

IoT Project Document

Healthcare Advanced Protection System (HAPS)

Professor:

Dr. Somayeh Sobati Moghadam

T.A.:

ENG. Abolfazl Rajaiyan

Team Members:

Kiumars Javan, Sina Razaghi, Davood Hoseini

June 2022

Abstract

IoT is one of the popular and high-demand technologies and many companies of different sizes are working in this area. One of the interesting and helpful uses of IoT is in the Medical and Healthcare industry. There are many devices and gadgets with their pros & cons in the market for healthcare and patient monitoring, but most of them are not integrated and optimized for personal use. They cannot be used in organizational and large-scale scenarios, also those existing solutions are not available for domestic organizations and are not compatible with their needs. We present a great solution and integrates and complete platform for their need with complete personalization capability. Our platform makes the reception and management of patient so easy for treatment staff. They can monitor patient vital signs real-time on all of their devices though our mobile phone app, web application, desktop application etc. In our implementation's architecture all of the necessary needs of an IoT-based healthcare monitoring system are considered and tried to utilized best match technologies for them.

Table of Contents

Introduction	2
HAPS.....	2
Back-end	2
Smart Gadget.....	3
Mobile App	9
Web App.....	11
Other	13
Future Works	15

Introduction

Internet of Things (IoT) rises as a powerful domain where embedded devices and sensors can connect and exchange information over the Internet. High-quality health care helps prevent diseases and improve quality of life. Rising interest in body wearable sensors has recently emerged as powerful tools for healthcare applications and different devices are currently available commercially for different purposes including personal healthcare, activity awareness, and fitness [1]. But in this project, we are focused on organizations and hospitals' needs in improving the quality of healthcare services and patients' monitoring systems. In this project, we propose and implement the Healthcare Advanced Protection System (HAPS).

HAPS

HAPS is an integrated and smart patient monitoring platform that can be used in various type of hospitals and clinics with different medical services. Patients monitoring using latest cutting-edge IoT technologies today is one of the biggest interest of researchers, engineers and companies in medical field around the world. With this platform hospitals, doctors and nurses can always be aware of their under-supervision patient immune vital signs and keep them under control.

The HAPS is consisting of three main parts:

1. The Smart gadget which is physically attached to the patient and measure patient's body temperature, heartbeat rate, and pulse oximetry and send them to the main cloud server real-time.
2. An application that the treatment staff can interact with the system and monitor their patient using their mobile phone through the mobile application or with web application.
3. The core software or the back-end of the application which is responsible for communicating with gadgets and collect their transferred data, storing and presenting needed information by the applications.

Back-end

HAPS core software is a fast message bus for collecting data that is coming from patients' gadget and a powerful API for dealing with applications requests. In this project we used one of the best and powerful solutions for implementing a reliable back-end software, because the back-end is the back bone of the HAPS, so we are using .NET Core framework for this part.

Smart Gadget

For this project we created our very first own gadget and, in this prototype, there are three sensors for measuring vital signs and an Arduino Uno for managing sensors, keypad and the character LCD.

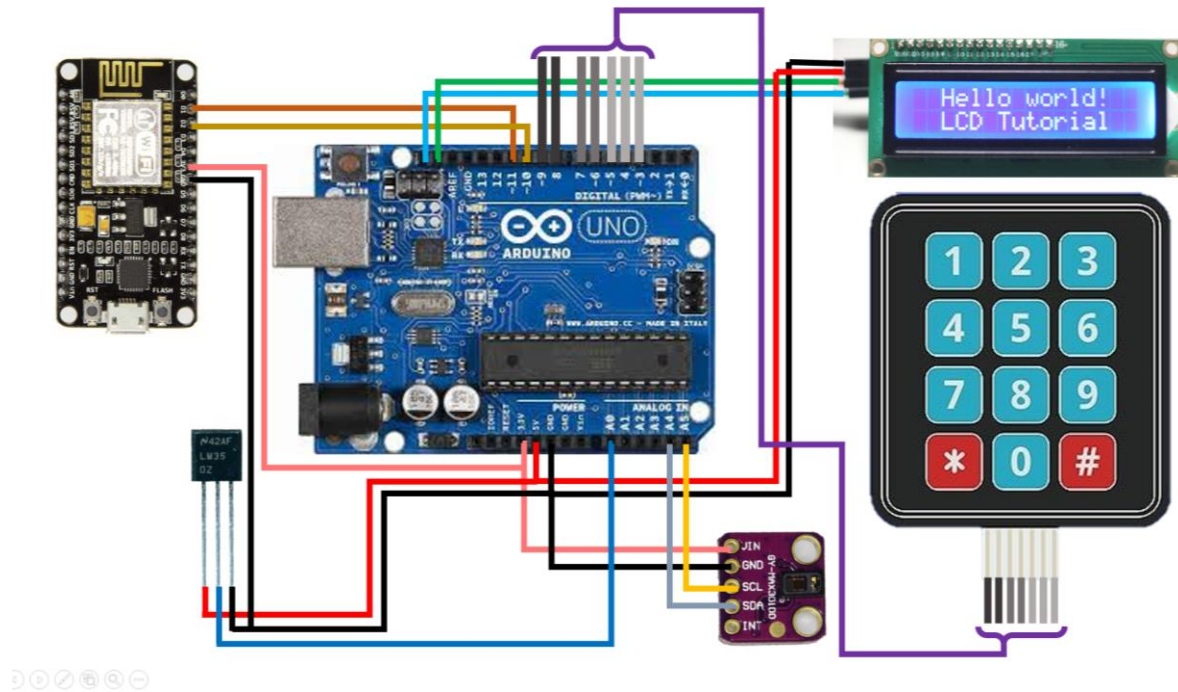
Connections

So according to the said parameters to make the first example of this device and connect it to the API, we use the following parts:

- For the first two cases of the Max 30100 sensor
- For body temperature from LM35
- 16*2 Character LCD i2c to display information
- 3 * 4 keypad to control different parts
- Arduino Uno R3 processing part
- NodeMCU esp8266 for wireless connection to server

In performing it, an attempt has been made to perform a certain amount of error test, so that one device does not need to receive information from several people (the device can be made smaller and more specific and cheaper). But since it is a prototype and has a research framework, we have tried to manually control and examine a series of features and applications.

For the prototype, no special power supply is used, and since the Arduino works with 5V voltage, it can be connected directly to a mobile charger adapter or a USB laptop. NodeMCU uses 3V voltage, which can be obtained from the Arduino itself (pin 3.3V). Now it's time for the connections section:



Picture 1

Keypad to 3 to 9 Arduino pins. Two VCC and GND LCD pins to 5V and GND Arduino pins and I2C pins to Arduino I2C pins.

The reason for using NodeMCU is very simple. A purchase error, which in itself posed a major challenge, to connect the Arduino to the NodeMCU serially via pins 10 and 11 of the Arduino to pins D1 and D2 of the NodeMCU. For the NodeMCU power supply, we use the 3.3V and GND pins of the Arduino.

Note: NodeMCU and Arduino must be shared power supply or NodeMCU must use Arduino itself due to serial connection.

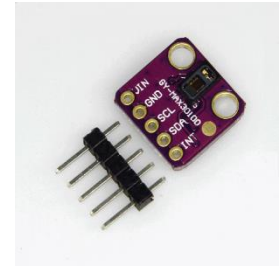
The MAX30100 is an integrated pulse oximetry and heartrate monitor sensor solution. It combines two LEDs, a photodetector, optimized optics, and low-noise analog signal processing to detect pulse oximetry and heart-rate signals. The MAX30100 operates from 1.8V and 3.3V power supplies and can be powered down through software with negligible standby current, permitting the power supply to remain connected at all times.



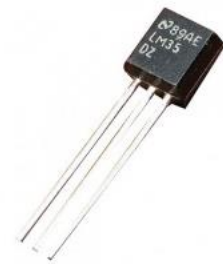
Applications

- Wearable Devices
- Fitness Assistant Devices
- Medical Monitoring Devices

Now it's time to connect the sensors to Arduino. The sensor that is going to be very annoying to start is the max30100 sensor. 4 of its pins should be connected, but the INT pin is not required.



The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracy of $\pm 1/4^\circ\text{C}$ at room temperature and $\pm 3/4^\circ\text{C}$ in full -55 to $+150^\circ\text{C}$ temperature range. Low cost is assured by trimming and calibration at the Wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies.



As it draws only $60\text{ }\mu\text{A}$ from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a -55° to $+150^\circ\text{C}$ temperature range, while the LM35C is rated for a -40° to $+110^\circ\text{C}$ range (-10° with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

Features:

- Calibrated directly in ° Celsius (Centigrade)
- Linear + $10.0\text{ mV}/^\circ\text{C}$ scale factor
- 0.5°C accuracy guaranteed (at $+25^\circ\text{C}$)
- Rated for full -55° to $+150^\circ\text{C}$ range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than $60\text{ }\mu\text{A}$ current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only $\pm 1/4^\circ\text{C}$ typical
- Low impedance output, $0.1\text{ }\Omega$ for 1 mA load

How the program code works?

Functional summary: The first important issue is to communicate wirelessly with the app, in this part we are connected to a local router modem by NodeMCU via SSID and password. Now that the internet connection is established, we can easily send the request to the API and receive the answer, and we can decode and use that JSON. The next step is to check the connection to the API, which returns the value 'haps_api' if the address

and request are correct. We can get user information from the API by receiving the user code from the user. Considering the user information, a new record can be recorded in the system using sensors.

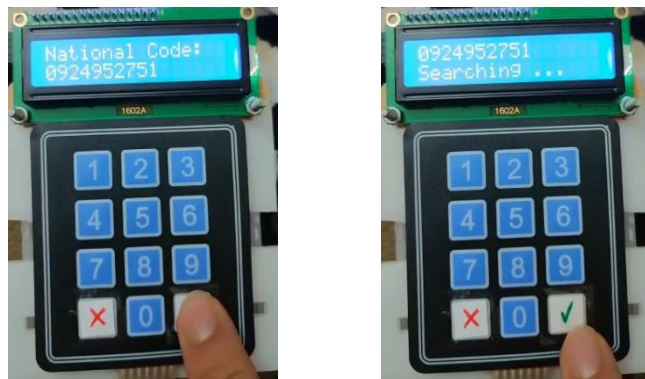
This device can be configured to be attached to a user code and used specifically for it. With this method, you no longer need a keyboard and you can also remove the Arduino.

The first method:

1. The device operates almost automatically and the user is not very familiar with how it works. So, what the user sees is simpler and more limited.



2. At the same time, it was connected to both the Internet and the API, and it was automatically transferred to the national code download stage.



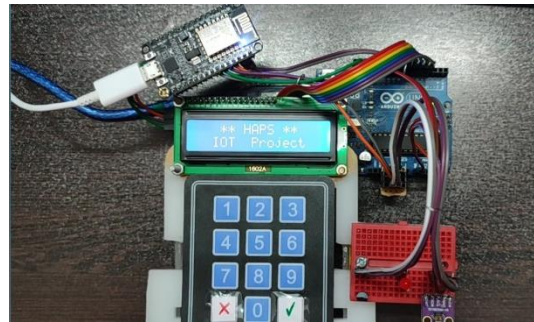
3. After confirming the code and finding the user, the sensors start working and the data is sent to the API.



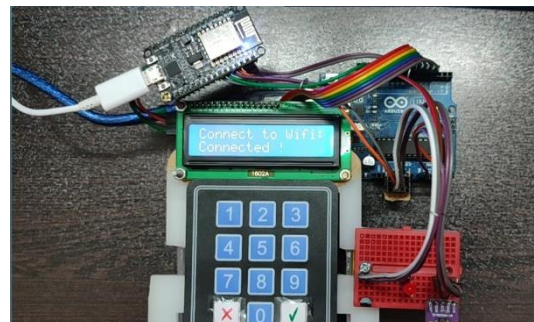
The second method:

Because we are building the first example of this device, so we are looking to find the limitations and capabilities of these boards. So, we change the working process of the device to a menu to find a more complete view of it by testing.

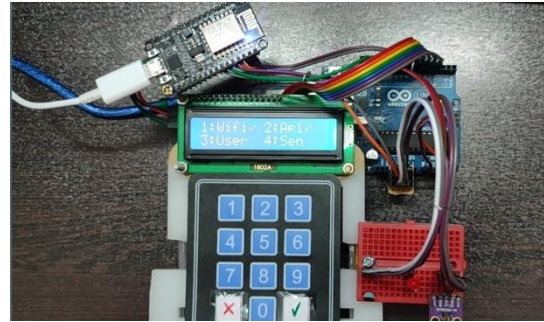
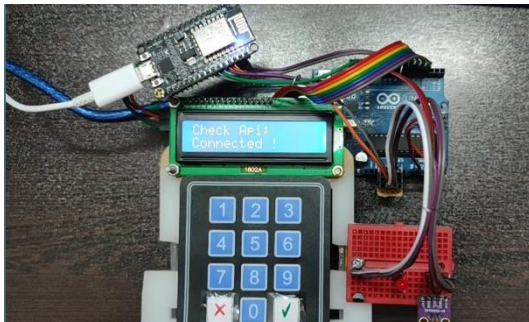
1. The program gives you a menu at the beginning that you can use the keyboard to choose which section to enter.



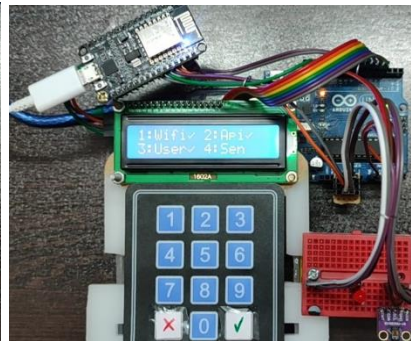
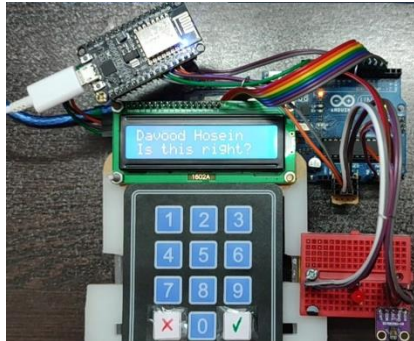
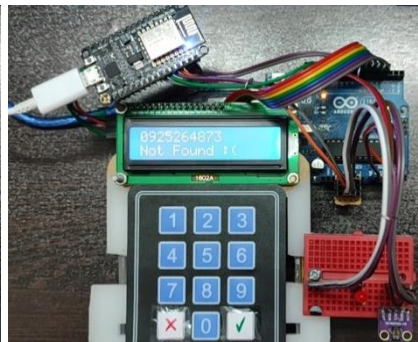
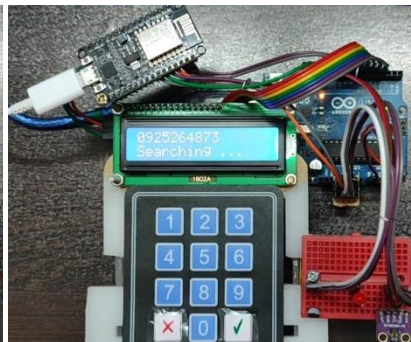
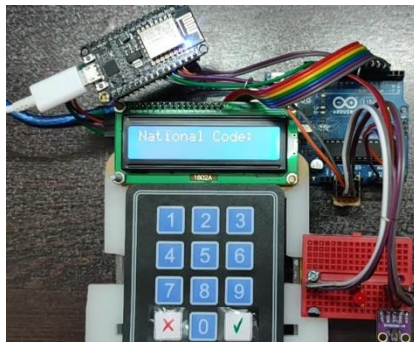
2. By selecting 1 to NodeMCU, we give the command to connect to the router modem, and after connecting, this option will be checked.



3. Selecting 2 instructs NodeMCU to check the API connection whether it is enabled or not, whether the address is correct or not. ... and is checked after communication.



4. By selecting 3, NodeMCU receives the code and then asks you if the information displayed (information sent by the API) is correct? Or it does not find the user and returns to the menu.



5. By selecting 4 sensors are activated and gives the user 3 seconds to get ready and put his hands on the sensors, and after the measurement, the information is sent by pressing the confirm button to save the new record in the API.

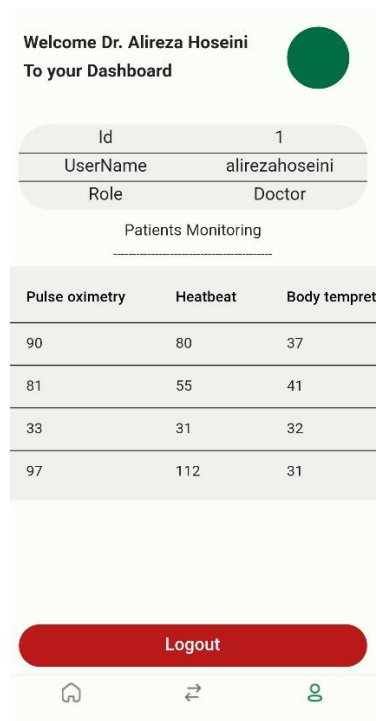
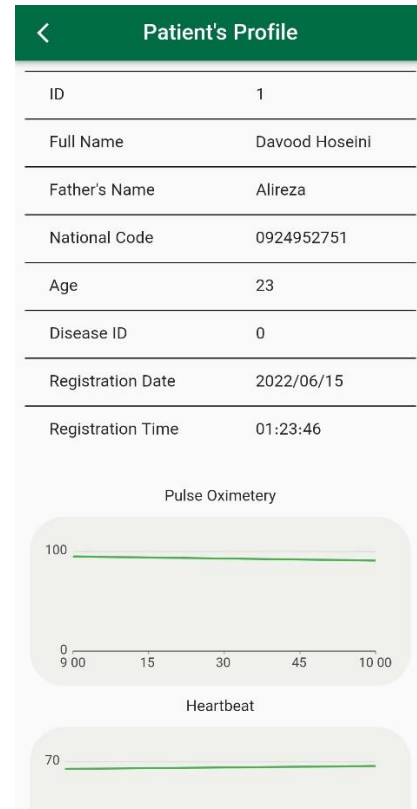
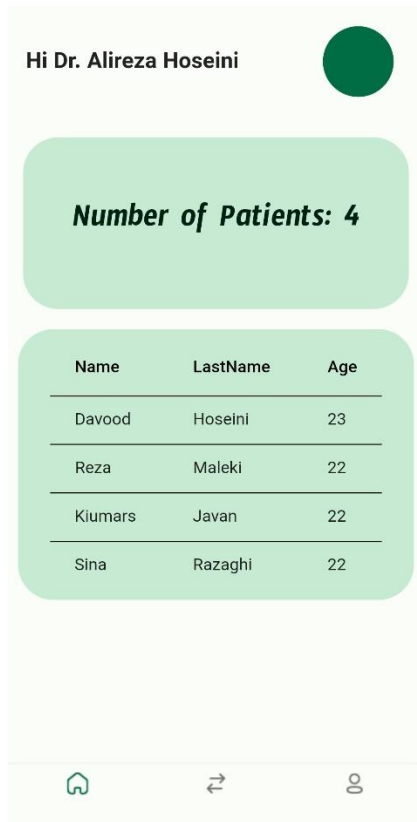


Mobile App

It is so hard to find a person that doesn't use a smart phone these days and they are so comfortable with them because phones are light, portable and have many capabilities. So, with all these feature and advantages, having a mobile application for this service is necessary. For this purpose, we use the flutter framework. Flutter framework is introduced and developed by google and is a cross-platform framework for creating beautiful, fast and interactive applications for IOS, Android, Web, macOS, Linux and Windows with just one codebase.

Our App follows an MVC-like architecture and keep Data Management (Model), Screens (View) and Services (Controller) in separate parts and also our app is written modular and the code is reusable. Doctors and nurses can sign in to the app with their account and monitor their under-supervision patients' profile, their latest and historical vital signs in real-time. They can also receive alert in case of emergency.

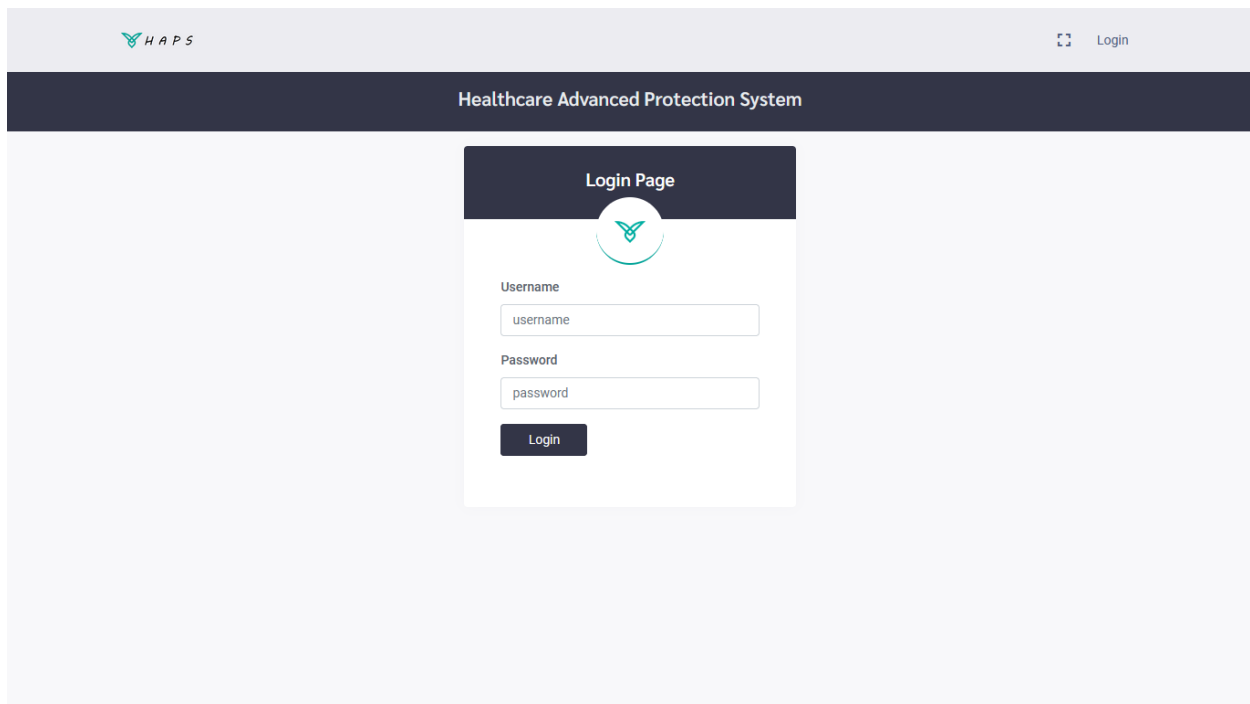
HAPS Mobile app uses SOCKETS technology and keep a super-fast and continually open connection with the server and send and receive data with the minimum delay possible.

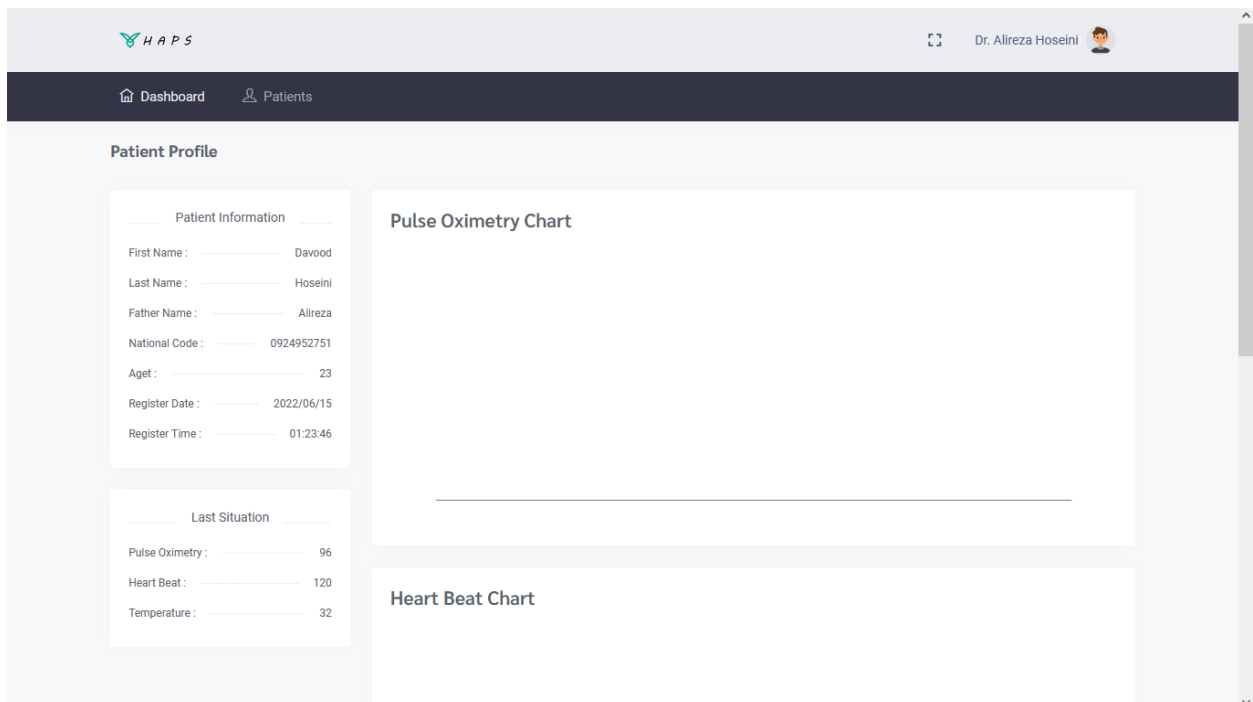
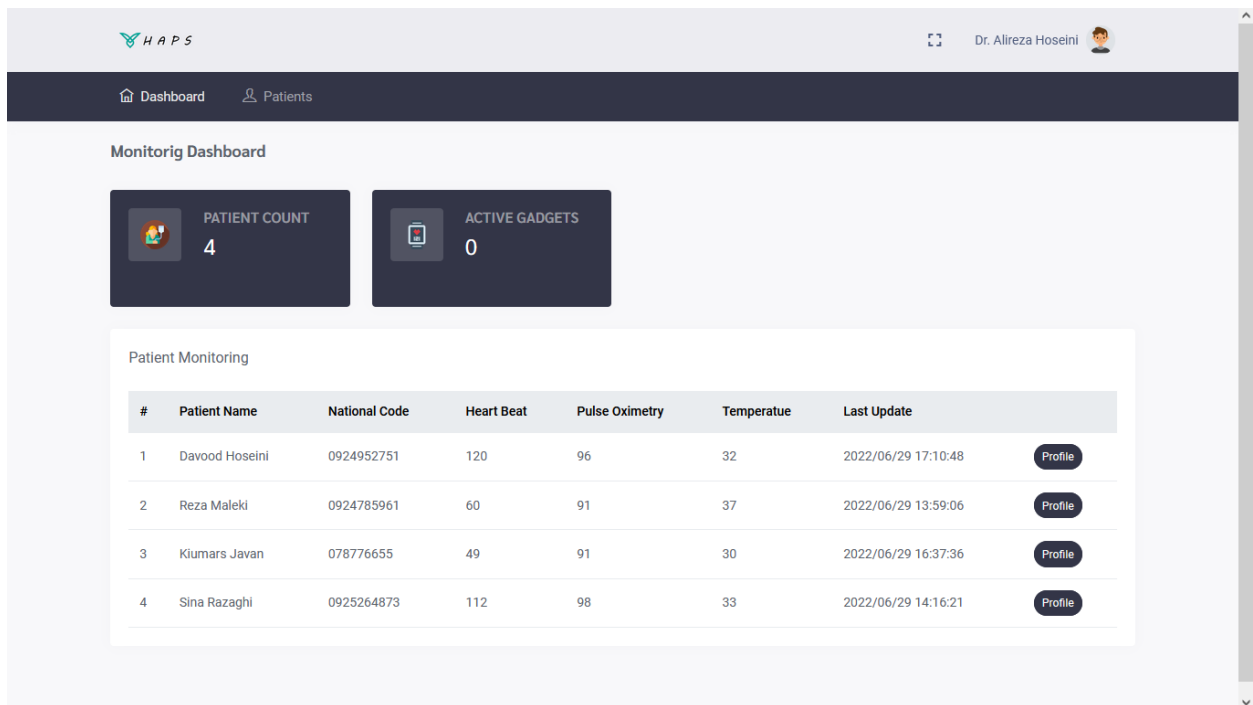


In the above screenshots of the application, you can see the different section of the mobile app and also there is an Alert Screen for when and critical situation happens and the server raises an alert flag.

Web App

The web app gives us a second way to monitor and manage the patients profile and helps us keeping patient under control. This Web panel can get extended to be used by many other users with different roles in a medical clinic. For example, the reception can use it for watch manage all the patients and even nurses and doctors' tasks and shifts. And also, when a patient checks into the clinic they can manage him/her profile and assign a gadget to them. This web app is implemented using react.js technology and is a user-friendly and responsive management dashboard. You can see a couple of screenshots from its UI:





Other

For this project our team mates work on their task parallelly and they needed to be comfortable so we used our own gadget simulator for that time that we wanted to test our apps and simulate a real gadget's performance. Here you can see our desktop gadget simulator:

The screenshot shows a desktop application window titled "MonitorForm". The interface is divided into three main sections: "Online Devices", "Patients", and "Log".

Online Devices		Patients						Log
DeviceName	ConnectionId	Id	PatientName	PulseOximetry	HeartBeat	Temperature	LastUpdate	
gadget_sim	8VN5SyIVkKs...	1	Davood Hoseini	90	47	29	21:46:02	
		2	Reza Maleki	86	42	27	21:46:15	

Below the "Patients" table, there is a form for adding or updating patient data:

PatientId :

PulseOximetry :


HeartBeat :

Temperature :

☐ Is Critical

At the bottom of the window, there is a status bar showing "Socket Connection : Connected" and a "Close" button.

Also, when our very first version of our API released, we used the Swagger for better and simpler use of the API and for documentation. Here are some screenshots:


Swagger
Powered by SMARTBEAR

Select a definition
HAPS Api V1.0.0

HAPS Api V1.0.0 OAS3
/Swagger/v1.0.0/Swagger.json

Authorize

Auth >

Hub >

Log >

Patients >

Tools >

Schemas >

GET /api/Hub/GetOnlineGadgets

POST /api/Hub/InvokeHubMethod

Log >

Patients >

POST /api/Patients/GetPatients

POST /api/Patients/GetPatientById

POST /api/Patients/Add

POST /api/Patients/Edit

POST /api/Patients/Delete

POST /api/Patients/AddPatientSituation

GET /api/Patients/GetPatientHistory

GET /api/Patients/GetPatientLastSituation

GET /api/Patients/GetPatientSituationChart ReportType : hour - min

POST /api/Patients/CriticalCondition

Future Works

This area, this industry and specifically this case has many potentials and can be extended in many ways and developed to be more efficient with more features. For example, our prototype of the smart gadget can gets developed to gets smaller, lighter and compact in a smart wearable watch. It can use a RFID interface for setting patients' id and other security configuration. Our mobile app in next version can uses a always open socket to the server to always be aware of patients situation and receive alerts.

References

- [1] Azzawi, Mustafa & Hassan, Rosilah & Abu Bakar, Khairul Azmi. (2016). A Review on Internet of Things (IoT) in Healthcare. International Journal of Applied Engineering Research. 11. 10216-10221.
- [2] <https://how2electronics.com/interfacing-max30100-pulse-oximeter-sensor-arduino/>
- [3] <https://datasheets.maximintegrated.com/en/ds/MAX30100.pdf>
- [4] <https://www2.ece.ohio-state.edu/~passino/LM35.pdf>
- [5] <https://forum.arduino.cc/t/exit-status-1-i2cwrite-was-not-declared-in-this-scope/572515>
- [6] https://issuehint.com/issue/sparkfun/SparkFun_MAX3010x_Sensor_Library/40
- [7] <https://www.instructables.com/Pulse-Oximeter-With-Much-Improved-Precision/>
- [8] https://github.com/aromring/MAX30102_by_RF/issues/1
- [9] <https://microcontrollerslab.com/esp8266-heart-rate-pulse-oximeter-max30102/>
- [10] <https://www.arduino.cc/reference/en>