

**Faculty of Electrical Engineering, Computer Science and Information Technology Osijek**

FINAL PAPER

e-Health Monitoring

**Student :** Mahmut Aksakalli

**Advisor :** Goran Martinovic

**Date :** 24.06.2018

**CONTENT**

1. INTRODUCTION 3
2. PROBLEM DEFINITION 4
   1. Main Challenges 4
   2. Existing Similar Solutions 4
   3. Proposal of the Own Solution4
3. HARDWARE AND SOFTWARE SOLUTION 5
   1. Setting up Raspberry Pi
      1. Basics of Raspberry Pi 5
      2. Installation of Operating System 7
      3. Control Options of RaspBerry Pi 9
   2. Setting up e-health shield V2.0
      1. e-health shield 11
      2. Electrocardiogram sensor (ECG) 11
      3. Body temperature sensor 12
      4. Blood pressure sensor (sphygmomanometer) 13
4. OWN SOLUTION
   1. Solution Description 14
   2. Implementation of Server 14
   3. Implementation of Client 19
   4. Obtained Results 24
5. CONCLUSION 25
6. REFERENCES 26
7. APPENDIX 27
8. **INTRODUCTION**

Body health is the most important issue for people. Every person wants to maintain physical health of body and protect it against diseases. Developing technologies give opportunity to measure most of the physical parameters related to body health such as blood pressure, body temperature, oxygen in blood. However, people usually go hospital to measure health parameters and this can be painful. A person may need continues health monitoring like old persons or a person who has chronic health issues. For these persons, it can be hard to go hospital frequently.

This project proposes a solution to make available to measure health parameters everywhere with a small device. It’s equipped with most of the sensors to measure health situation of a person and record them. In that way, people can monitor their health consistently. They may also store health records to show them doctors.

In this report, we described definition of problem in section 2. It shows purposes of this project, main challenges, existing similar projects and short description of our solution. In section 3, we explained software and components that we used in this project. Firstly, It describes basics of Raspberry Pi 3, installation of operating system on it and remote accessing techniques to Raspberry Pi 3. Then, it describes details of e-Health shield and explains specifications of each sensor such as blood pressure sensor and temperature sensor. In section 4, it describes our solution to defined problem. Subsection 4.2 explains our solution on server-side implementation which is responsible for collecting sensor data and send it to remote client. Subsection 4.3 explains a web based client application to send request to server and get sensor readings to display on a graphical interface. Subsection 4.4 shows obtained sensor data for 10 different measurements. Lastly, section 5 is a conclusion which summarize problem, aims of project and our solution.

1. **PROBLEM DEFINITON**

**2.1. Main Challenges**

The aim of this project is to develop a small, compact device that helps monitor health situation of a person and record them continually. To archieve that, we need a small-size computer which can read data coming from different health monitoring sensors, process them and send data records to remote client. We need also sensors to observe different physical properties of human body. These sensors should make measurements accurately and should be small enough to make our device movable. A user of device should be able to carry it everywhere he needs.

Visualization of obtained data from sensors is an important step of project. The graphical interface of device should be simple to read and understandable for a person from every age interval. Managing data records and initial setup of graphical interface should also be easy to adjust.

* 1. **Existing Similar Solutions**

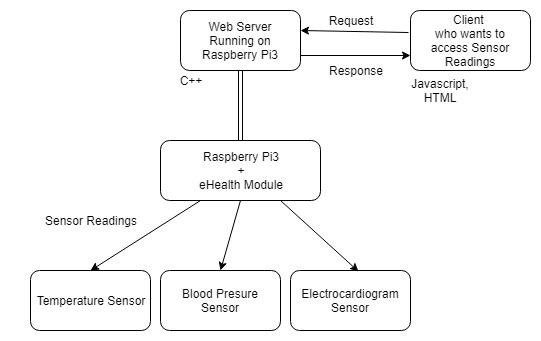
There are some previous solutions to monitor health parameters of a human and inform doctors about health situation of patient remotely. A project aims to send health data of a patient to doctors through a wireless network and give opportunity to monitor the patient in real time [1]. It’s also capable of keeping medical history of a patient so the doctors can see previous sickness records of a patient. Another project targets to monitor health of elderly and persons living alone [2]. They observe daily activities of patient and learn about person. They claim that their system is adaptive to patient’s behaviour so it predicts future health changes. There are also some commersial products to monitor health like wearable watches [3] [4]. They are able to measure heart beats, count steps and track sleeping periods. These products are more suitable for persons who are interested in sport activities.

* 1. **Proposal of the Own Solution**

Our solution consist of a small device that is capable of collecting sensor data and send them to remote client. That small compouter is a Raspberry Pi board in our project. It uses eHealth shield to communicate with sensors and read data from them. Raspberry Pi acts as a server and it binds itself to a port of network interface. If a client make a request to server, it will read sensor data and send response to client back. All the communication between server and client is in form of XML files which describes format of a request and response so both side can understand each other. In addition, server side coded with C++ to communicate with hardware interface easily and fast. Client side coded with HTML,Javascript to create a graphical interface that is easy to use.

1. **HARDWARE AND SOFTWARE SOLUTIONS**

The main goals of this project are visualizing health records of a patient and making it available to access remotely. Figure 3.1 shows block diagram of proposed solution to archieve these goals. Raspberry Pi is main controller of this project. It’s used to collect data from available health sensors. There is also a server that runs on it. The server is responsible for responding request coming from client. The server is a C++ code that listens a specified port number of network interface. It is able to respond all requests coming through this port. However, request should satisfy specific XML format. If a client send a request which fits known XML format, the server send collected server data to client in form of XML. In client-side, there is a web page which is coded with HTML,CSS, Javascript. It sends request to server and takes data of sensor readings. Then, it shows them user in a suitable graphical interface.



**Figure 3.1:** Block diagram

* 1. **Setting up Raspberry Pi**

This section includes an introduction to raspberry pi boards and installation process of an operating system on it.

* + 1. **Basics of Raspberry Pi**

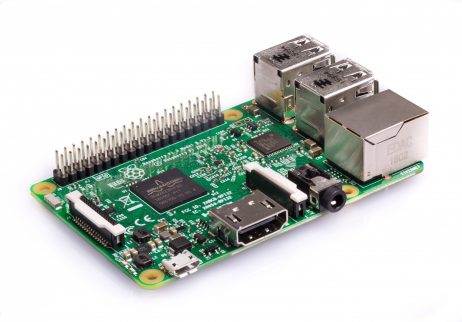
Raspberry Pi is a small size computer that have developed for educational purpose. It provides people of all ages to explore computer world and make interaction with programming languages. It can handle all tasks that computer does. Although, it is very small size, it is capable of having most of input devices such as monitor, keyboard, mouse. People can plug a monitor to display screen, then they can surf on the internet and watch videos etc. Raspberry Pi has an important difference from desktop computer. It has GPIO pins (General purpose input output pins) that can be programmed to control sensors and receive information from its environment via sensors. These pins can be configured as input or output to interact with outside world. They have also different functionalities to communicate with sensors such as I2C, SPI, UART. The thing that makes Raspberry Pi much popular is the large number of user community. It is the most popular devices is used in projects nowadays. You can find lots of well-prepared documents related to Raspberry Pi [5].

The latest product in Raspberry Pi world is Raspberry Pi 3. It’s hardware specifications are listed below [6].

* Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
* 1GB RAM
* BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
* 40-pin extended GPIO
* 4 USB 2 ports
* 4 Pole stereo output and composite video port
* Full size HDMI
* CSI camera port for connecting a Raspberry Pi camera
* DSI display port for connecting a Raspberry Pi touchscreen display
* Micro SD port for loading your operating system and storing data
* Upgraded switched Micro USB power source up to 2.5A

It contains most of popular communication technologies on it. You can display computer screen via HDMI, and connect internet via Ethernet or using wireless networks. You can choose to communicate with sensors via cable or using wireless bluetooth technology to get data from environment. Pi camera gives you an opportunity to see what’s going on outside world and process images to record and detect objects around your device.

Raspberry Pi contains a Quad Core 1.2 Ghz processor and 1GB RAM. Although these components consume low energy, they provide fast and enough computing power to handle environmental devices. There is also an SD card to store operating system and data on it. A user can storage personal files and install additional programming tools or libraries on SD card



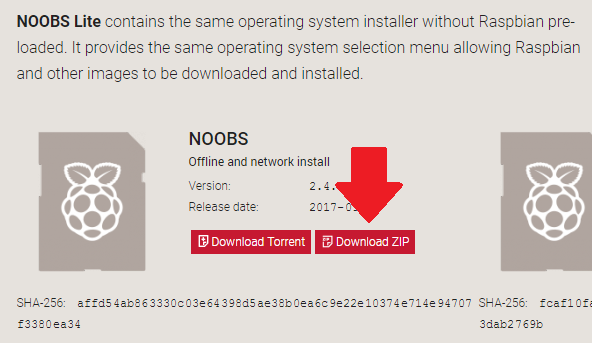
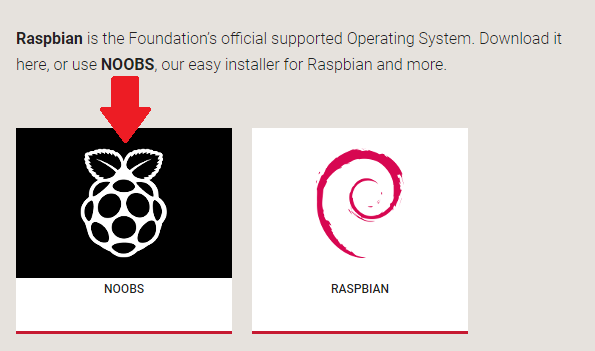
**Figure 3.2:** Raspberry Pi 3 Model B

Every Pi device contains an operating system which is generally Linux based system. Official operating system for Raspberry Pi is Raspbian. However, you can also install different distributions of linux such as Ubuntu Mate, PINET, LIBREELEC, RISC OS etc. Raspbian OS comes with pre-installed software tools for programming and other purposes.Some of installed softwares are Python, Scratch, Sonic Pi, Java, Mathematica.

* + 1. **Installation of Operating System**

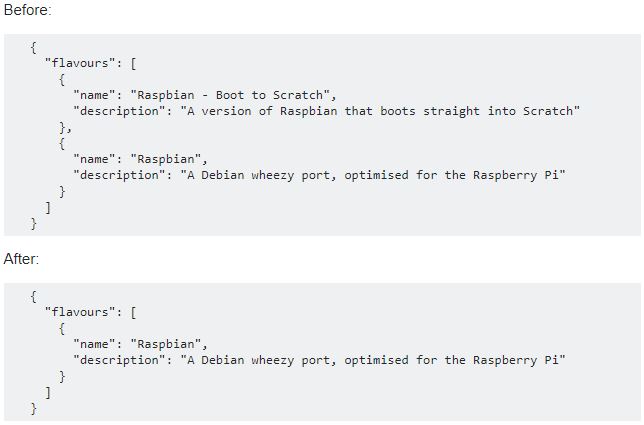
Official Raspberry website provides operating system images for most of the OS of Raspberry Pi. You can download an OS image and install it on your device but there is also one more option. They release an easy install tool called NOOBS (New Out Of the Box Software). It’s an operating system installer that give option to install OS with reduced installation steps [7].

* Download NOOBS from <https://www.raspberrypi.org/downloads/>



**Figure 3.3:** Download NOOBS

* After downloading NOOBS, unzip the contents to a folder
* You need to copy content of folder to SD Card
* Generally, you need a monitor and keyboard for first installation because there is an option menu on installation process to select OS. You also need to arrange remote control settings of Raspberry Pi to connect it to wireless network. In this way, you can access your device using SSH connection. However, if you don’t have a monitor to display screen, you can also set same parameters of NOOBS to make installation automatically and connect it to a wireless network.
* Edit file \os\Raspbian\flavours.json to set desired version of Raspbian OS as an only option



**Figure 3.4:** content of \os\Raspbian\flavours.json before and after editing

* Edit file recovery.cmdline to append silent install option

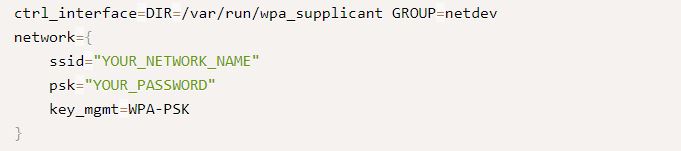
Before

runinstaller quiet vt.cur\_default=1 coherent\_pool=6M elevator=deadline

After

runinstaller quiet vt.cur\_default=1 coherent\_pool=6M elevator=deadline silentinstall

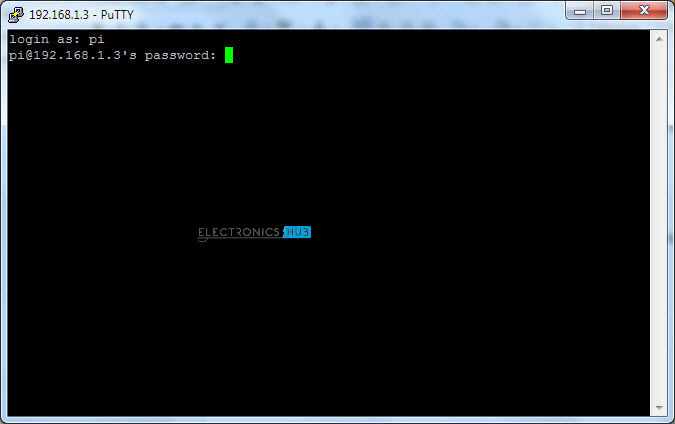
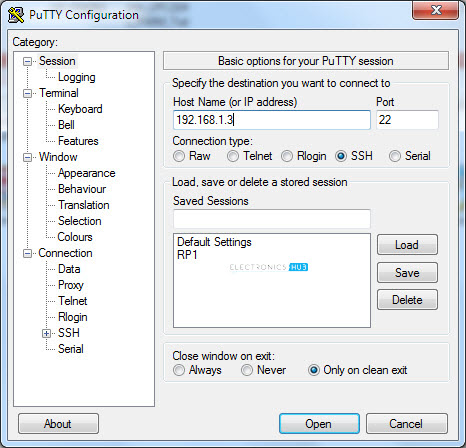
* Open SD Card folder and create an empty file named **SSH**, NOOBS activate SSH port automatically after installation finished.
* Create a file named **wpa\_supplicant.conf** in SD Card, and edit its content as below



**Figure 3.5:** Content of wpa\_supplicant.conf

This file enables your device to connect given wireless network.

* Now, your SD Card is ready to make installation automatically, then it will connect wireles network defined in wpa\_supplicant.conf file. Plug your SD Card to Rapberry Pi device and power it. Installation process generally takes time between half an hour and an hour.
* You can use [Advanced Ip Scanner](http://www.advanced-ip-scanner.com/) to detect ip address of your device to connect it via SSH
* Install [Putty](https://www.putty.org/) to establish an SSH connection between your computer and Raspberry Pi. You should write ip address of Pi to Host Name area of Putty and default SSH port is 22. You need to set connection type as SSH [7]



**Figure 3.6.a:** Putty User Interface **Figure 3.6.b:** SSH Command Prompt

* Default SSH username is **pi** and password is **raspberry .**
  + 1. **Control Options of RaspBerry Pi**

You are able to control your device using commands. You can use all Linux bash commands to configure some settings of operating system or access files to make operations on them such as creating, removing or moving files.

Alternatively, you can access the graphical interface of Raspberry Pi using VNC. It is a graphical desktop sharing software that allows you to access another computer desktop remotely. It also allow you to use your keyboard for typing and mouse events to control remote computer [8].

You need to install VNC client to your pi computer using command below.

sudo apt-get update

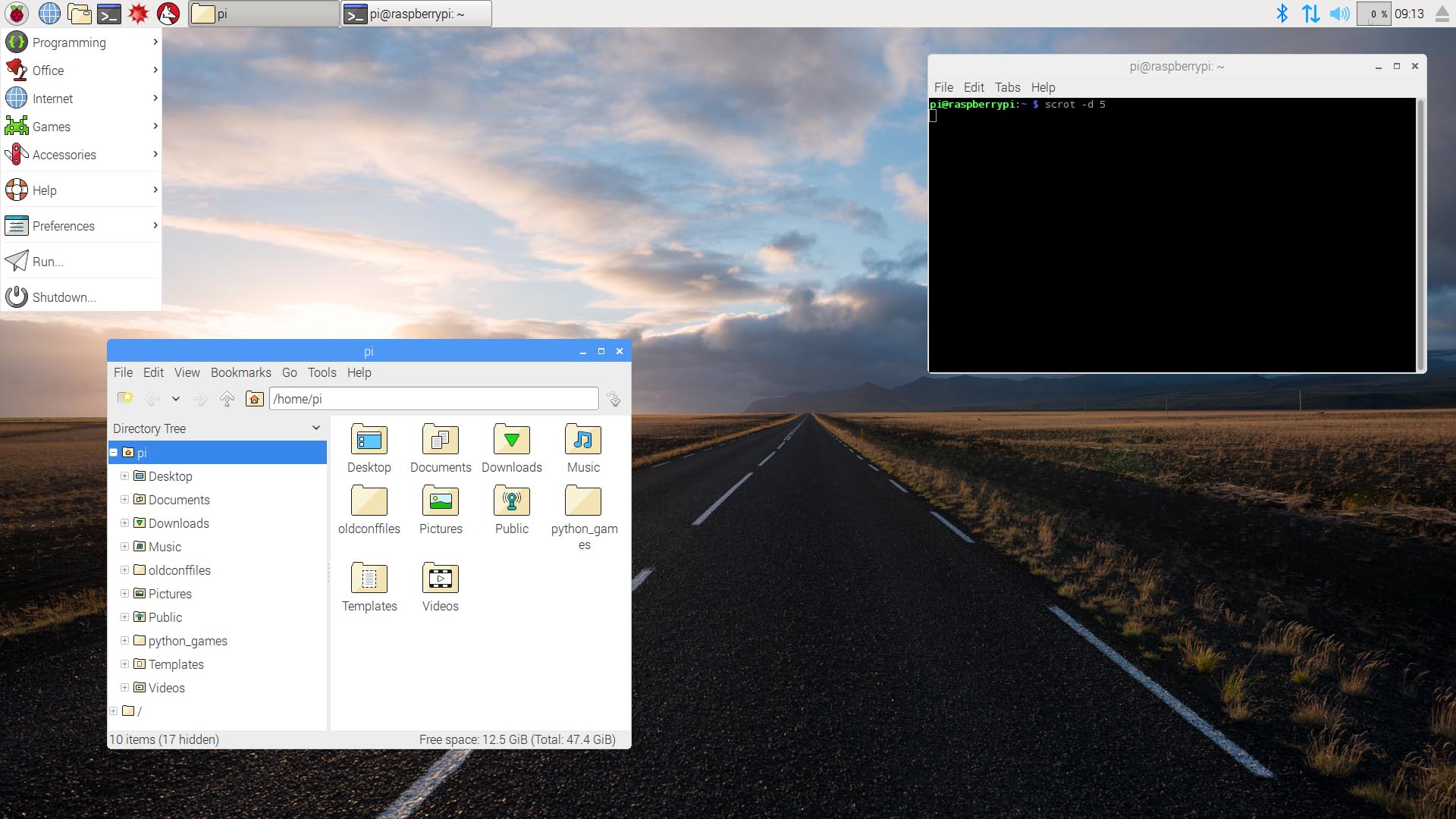
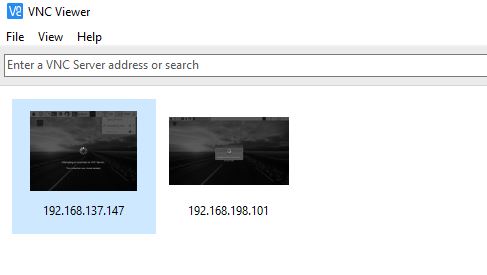
sudo apt-get install realvnc-vnc-server realvnc-vnc-viewer

Then, you need to enable VNC. To do that, you should write below command to command line

sudo raspi-config

and go to **Interfacing Options**, select **VNC** > **Yes**

In computer side, you need to download a VNC viewer software. You can use RealVNC which can be download from <https://www.realvnc.com/download/viewer/>

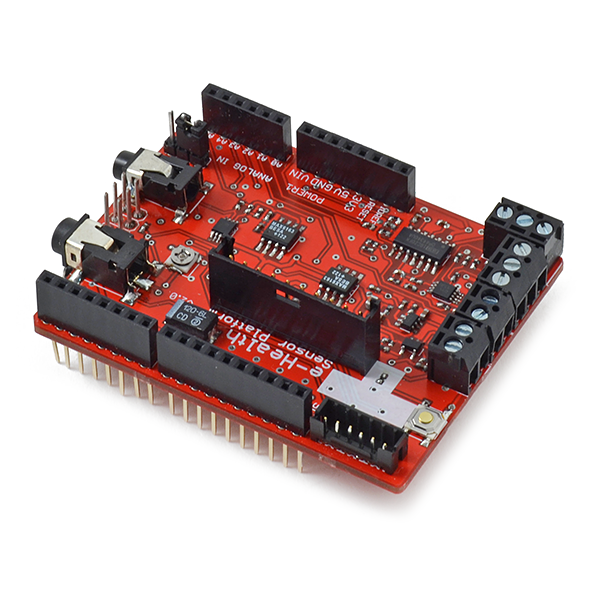


**Figure 3.7.a:** Interface of RealVNC software **Figure 3.7.b:** Desktop view of Raspberry Pi

Figure 3.7.a shows interface of RealVNC software. You can use it to connect your Raspberry Pi Desktop, type ip address of your device to RealVNC software. Figure 3.7.b shows desktop of your device on a new window after you connected Pi Desktop with RealVNC..

* 1. **Setting up e-health shield V2.0**

This section includes an introduction to e-health shield and sensors used in this project.



**Figure 3.8:** e-health shield v2.0

* + 1. **e-health shield**

The e-health sensor shield v2.0 is a development board which is developed for Arduino and Raspberry Pi platforms to perform medical applications easily.It’s developed by Cooking Hacks company in order to help researches, developers to measure physical body data for test and experimentation. It gives opportunity to use 9 different health sensor to monitor situation of patient real time and collect physical data frequently. It supports wireless options to send collected data to remote server using one of wireless technologies such as Wi-Fi, 3G, GPRS, Bluetooth, 802.15.4 and ZigBee. It’s compatible with Arduino, Raspberry Pi and Intel Galileo boards [9]

It supports sensors listed below.

- Pulse and oxygen in blood sensor (SPO2)

- Airflow sensor (breathing)

- Body temperature sensor

- Electrocardiogram sensor (ECG)

- Glucometer sensor

- Galvanic skin response sensor (GSR - sweating)

- Blood pressure sensor (sphygmomanometer) V2.0

- Patient position sensor (Accelerometer)

- Electromyography Sensor (EMG)

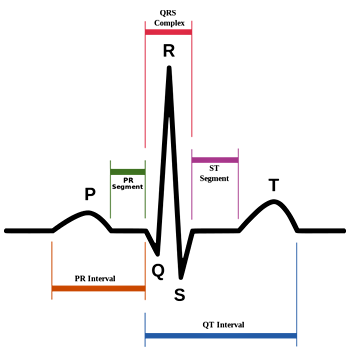
We have three different sensors to use in this project. They are electrocardiogram sensor (ECG), body temperature sensor and blood pressure sensor (sphygmomanometer).

* + 1. **Electrocardiogram sensor (ECG)**

Electrocardiogram is a test that records the electrical activity of your heart through small electrode patches that are attached to the skin of chest. While heart is pumping blood, a series of electrical signals are sent to heart by autonomic nervous system. These signals can be measured on skin using two or more electrodes placed in various postions on the chest and limbs. It’s useful to observe situations listed below [9].

* Check your heart rhythm
* See if you have poor blood flow to your heart muscle (this is called ischemia)
* Diagnose a heart attack
* Check on things that are abnormal, such as thickened heart muscle

ECG may not able to detect these heart problems evey time. The accuracy of ECG depends sensor quality and test conditions.



**Figure 3.9:** ECG signal sample

Figure 3.9 shows a sample ECG signal. To quantify a signal, QRS complex is the most used feature. It indicates the main pumping contractions of the heart. R peak in QRS complex is used by heart rate algorithms to measure the amount of time that occurs between each heartbeat [10].

* + 1. **Body temperature sensor**

Body temperature sensor is used to measure temperature of human body. The temperature of body can change depends on the time of day and level of activity of a person. 37°C is normal body temperature for an adult person.The temperature sensor can measure temperature between -40°C and 100°C. Its accuracy is +/-0.4°C% [9]



**Figure 3.10:** Temperature sensor

Different part of a body can be used to measure temperature. The result of measurement is an output voltage which is proportional to temperature level.

* + 1. **Blood pressure sensor (sphygmomanometer)**

When heart pump blood to body, pressure occurs in the arteties called blood pressure. It’s expressed as a measurement with two numbers which are top(systolic) and bottom(diastotic) like 120/80 mm Hg. The top number refers to the amount of pressure in your arteries during the contraction of your heart muscle. This is called systolic pressure. The bottom number refers to your blood pressure when your heart muscle is between beats. This is called diastolic pressure. Both numbers are important in determining the state of your heart health [9].



**Figure 3.11:** Blood pressure sensor

Systolic pressure should be higher than 90 and lower than 120 for ideal reading. Diastotic pressure should be higher than 60 and lower than 80 for ideal reading.

1. **OWN SOLUTION**
   1. **Solution Description**



**Figure 4.1:** Our solution

Our project has two main section. One of them is explained in subsection 4.2 called server-side. It has both hardware and software part. Hardware part is responsible for getting measurements about health situation of patient and software part is responsible for communicating with client to send measurements. Other section is subsection 4.3 called client-side. It’s able to send request to get data from remote server. Client-side has only software part. It’s responsible for visualizing data coming from server and make it easy to access measurements for patients. Lastly, subesection 4.4 shows obtained measurement results of our solution.

* 1. **Implementation of Server**

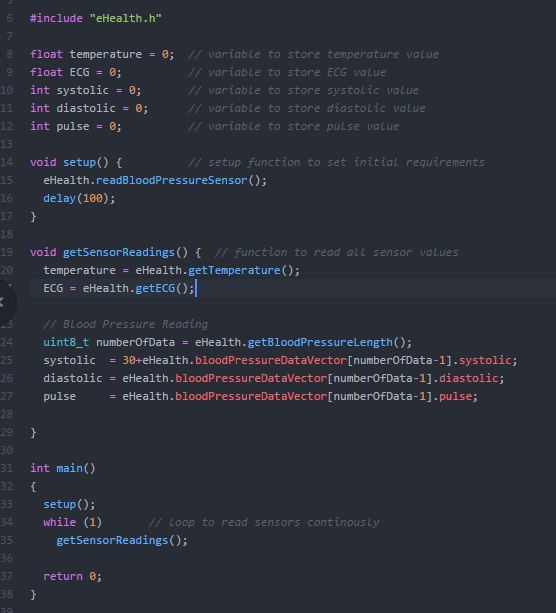
The aim of server-side implementation is reading sensor values and sending them to client in a suitable form. To archieve this, Raspberry Pi reads sensor values via eHealth module. The developers of eHealth module released an ArduPi library and e-Health library. These libraries makes easy to read data from connected sensors to eHealth module. The code snippets for reading sensor values is represented Figure 4.2 below.

To obtain an application, you need to compile C++ code with libraries mentioned above. g++ is a suitable tool to compile C++ projects. After putting all library files on same folder, you can use below commands to compile project [9]

g++ -c arduPi.cpp -o arduPi.o

g++ -c eHealth.cpp -o eHealth.o

g++ -lpthread -lrt main.cpp arduPi.o eHealth.o -o server



**Figure 4.2:** main.cpp for sensor readingds

In this project, Raspberry Pi3 acts as a server which reads sensors and sends them to client when it took request from client. The server listens a specified network port continuously and it uses an application coded with C++ to handle server-side operations. C++ is a strong and efficient language to handle both sensor readings and respond request of client. That’s why it fits requirements of this project well. However, a server needs a protocol for exchanging information between client and itself. SOAP(Simple Object Access Protocol) is an XML-based messaging protocol for transferring information among computers. It enables client applications to easily connect to remote services and call remote methods via HTTP.

A SOAP message consist of three parts [11]

* An envelope which defines the message structure and how to process it.
* A set of encoding rules for expressing instances of defined datatypes.
* A convention for representing procedure calls and responses

To use SOAP protocol in C++ language, there is an API called gsoap. It automatically generates C++ source code to consume and deploy XML request and responses. When you installed gsoap library, it comes with two tools called wsdl2h, soapcpp2. Wsdl2h tool converts a set of WSDL or XSD file to a header file. Then, soapcpp2 tool takes header file as an input to generate C++ code implementation and services automatically.This header file contains data binding interface which describe format of data exchanged between server and client. It also contains services that is allowed to run remotely by client.Because of that, our project needs a header file to define data exchange format between server and client. We should transfer following sensor data to client [12].

- float temperature

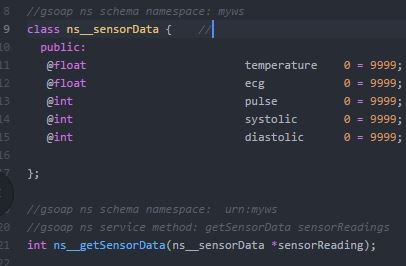
- float ecg

- int pulse

- int systolic

- int diastolic

To transfer all these information, we defined a class which has all of the attributes above. Header file contains also available service methods that is accessible remotely by client.The content of header file is shown Figure 4.3 below .



**Figure 4.3:** sensorData.h

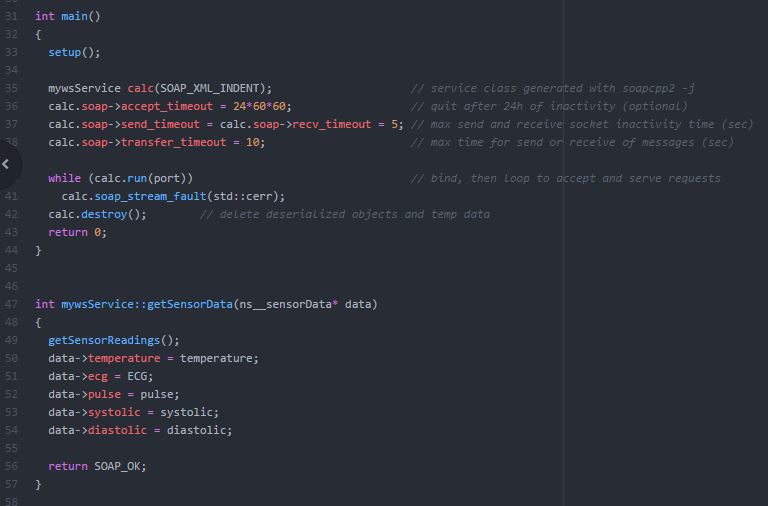
To generate C++ source code from header file, we can use soapcpp2 tool.

soapcpp2 -j -SL -I/path/to/gsoap/import sensorData.h

After executing above command, it creates files listed below [12].

* soapC.cpp core methods and classes
* soapService.cpp class and methods to construct soap server
* soapService.h header file for soapService.cpp
* soapProxy.cpp Client Proxy class
* soapProxy.h header file for soapProxy.cpp
* myws.nsmap namespace mapping table
* myws.getSensorData.req.xml xml format of soap request of client
* myws.getSensorData.res.xml xml format of soap response of server

We can use these files to create our application-specific server. A client of these server should make request in form described myws.getSensorData.req.xml .Then, server responds correct requests in form described myws.getSensorData.res.xml. Updated version of server-side code is given Figure 4.4 below.



**Figure 4.4:** Stand-alone server code

In main, setup function remained like Figure 4.2 but the end condition of while loop changed with state of specified port. When we bind our server to a specific port of network interface, it starts to listen this port. Then, request coming from client is redirected our server application by operating system. If a request has correct xml format, server invokes getSensoData method and returns sensor readings as a response xml. getSensorData function defined in header file so we implemented it in this stand-alone server code.

Figure 4.5 shows XML format of request coming from client. It contains all three part of SOAP message mentioned above.

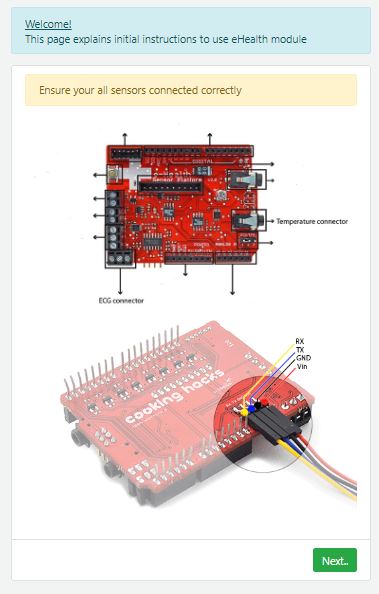


**Figure 4.5:** XML request format

* 1. **Implementation of Client**

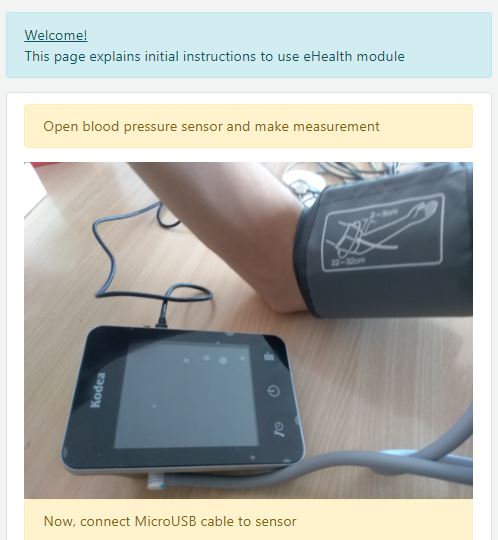
We created server-side application of our project which returns sensor values if a client send a request. There is an XML based protocol between server and client to communicate so we need to implement a client that is able to send SOAP request and receive SOAP response. In our project, we implemented web based user interface that acts like a client.

Client side application starts with instructions about initial settings of sensors. Then, it shows sensor readings an display them on a web page.Figure 4.6 shows welcome screen of client-side application. It warns user to ensure sensors connected properly.



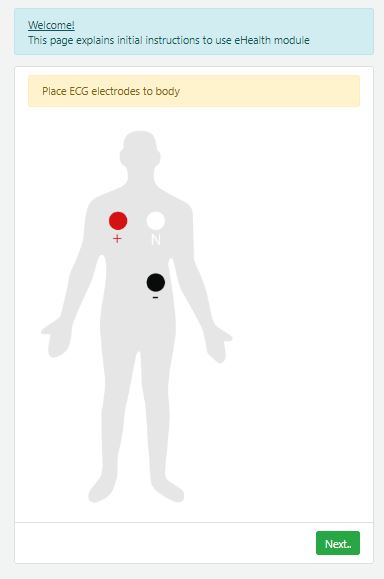
**Figure 4.6:** Welcome Screen

Then, user clicks next and Figure 4.7 is displayed.



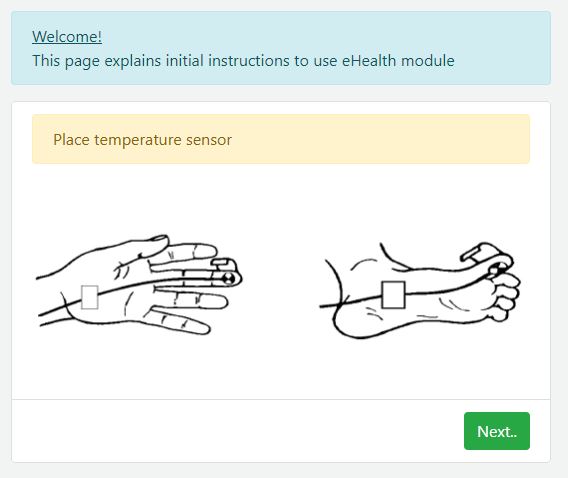
**Figure 4.7:** Blood pressure sensor instructions

Then, user clicks next and Figure 4.8 is displayed. It contains instructions about ECG sensor.



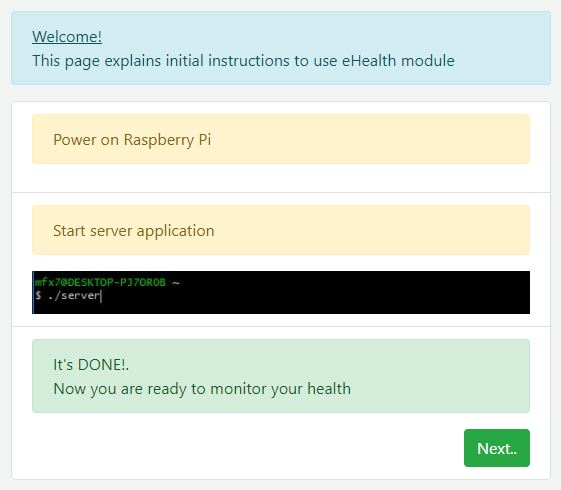
**Figure 4.8:** ECG sensor instructions

Figure 4.9 shows instructions about temperature sensor.



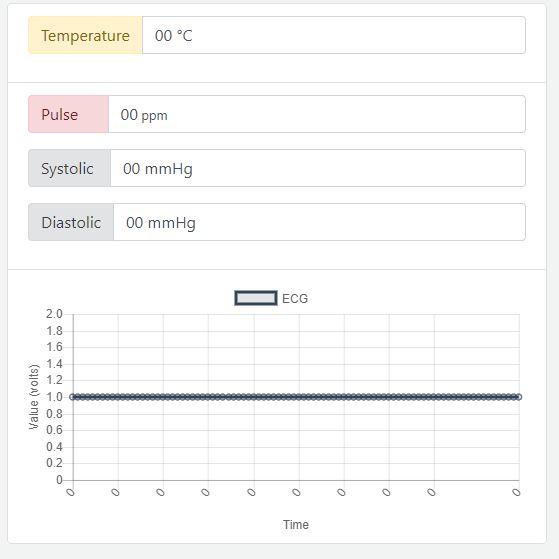
**Figure 4.9:** Temperature sensor instructions

After checking all instructions about sensors. It’s ready to start Raspberry Pi and initialize server application.



**Figure 4.10:** Starting server application

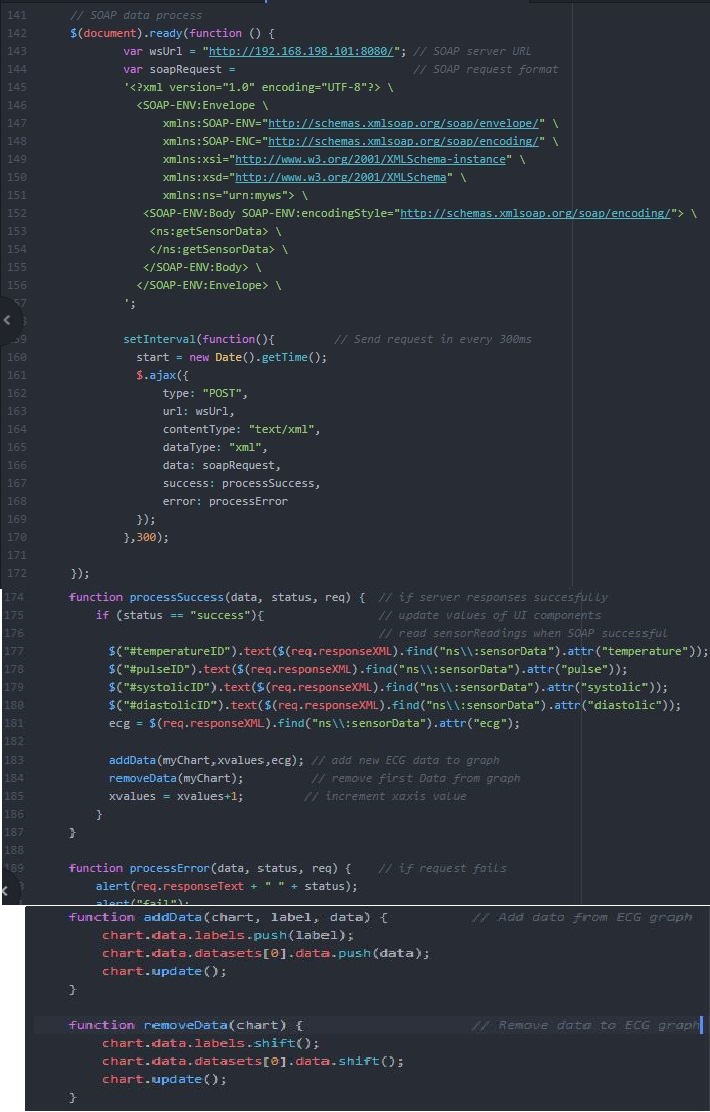
Our project has sensor data coming from three different sensor. Temperature sensor gives output in terms of Celcius(°C) degree. Blood pressure sensor gives number of heart beats per minute(ppm) and blood pressure is expressed as systolic and diastolic parameters(mmHg). ECG sensor collects data about electrical activity of heart in terms of voltage(V). Graphical interface of client-side application contains all of these parameters. It sends requests to server frequently and display received values on screen. All sensor data are shown directly except ECG values. ECG values are drawn on a graph because individual values are meaningless to interpret about health situation.



**Figure 4.11:** Graphical interface to show sensor values

Client-side code consist of HTML,CSS and Javascript. HTML,CSS are used to create structure of graphical interface and Javascript is used to handle communication with server. CSS framework bootstrap is added to make graphical design easy, Bootstrap contains ids and classes to edit style properties of HTML tags. In javascript side, Jquery API is used to send request to server and process received response. It’s also responsible for updating graph of ECG values. Another API used in Javascript side is Chart.js. It’s used to create a graph for displaying ECG values.

Jquery API manages communication with server. It send SOAP request in every 300ms. When server responded request, it updates components of graphical interface. If request fails, it will display alert that indicates fail status. Figure 4.12 shows code snippets related to SOAP request and update of graphical interface components.



**Figure 4.12:** Javascript code to send request

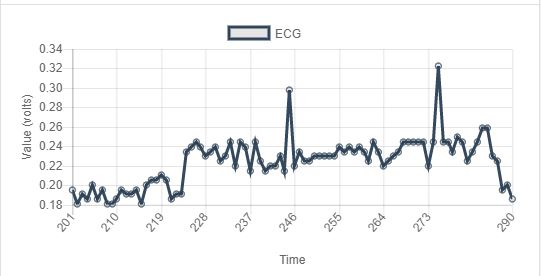
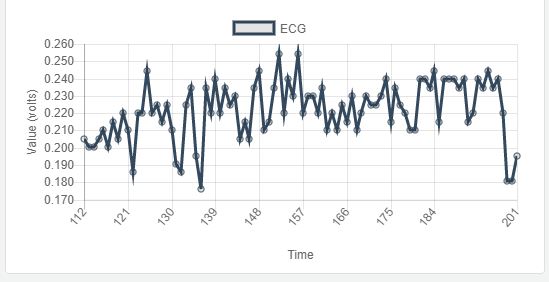
* 1. **Obtained Results**

In this section, you can observe obtained measurement results using our developed system. Table 4.1 shows ten different measurement results for temperature, pulse, systolic and diastolic parameters. Temperature measurements are between 36-37 °C which is normal for a human. Heart beats per minute, pulse, parameters are between 60 and 85 bpm. Interval between 60-100 bpm is normal for a adult human. Systolic and diastolic parameters are also changing in an acceptable interval except first measurement. It has 87 mmHg diastolic value which is high respect to normal acceptable interval. For an adult, systolic is between 90-120 mmHg and diastolic is between 60-80 mmHg.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Measurements** | | | | | | | | | | | | | |
| **Temperature** | 36.3 | 36.7 | 37 | | 36.1 | 36.2 | | 37.0 | | 36.2 | | 36.3 | 36.5 | 36.6 |
| **Pulse** | 80 | 75 | 62 | 75 | | 69 | 71 | | 70 | | 84 | | 81 | 84 |
| **Systolic** | 119 | 114 | 96 | 120 | | 98 | 106 | | 105 | | 106 | | 100 | 104 |
| **Diastolic** | 87 | 67 | 75 | 78 | | 58 | 62 | | 68 | | 67 | | 71 | 63 |

**Table 4.1:** Obtained Results

ECG sensor readings are interpreted as a graph. Figure 4.13 shows two different measurement for ECG parameter.

** **

**Figure 4.13:** ECG data

1. **CONCLUSION**

This project help patient monitor health parameters such as temperature of body, blood pressure, ECG. Sensors to monitor health parameters became easy to access for people. Because of that, people are able to measure health parameters in places other than hospitals. This created huge opportunity to observe health parameters more frequently and regularly for patients and old people. However, they need some visualization tools to monitor data coming from health sensors.

Our project uses Raspberry Pi 3 as a main controller and takes sensor values with collaboration of e-health shield. At the same time, it acts like a server so it listens specified port number and collects request coming from client. If request has a suitable XML format, it will response client with current sensor values. This communication between server and client occurs through SOAP protocol. In client-side, there is a web based structure so a user can monitor sensor readings on browser. Client-side application starts instructions about initial configurations of components.Then, it sends request to server in every 300ms and updates displayed sensor values. It makes monitoring health parameters easy due to its simple and basic user interface

Developing technologies affect monitoring ways of human health. People don’t need to go hospital to observe their health parameters anymore. They are able to use small and effective sensors to measure their situation everywhere. Our solution offers these people to monitor sensor readings more easily and compact way. As a result, our project gives opportunity to observe health parameters like body temperature, blood pressure, heart beat rate and ECG of patient everywhere.

1. **REFERENCES**
2. V. Chan ; P. Ray ; N. Parameswaran (2008), Mobile e-Health monitoring: an agent-based approach
3. HaiderMshali ; TayebLemlouma ; DamienMagoni, Pervasive and Mobile Computing, Elsevier, 2018, 43, pp.1 – 19
4. Fitbit smart watch, <https://www.fitbit.com/eu/home>
5. Watches for health monitoring, <http://gadgetsandwearables.com/2018/04/29/best-fitness-trackers/>
6. Raspberry Pi 3 features, <https://www.zdnet.com/article/what-is-the-raspberry-pi-3-everything-you-need-to-know-about-the-tiny-low-cost-computer/>
7. Rapberry Pi 3 specifications, <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>
8. NOOBS installation, <https://www.electronicshub.org/raspberry-pi-basic-setup-without-monitor-keyboard-headless-setup/>
9. Remote access to Pi 3, <https://www.raspberrypi.org/documentation/remote-access/vnc/README.md>
10. e-health shield and sensor details, <https://www.cookinghacks.com/documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical/>
11. how ECG works, <https://imotions.com/blog/measuring-the-heart-how-does-ecg-and-ppg-work/>
12. SOAP Protocol, <https://en.wikipedia.org/wiki/SOAP>
13. tutorials about gsoap library, <https://www.genivia.com/tutorials.html>
14. documentation about Bootstrap API, <https://getbootstrap.com/docs/4.0/getting-started/introduction/>
15. documentation about Query API, <https://api.jquery.com/>
16. documentation about Chart.js, <http://www.chartjs.org/docs/latest/>
17. **APPENDIX**

**Appendix 1**

Client-side and Server-side code implementation of project can be found in link below.

<https://github.com/mahmut-aksakalli/eHealthMonitoring>