# USING THE ESP8266 MODULE TO TRANSMIT DATA TO A MOBILE PHONE OF THE VITAL PARAMETERS OF THE HUMAN BODY

**Lecturer PhD. Eng., Marian IONESCU, "Constantin Brâncuși" University from Tîrgu-Jiu, ROMANIA, i_marian_19@yahoo.com**

**ABSTRACT:** *This paper presents a system for measuring and monitoring blood oxygen levels ($SpO_2$) and heart rate (BPM) using the MAX30102 sensor. The transmission and monitoring of these vital parameters of the human body is carried out via a mobile phone using an Arduino Nano motherboard and the ESP8266 microcontroller within the system. Recent studies confirm the spectacular evolution of portable biomedical sensors, integrated into devices such as bracelets, watches or patches. The hardware integration of the MAX30102 sensor within the monitoring system was achieved using the standard $I^2C$ interface, which allows efficient communication with the ESP8266 microcontroller..*

**KEYWORDS:** blood oxygen level ($SpO_2$) heart rate (BPM), MAX30102 sensor, ESP8266 module, parameter monitoring, measurement and monitoring system, Arduino Nano

## 1. INTRODUCTION

In the context of rapid technological evolution, wearable biomedical devices have become essential for monitoring personal health. The increasing incidence of respiratory and cardiovascular diseases, along with the increasing desire for autonomous monitoring of vital parameters, has generated a growing demand for affordable, accurate and easy-to-use systems [1]. Recently, interest in wearable devices such as pulse oximeters has increased considerably, as home monitoring of blood oxygen saturation ($SpO_2$) has become increasingly important. This parameter is essential for assessing health status in situations such as respiratory infections, chronic diseases or in the context of pandemics. According to an article published by Verywell Health, tracking $SpO_2$ levels is particularly valuable in monitoring the progress of patients with lung or heart conditions, as it allows for early identification of reduced oxygenation and can guide decisions regarding the need for medical intervention [2].

## 2. PROBLEM FORMULATION

Oxygen is vital for the functioning of the body, being transported to the cells through the blood, with the help of hemoglobin. An essential indicator of the efficiency of this process is the oxygen saturation in the peripheral blood, called $SpO_2$, which expresses the percentage of hemoglobin loaded with oxygen compared to the total hemoglobin. In a healthy person at rest, normal $SpO_2$ values are between 95% and 100%.

Simultaneous monitoring of $SpO_2$ and heart rate (BPM) is extremely useful, as it provides an integrated perspective on how the heart and lungs collaborate to ensure proper oxygenation of the tissues. Technological advances have made possible the emergence of compact and affordable devices capable of measuring these parameters in real time, without punctures and completely non-invasive [3].

Currently, these technologies have gone beyond the hospital setting and are found in smart watches, fitness bracelets and other accessible solutions that facilitate continuous monitoring and prevention, directly from home, in a simple and practical way.**3. STRUCTURE OF THE MEASUREMENT SYSTEM OF VITAL PARAMETERS $SpO_2$ and BPM**

**MAX30102 module description**

To measure blood oxygen saturation ($SpO_2$) and heart rate (BPM), the MAX30102 sensor, an improved and optimized version compared to previous models [4], was used in the measurement system. This sensor, manufactured by Maxim Integrated, is specially designed for portable biomedical applications, featuring high accuracy and easy integration with modern microcontrollers.

The MAX30102 sensor incorporates two light sources – a red LED and an infrared LED – along with a highly sensitive photodetector, an amplifier and a high-performance analog-to-digital converter. It allows for precise measurements based on the principle of photoplethysmography (PPG), i.e. by analyzing how the light emitted by the LEDs is absorbed and reflected by the blood flow in the skin capillaries [5].

Compared to previous versions, the MAX30102 offers lower power consumption, better protection against ambient light and a higher quality signal, making it more suitable for use in portable battery-powered devices. It also has a wider dynamic range and can perform more stable measurements even under difficult conditions, such as involuntary user movements.

The hardware integration of the MAX30102 into the project was achieved using the standard I²C interface, which allows efficient communication with the ESP8266 microcontroller. In this way, the raw data captured by the sensor is taken and processed by the software developed on the NodeMCU board, resulting in useful values for $SpO_2$ and BPM.

An important aspect in using the MAX30102 sensor is optimizing its positioning in contact with the skin, to ensure a correct and constant measurement. It is also necessary to filter the signal through appropriate software algorithms, to reduce the effects of noise and obtain reliable readings. The MAX30102 sensor structure is shown in Figure 1 below.

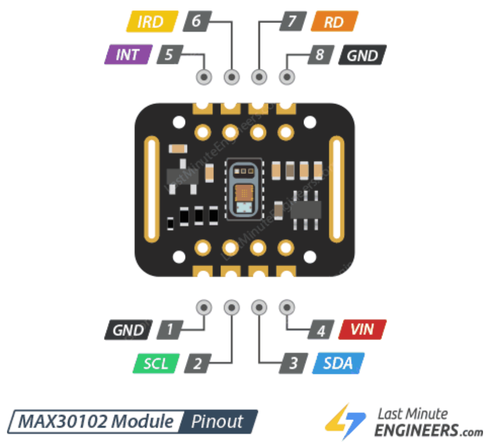


Fig. 1. MAX30102 sensor structure [5]

## ESP8266 module description

The ESP8266 is a microcontroller with integrated Wi-Fi capabilities that has become extremely popular in IoT (Internet of Things) projects due to its combination of performance, small size, optimal power consumption, and low price. Originally released by Espressif Systems, this chip has revolutionized the way embedded devices can communicate with local networks or the internet, without requiring additional external modules [6].

Essentially, the ESP8266 is a 32-bit microcontroller based on RISC architecture, running at a frequency of up to 80 MHz or 160 MHz, depending on the configuration. It has sufficient internal flash memory and SRAM for medium-sized embedded applications, being able to run code written in languages such as C or Arduino C++. One of the biggest advantages of this module is that it can function both as a Wi-Fi client device and as an access point, thus providing flexibility in the development of connected applications.

In this work, the ESP8266 was chosen as the main platform for controlling and processing the data provided by the max30102 sensor. This microcontroller not only reads the information regarding heart rate and oxygen saturation, but also processes and transmits it further to a server, thus providing the possibility of remote monitoring in real time. Communication with the sensor is done via the I²C protocol, and the data is processed locally to filter noise or to establish meaningful average values.

At the same time, there are numerous resources and code examples available in online communities, which reduces development time and the risk of errors. The choice of the ESP8266 microcontroller was dictated both by the technical requirements of the application – Wi-Fi connectivity, local processing, I²C integration – and to build a simple, efficient and scalable platform. By using it, a solid basis is provided for expanding functionalities, such as displaying values in a web interface or automatically transmitting data to external applications. In addition to its hardware features, the ESP8266 also stands out for its software versatility. It is compatible with multiple development environments, not only Arduino IDE, but also platforms such as PlatformIO, MicroPython or even advanced firmware such as ESPHome, frequently used in automation ecosystems (for example, Home Assistant). This flexibility opens up prospects for further expansion of the project, including more advanced data processing functionalities or integration with other IoT platforms. By combining these features — advanced wireless connectivity, extensive software support, optimized power consumption, and low cost — the ESP8266 microcontroller has established itself as one of the most popular solutions for modern embedded projects. Its choice provides not only the necessary technical support, but also the flexibility to develop additional functionalities in the future, such as cloud monitoring or integration into more complex IoT networks [6]. The internal structure of the ESP8266 module is shown in Figure 2 below.

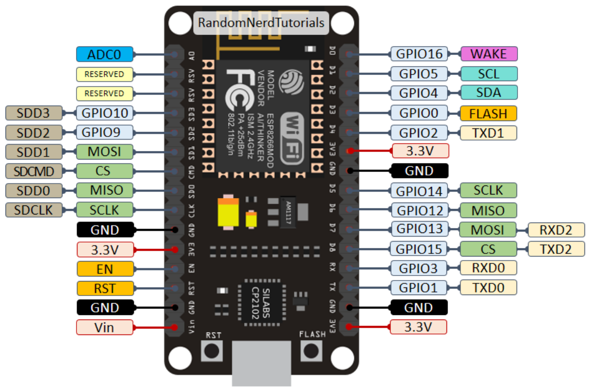
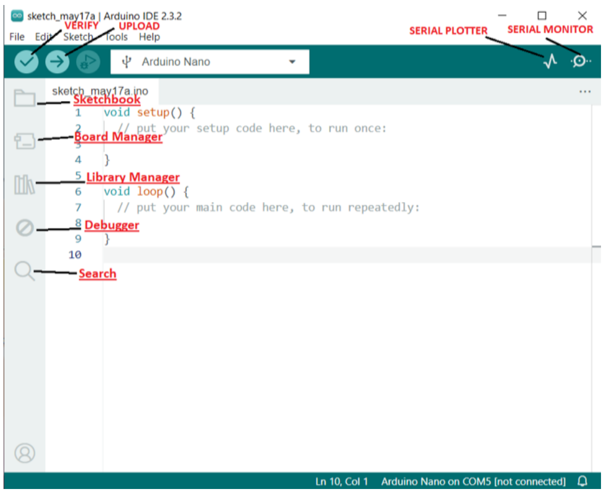


Fig. 2. Pinout NodeMCU - ESP8266 [6]

**Description of the Arduino IDE development environment**

The Arduino Integrated Development Environment (IDE) shown in Figure 3 below is an open-source software for developing applications for Arduino and compatible microcontroller boards [7] [8]. Arduino IDE provides an intuitive, cross-platform interface (available for Windows, MacOS, and Linux) that allows you to write, compile, and upload code directly to the hardware board. Supporting the C and C++ programming languages, Arduino IDE simplifies the embedded development process by providing a complete set of project management tools and a vast ecosystem of libraries and extensions [8]. Over time, many advanced features have been added to the IDE, such as board package management, Arduino Cloud integration, library manager, and debugging tools.



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Fig. 3. Interfața Arduino IDE.

## 4. SpO₂ and BPM PARAMETER MEASUREMENT SYSTEM AND DATA TRANSMISSION

The complete electrical schematic of the assembly is shown in Figure 4 below. It illustrates the connections between the LiPo battery, the TP4056 charging module, the NodeMCU ESP8266 development board, and the MAX30102 sensor.

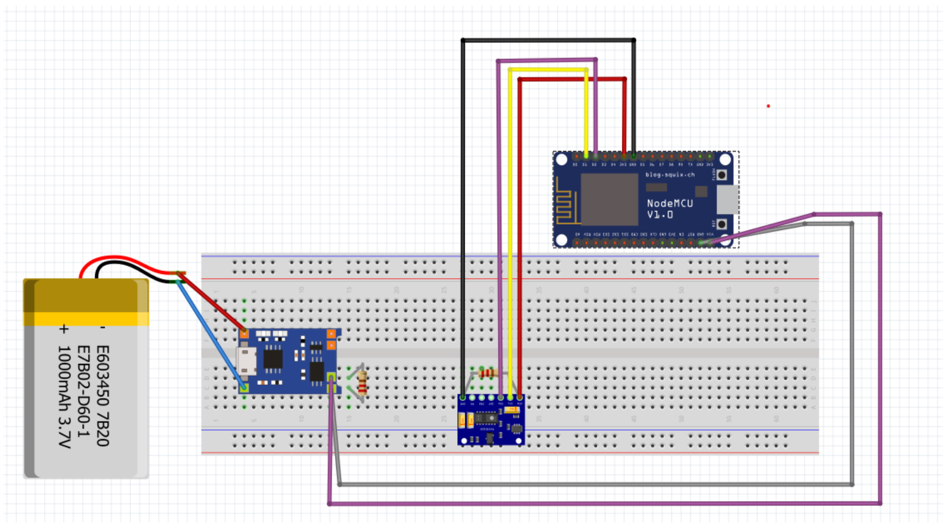


Fig.4 Diagram of the vital parameters measurement system and data transmission [6]

It can be seen the simple and efficient integration mode, which ensures stable system power supply and correct communication between components. This hardware configuration allowed the successful testing of the entire system, both in USB-powered mode for programming and in fully portable, battery-powered mode, while maintaining stable and reliable operation. The source code implementation is the essential component that links the hardware part of the device with the software part, ensuring the integrated operation of the system. The code was developed using the Arduino IDE environment, due to its excellent compatibility with the ESP8266 platform and the extensive support of available libraries. To implement these functionalities, several open-source libraries were used, which contributed to simplifying and streamlining code development:

- ESP8266WiFi.h — for managing the connection to the Wi-Fi network;
- ESP8266HTTPClient.h — for making HTTP requests to the Firebase server;
- MAX30105.h — SparkFun library for MAX3010x family sensors, which allows configuring and reading raw values from the sensor;
- heartRate.h — the heart rate calculation module, included in the SparkFun library. **Transmitting data to the Firebase server**

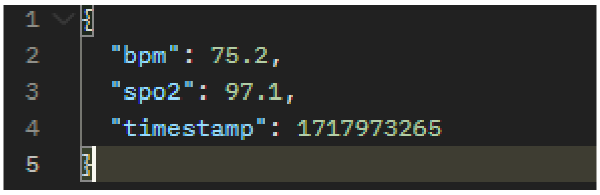
The transmission of measured parameters ($SpO_2$ and BPM) to a cloud server has been seen as an essential component of the software architecture. This offers the possibility not only to visualize the data in real time, but also to store, analyze and correlate them over time, thus creating a valuable history that can be used by both the user and the medical staff. [9]. In the field of biomedical applications, remote data transmission offers multiple advantages. First, it allows continuous monitoring of the user's health status, without the need for physical presence in a medical office. Second, it facilitates rapid intervention in case of anomaly, since the data can be analyzed in real time by specialists or can trigger automatic alerts if critical thresholds are exceeded. Also, cloud storage allows the creation of a detailed history of physiological parameters, which can be extremely useful for long-term analysis of the health status. This history can help diagnose conditions more accurately, track the effectiveness of a treatment, or identify subtle trends that might be missed by point-in-time measurements [9].

**Description of software architecture and cloud platform**

To implement the data transmission component, an architecture based on the Firebase Realtime Database platform [9], part of the Google Cloud ecosystem, was chosen. From an architectural point of view, the data flow is as follows: the NodeMCU ESP8266 board [10] collects data from the MAX30102 sensor, processes it, and transmits it as a JSON payload to the REST endpoint of the Firebase database. The connection is made via the local Wi-Fi network to which the device is connected.

**Data structure and benefits of using JSON format**

Using JSON for data transmission offers multiple advantages. First, it is a standard format, widely used in web and cloud applications, which facilitates the subsequent integration of data into other systems. Second, it is an easy-to-generate and process format, even on embedded devices with limited resources [11]. The payload structure used in the project is simple and clear, containing exactly the necessary information. Figure 5 below shows the data sent by the MAX30102 sensor to Firebase.



```
{
  "bpm": 75.2,
  "spo2": 97.1,
  "timestamp": 1717973265
}
```

Fig.5 Transmitting data sent by the MAX30102 sensor to Firebase

This structure allows not only to display values in real time, but also to build a detailed history of the monitored parameters. The current architecture of the transmission component is scalable and can be easily extended. As the architecture of the monitoring system evolves, new data types (e.g., body temperature, physical activity level) can be added to the JSON payload. Support for more power-efficient and reliable protocols, such as MQTT, which is specifically designed for IoT communications, can also be implemented [12]. By integrating a web interface or a mobile application, the user can have real-time access to their own data, with advanced functionalities such as graphical visualization of the evolution over time, setting personalized alerts, or exporting data for medical analysis.

## 5. DEVELOPMENT OF THE WEB APPLICATION FOR MONITORING $SpO_2$ and BPM PARAMETERS

In recent years, with the accelerated advancement of IoT (Internet of Things) technologies and the growing interest in personal health monitoring solutions, user requirements have evolved significantly. Today, a portable biomedical device is no longer perceived as a simple device that collects local data, but as an integrated element in a connected ecosystem, capable of providing relevant information in real time and allowing continuous analysis of health parameters. Starting from this premise, the web component of the project was developed to significantly extend the capabilities of the hardware device. If the hardware part (composed of the MAX30102 sensor and the ESP8266 microcontroller) is responsible for measuring and transmitting data, the web application has the role of transforming this raw data into valuable and easy-to-interpret information for the end user.

The main purpose of the application is to provide an intuitive and accessible platform for real-time visualization of vital parameters, namely heart rate (BPM) and blood oxygen saturation ($SpO_2$). These data are continuously measured by the wearable device and transmitted, via a Wi-Fi connection, to the Firebase Realtime Database. From here, the web application retrieves the data in real time and displays it to the user in a clear and user-friendly visual way. In addition to real-time visualization, the application also provides a recent history of measurements, in the form of a table with the last 20 records. This functionality is particularly useful for tracking the short-term evolution of the parameters, allowing the identification of any significant variations or trends. Thus, the web application naturally completes the architecture of the built biomedical IoT system, implementing a complete data flow, namely:

- local collection of physiological data using the MAX30102 sensor;
- data processing and transmission by the ESP8266 microcontroller;
- data storage and synchronization in the cloud (Firebase Realtime Database);
- real-time visualization and short-term analysis through the React interface.

This architecture not only ensures easy access to personal health data, but also offers the possibility of universal access. The user can consult the application from anywhere, on any device connected to the Internet, be it a desktop computer, a laptop, a tablet or a mobile phone. Without the web application, the presented system would be limited to local monitoring, with direct interaction with the hardware. By including the web application, the data becomes:

●visible and interpretable directly by the user;

●stored in the cloud, allowing retrospective access to their history [13];

●available for eventual integration into wider health monitoring ecosystems (e.g. mobile applications, monitoring interfaces for doctors, research platforms).

In addition, the web application opens up clear perspectives for the evolution of the project. In addition to the functionalities already implemented (live visualization + recent history), some advanced functions can be integrated, such as:

●automatic alerting (via notifications or e-mail / SMS messages) in case of detection of critical parameter values;

●advanced graphical visualization (long-term trends, correlation of values);

●multi-user authentication, to personalize the experience and protect data confidentiality.

In conclusion, the developed web application is not just a visualization interface, but is a strategic component of the overall architecture, which transforms raw data into information with practical value for the user. This clearly demonstrates the application of the principles of a modern biomedical IoT system, in which data is collected, processed, transmitted, stored and visualized in a complete and efficient flow. Figure 6 below shows the information flow of data that is transmitted within the $SpO_2$ and BPM parameter monitoring system.

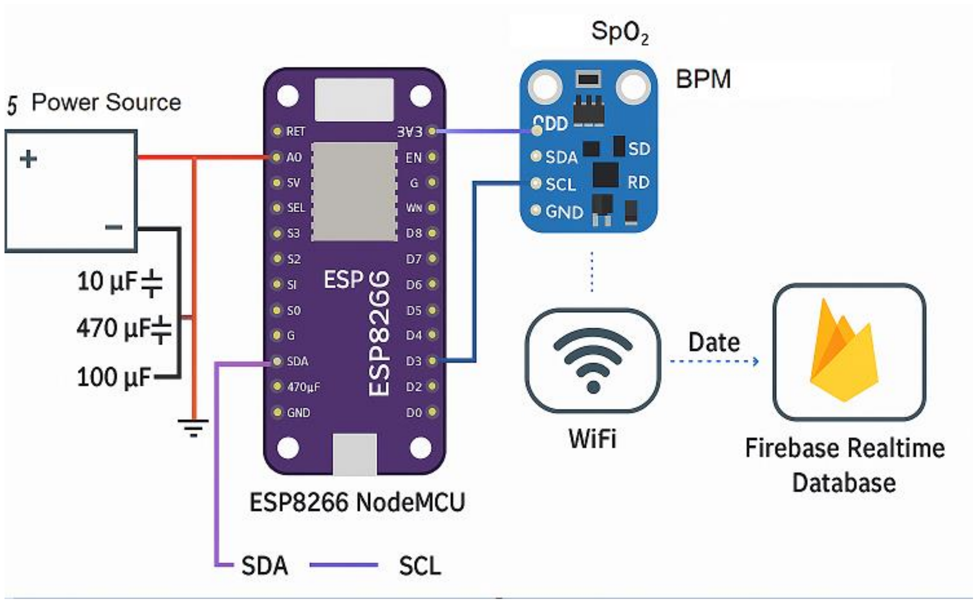
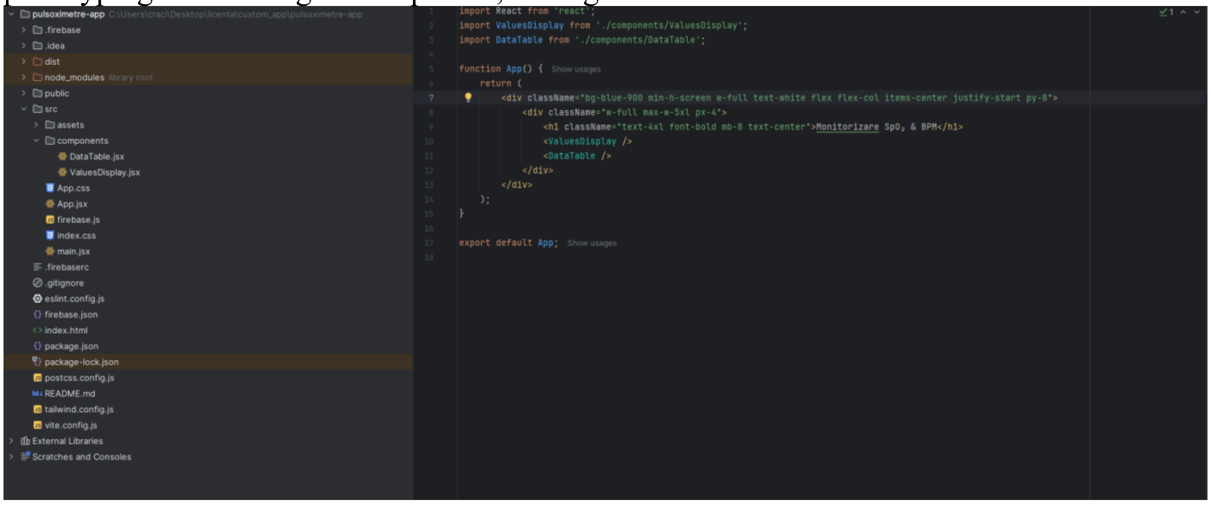


Fig. 6. The information flow of data transmitted within the $SpO_2$ and BPM monitoring system.

## Presentation of the development environment used

The development of the web application of this project was carried out in a programming environment, adapted to the modern requirements of front-end development with React and Firebase. The correct choice of the development environment was essential to ensure an efficient process of implementation, testing and publishing the application, as well as to guarantee a stable and high-performance interface for the end user[14]. The main IDE (Integrated Development Environment) used was WebStorm, a product developed by JetBrains, internationally recognized for its extensive capabilities in working with JavaScript, React, TypeScript, HTML and CSS. It offers a fluid and productivity-focused development experience. WebStorm enables efficient work with React projects through native support for JSX, static code analysis, and visual debugging. Within the monitoring system, it provided a major advantage in the prototyping and integration phases, being used in parallel with the Chrome browser for live testing, via the local development server [15]. Figure 7 below shows the WebStorm IDE graphical interface.



```
pulsoximetre-app C:\Users\craci\Desktop\licenta\custom_app\pulsoximetre-app
  .firebase
  .idea
  dist
  node_modules library root
  public
  src
    assets
    components
      DataTable.jsx
      ValuesDisplay.jsx
    App.css
    App.jsx
    firebase.js
    index.css
    main.jsx
  .firebaserc
  .gitignore
  eslint.config.js
  firebase.json
  index.html
  package.json
  package-lock.json
  postcss.config.js
  README.md
  tailwind.config.js
  vite.config.js
External Libraries
Scratches and Consoles
```

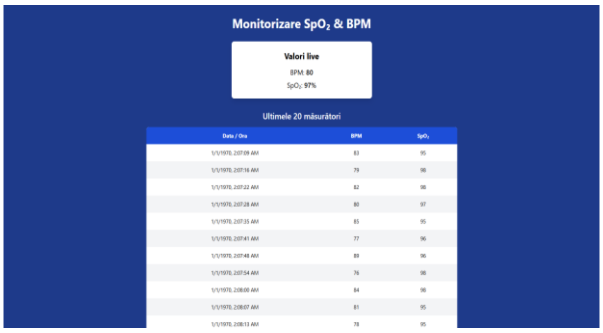
```jsx
import React from 'react';
import ValuesDisplay from './components/ValuesDisplay';
import DataTable from './components/DataTable';

function App() { Show usages
    return (
        <div className="bg-blue-900 min-h-screen w-full text-white flex flex-col items-center justify-start py-8">
            <div className="w-full max-w-5xl px-4">
                <h1 className="text-4xl font-bold mb-8 text-center">Monitorizare SpO₂ & BPM</h1>
                <ValuesDisplay />
                <DataTable />
            </div>
        </div>
    );
}

export default App; Show usages
```

Fig. 7. Interfață grafică IDE WebStorm

The JavaScript programming language provided the foundation of the application logic: it connected the visual display component with data obtained from Firebase, triggered automatic updates in the interface, and controlled the behavior of React components based on user interaction or changes in the database [16]. A key role in the data flow was played by JSON (JavaScript Object Notation), a standard data representation format widely used in web applications. Firebase Realtime Database natively uses JSON-like structures to store data, and React, through the Firebase SDK, provides easy-to-use methods for extracting, interpreting, and displaying this data. All values measured by the ESP8266 device heart rate (BPM), oxygen saturation ($SpO_2$), and the times associated with each measurement are transmitted and stored as JSON objects in Firebase. These are then retrieved by the web application and interpreted by JavaScript logic to be displayed in the interface as shown in Figure 8 below.



Monitorizare $SpO_2$ & BPM

Valori live

BPM: 80

$SpO_2$: 97%

Ultimele 20 măsurători

| Data / Ora | BPM | $SpO_2$ |
|---|---|---|
| 1/1/1970, 2:07:09 AM | 83 | 95 |
| 1/1/1970, 2:07:16 AM | 79 | 98 |
| 1/1/1970, 2:07:22 AM | 82 | 98 |
| 1/1/1970, 2:07:28 AM | 80 | 97 |
| 1/1/1970, 2:07:35 AM | 85 | 95 |
| 1/1/1970, 2:07:41 AM | 77 | 96 |
| 1/1/1970, 2:07:48 AM | 89 | 96 |
| 1/1/1970, 2:07:54 AM | 76 | 98 |
| 1/1/1970, 2:08:00 AM | 84 | 98 |
| 1/1/1970, 2:08:07 AM | 81 | 95 |
| 1/1/1970, 2:08:13 AM | 78 | 95 |

Fig. 8. Captură de ecran cu interfața aplicației, afișând valorile $SpO_2$ și BPM

The user interface is the visible and interactive component of a web application, playing a key role in facilitating communication between the user and the back-end IT system. In this project, the interface was designed with a focus on simplicity, clarity, and accessibility, in a way that allows users – regardless of their technical level – to easily interpret the biometric data provided by the MAX30102 sensor. In the screenshot showing the application interface, live values and the $SpO_2$ and BPM measurement table are displayed.

## 5. CONCLUSIONS

On the embedded software side, the ESP8266 microcontroller was programmed to capture biometric data, process it locally, and periodically transmit it to a Firebase database. This was done using the Arduino IDE and specialized libraries that allow communication via the I²C protocol and sending data via HTTP, over a secure Wi-Fi connection. The web interface was developed in React, using a basic system for displaying measured values and recent history in real time. TailwindCSS was integrated for modern and fast styling, and the Firebase Realtime Database allowed for efficient data synchronization between the measuring device and the web application. This work provides a clear example of integrating several technical fields – electronics, embedded programming, web development, and cloud computing – into a single functional product. Future development directions may include adding a dedicated mobile application, integrating with external services (email/SMS) for alerting, implementing user authentication, or extending functionalities with new types of sensors (temperature, EKG, etc.). In addition, the automated data analysis part can be explored, using artificial intelligence or simple machine learning models. Finally, this paper presents not only a practical application, but also a starting point for other innovative ideas in the field of digital health. The implementation and detailed documentation of all steps provide a solid basis for learning, replication or further development. The paper reflects a comprehensive, multidisciplinary and future-oriented approach, emphasizing efficiency, accessibility and practical value.

## REFERENCES

[1] Verywell Health. "Normal Pulse Oximetry Ranges and How to Take a Reading." Verywell Health (https://www.verywellhealth.com/pulse-oximetry-8708212)

[2] Vijenthira, O., Sahni, V., Juurlink, D., et al. (2022). Home pulse oximetry monitoring during the COVID-19 pandemic. NPJ Digital Medicine. (https://pmc.ncbi.nlm.nih.gov/articles/PMC10291857/)

[3] Beaney, T., Clarke, J., Alboksmaty, A., et al. (2022). The impact of remote home monitoring of people with COVID-19 using pulse oximetry: a national population and observational study. EClinicalMedicine. https://journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0310822

[4] Smith, J., & Lee, R. (2023). A Comprehensive Review on Wearable Health Monitoring Systems. Open Biomedical Engineering Journal, 15, 213–225. (https://openbiomedicalengineeringjournal.com/VOLUME/15/PAGE/213/)

[5] SparkFun Electronics. SparkFun MAX3010x Sensor Library. GitHub repository. (https://github.com/sparkfun/SparkFun_MAX3010x_Sensor_Library)

[6] Espressif Systems. (2016). ESP8266EX Datasheet. (https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf)

[7] Ilie Borcoși, Bogdan Sanda, THE DEVELOPING INTERACTIVE APPLICATIONS USING ARDUINO BOARD, Annals of the „Constantin Brancusi" University of Targu Jiu, Engineering Series, No. 1/2017, pp. 20-24, ISSN 1842-4856.

[8] Borcosi, I., The control of a SUMO robot, Analele UCB, Seria Inginerie, nr.4/2017, ISSN 1842-4856, pag. 140-144.

[9]Google Firebase. (2024). Firebase Realtime Database REST API documentation. https://firebase.google.com/docs/database/rest/start

[10]Espressif Systems. (2023). ESP8266 Arduino Core — HTTPClient Library Documentation. GitHub repository. https://github.com/esp8266/Arduino/tree/master/libraries/ESP8266HTTPClient

[11]ECMA International. (2017). The JSON Data Interchange Format (ECMA-404). https://www.ecma-international.org/publications-and-standards/standards/ecma-404/

[12]Nabto. (2021). IoT Data Transmission Best Practices.

https://www.nabto.com/iot-data-transmission-best-practices/
[13]Google Firebase. (2024). Firebase Realtime Database documentation. https://firebase.google.com/docs/database
[14] Adrian Runceanu, Mihaela-Ana Runceanu, “Challenges in teaching programming and algorithms”, INTED2016 Proceedings, 10th International Technology, Education and Development Conference Valencia, Spain. 7-9 March, 2016, ISBN: 978-84-608-5617-7 / ISSN: 2340-1079, Pages: 4120-4126
DOI: 10.21125/inted.2016.2003, WOS:000402738404019
[15]JetBrains. (2024). WebStorm: The Smartest JavaScript IDE. https://www.jetbrains.com/webstorm/
[16] Adrian Runceanu, “A horizontal fragmentation algorithm for distributed databases” 14th International Multidisciplinary Scientific GeoConference & EXPO Modern Management of Mine Producing, Geology and Environmental Protection, SGEM 2014,
DOI: 10.5593/SGEM2014/B21/S7.002
WOS:000371297900002
https://www.webofscience.com/wos/woscc/full-record/WOS:000371297900002