimeni nu stie despre situatia in care te afli. Daca ai asupra ta acest dispozitiv, ai siguranta ca esti monitorizat permanent de catre un doctor specialist si in orice moment daca ai o problema.

In concluzie, aceast proiect aduce beneficii inspre un trai mai bun si poate fi dezvoltata intr-o aplicatie mai complexa ce poate si instiinta cel mai apropiat doctor specialist de catre locatia utilizatorului printr-un sms sau email, sau in cazuri extreme sa sune direct la 112.

# Appendix

# Codul sursa

## Conectiune Wifi

include <WiFi.h>,

const char\* ssid = "Galaxy A416D98";

const char\* password = "parolapas";

void setup(){

Serial.begin(115200);

delay(1000);

WiFi.mode(WIFI\_STA); //Optional

WiFi.begin(ssid, password);

Serial.println("\nConnecting");

while(WiFi.status() != WL\_CONNECTED){

Serial.print(".");

delay(100);

}

Serial.println("\nConnected to the WiFi network");

Serial.print("Local ESP32 IP: ");

Serial.println(WiFi.localIP());

}

void loop(){},

## Header.jsx

import React from 'react',

import { Link } from "react-router-dom";

function Header() {

return (

<div className='flex bg-gray-700 w-full justify-between items-center p-4 text-white'>

<div>

<h1 className='text-3xl font-bold'>Proiect</h1>

</div>

<ul className='flex'>

<li className='px-4 cursor-pointer capitalize hover:scale-105 duration-200'>

<Link to="/">Introducere</Link>

</li>

<li className='px-4 cursor-pointer capitalize hover:scale-105 duration-200'>

<Link to="/rezultate">Rezultate</Link>

</li>

</ul>

</div>

)

}

export default Header

## Introducere.jsx

import React, { useState } from 'react',

import { addDoc, collection, serverTimestamp } from 'firebase/firestore'

import { db } from '../firebase'

function Introducere() {

const [name, setName] = useState('');

const [email, setEmail] = useState('');

const [message, setMessage] = useState(null);

const handleSubmit = async (e) => {

e.preventDefault();

const messageDB = {

text: message,

name: name,

createdAt: serverTimestamp(),

}

await addDoc(collection(db, 'users'),

messageDB

)

setName('');

setEmail('');

setMessage(null);

};

return (

<div className='w-full min-h-screen bg-gray-400 p-12'>

<form onSubmit={handleSubmit} className='flex flex-col p-6 items-center justify-center'>

<label className="block">

<span className="text-gray-700">Nume:</span>

<input type="text" value={name} onChange={(e) => setName(e.target.value)} className="mt-2 block w-full rounded-md border-gray-300 shadow-sm focus:border-indigo-500 focus:ring focus:ring-indigo-200 focus:ring-opacity-50" />

</label>

<label className="block">

<span className="text-gray-700">Email:</span>

<input type="email" value={email} onChange={(e) => setEmail(e.target.value)} className="mt-2 block w-full rounded-md border-gray-300 shadow-sm focus:border-indigo-500 focus:ring focus:ring-indigo-200 focus:ring-opacity-50" />

</label>

<label className="block">

<span className="text-gray-700">Nivel de oxigen:</span>

<input type="number" value={message} onChange={(e) => setMessage(e.target.value)} className="mt-2 block w-full rounded-md border-gray-300 shadow-sm focus:border-indigo-500 focus:ring focus:ring-indigo-200 focus:ring-opacity-50" />

</label>

<button className='mt-4 px-4 py-2 bg-red-500 border-black rounded-xl' type="submit">Trimite</button>

</form>

</div>

);

}

export default Introducere

## Rezultate.jsx

import React, { useState, useEffect } from 'react';,

import { db } from '../firebase';

import { collection, onSnapshot, orderBy, query, where } from 'firebase/firestore';

import { Line } from 'react-chartjs-2';

import { Chart, CategoryScale, LinearScale, LineController, PointElement, LineElement } from 'chart.js';

Chart.register(CategoryScale, LinearScale, LineController, PointElement, LineElement);

function Rezultate() {

// Declare state variables for search query and search results

const [searchName, setSearchName] = useState('');

const [text, setText] = useState([]);

const [loading, setLoading] = useState(false);

// Define chart state

const [chartData, setChartData] = useState({});

// Define a function to handle search input changes

const handleSearchChange = (e) => {

setSearchName(e.target.value);

};

// Use useEffect to fetch search results from the database

useEffect(() => {

// Set loading state to true while fetching data

setLoading(true);

// Define a query to search for messages that match the search query

const q = query(

collection(db, 'users'),

where('name', '==', searchName),

orderBy('createdAt', 'desc')

);

// Subscribe to the query and update state when new data is received

const unsubscribe = onSnapshot(q, (querySnapshot) => {

const messages = [];

querySnapshot.forEach((doc) => {

messages.push({ id: doc.id, ...doc.data() });

});

setText(messages);

setLoading(false);

});

// Unsubscribe from the query when the component unmounts

return () => unsubscribe();

}, [searchName]);

// Use useEffect to update the chart data when the search results change

useEffect(() => {

// Define arrays to hold chart data

const labels = [];

const data = [];

// Loop through search results and add data to chart arrays

text.forEach((message) => {

labels.push(new Date(message.createdAt.toDate()).toLocaleString());

data.push(message.text);

});

// Define chart data

const chartData = {

labels: text.map((message) => new Date(message.createdAt.toDate()).toLocaleString()),

datasets: [

{

label: 'Nivel de oxigen',

data: text.map((message) => message.text),

fill: false,

borderColor: 'rgb(75, 192, 192)',

tension: 0.1

},

{

type: 'line',

label: 'Limita de siguranta',

data: Array(text.length).fill(90),

fill: false,

borderColor: 'red',

tension: 0.1

}

]

};

// Update chart data state

setChartData(chartData);

}, [text]);

const chartOptions = {

responsive: true,

scales: {

x: {

title: {

display: true,

text: 'Data si ora'

}

},

y: {

title: {

display: true,

text: 'Nivel de oxigen (%)'

}

}

},

plugins: {

title: {

display: true,

text: 'Nivel de oxigen in timp'

}

}

};

return (

<div className='flex flex-col'>

<input type="text" value={searchName} onChange={handleSearchChange} placeholder="Search by name" className="w-1/4 border border-gray-400 rounded-md p-2 mb-4" />

{loading && <div>Loading...</div>}

{!loading && text.length > 0 && (

<div className="">

<table className="table-auto border-collapse w-full">

<thead className="bg-gray-50">

<tr>

<th className="border border-gray-300 px-4 py-2">Nume</th>

<th className="border border-gray-300 px-4 py-2">Nivel de oxigen</th>

<th className="border border-gray-300 px-4 py-2">Date</th>

</tr>

</thead>

<tbody>

{text.map((message) => (

<tr key={message.id}>

<td className="border border-gray-300 px-4 py-2 capitalize">{message.name}</td>

<td className="border border-gray-300 px-4 py-2">{message.text}%</td>

<td className="border border-gray-300 px-4 py-2">{new Date(message.createdAt.toDate()).toLocaleString()}</td>

</tr>

))}

</tbody>

</table>

<div className='my-8 w-full md:w-2/3 lg:w-3/4 xl:w-1/2 mx-auto min-h-1/2'>

<Line data={chartData} options={chartOptions} />

</div>

</div>

)}

</div>

);

}

export default Rezultate;

# Firmware

# Bibliografie

[1] Imagini modul senzor puls cariac MAX30100 <https://www.emag.ro/modul-senzor-de-frecventa-cardiaca-puls-oximetru-max30100-ai465-s281/pd/DPRL27MBM/?X-Search-Id=248df114786a700de094&X-Product-Id=64974538&X-Search-Page=1&X-Search-Position=0&X-Section=search&X-MB=0&X-Search-Action=view>

[2] Imagine pini ESP32 [pinout.png (1617×548) (etechnophiles.com)](https://www.etechnophiles.com/wp-content/uploads/2021/02/pinout.png)

[3] Poza Breadboard 400 <https://cleste.ro/breadboard-400-puncte.html?utm_medium=GoogleAds&utm_campaign=&utm_source=&gclid=CjwKCAjw5pShBhB_EiwAvmnNV_TbNbWSr8a_QALbvOvl6X9S5fpPKKCiYXIa_sodLCfhht-mi8Lm4RoCXNUQAvD_BwE>

[4] Imagine Arduino Uno R3 <https://www.conexelectronic.ro/en/computer-mono-placa/4671-ARDUINO-UNO-R3-ORIGINAL.html>

[5]React.js <https://blog.hubspot.com/website/react-js>

[6]ArduinoIDE <https://docs.arduino.cc/learn/starting-guide/the-arduino-software-ide>

[7]Firebase https://fotc.com/ro/blog/firebase/?gclid=CjwKCAjwitShBhA6EiwAq3RqAzCYgVSzoyzV6tSsILg3FHZxF9VKuRePEoB5Vd2k7GbkChwiSDbJmRoCby8QAvD\_BwE