IOCA 2021

Article

Two Optimized IoT Device Architectures Based on Fast Fourier Transform to Monitor Patient's Photoplethysmography and Body Temperature

Janith Kodithuwakku [1], Dilki Dandeniya Arachchi [1], Saw Thiha[1] and Jay Rajasekera[1]

[1] Digital Business and Innovations, Tokyo International university, Saitama, Japan. mjpkodithuwakku@gmail.com (J.K.); dadmahindika@gmail.com (D.D.A.); michael.sawthiha@gmail.com (S.T.); jrr@tiu.ac.jp (J.R.)

Abstract: The measurement of Blood-Oxygen Saturation (SpO2), Heart Rate (HR), and Body Temperature are very critical in monitoring patients. Photoplethysmography (PPG) is an optical method that can be used to measure Heart Rate, Blood-Oxygen Saturation, and many analytics about Cardiovascular Health of a patient by analyzing the waveform. With the COVID-19 pandemic, there is a high demand for a product that can remotely monitor such parameters of a COVID patient. This paper proposes two major design architectures for the product with optimized system implementations by utilizing the ESP32 development environment and cloud computing. In one method it discusses edge computing with the Fast Fourier Transform (FFT) and Valley Detection algorithms to extract features from the waveform before transferring data to the cloud while the other method transfers raw sensor values to the cloud without any loss of information. This paper especially compares the performance of both system architectures with respect to bandwidth, sampling frequency, and loss of information.

Keywords: Photoplethysmography (PPG); Fast Fourier Transform (FFT); Internet of Things (IoT); Blood-Oxygen Saturation (SpO2); Heart Rate (HR); Body Temperature; Bandwidth Optimization; Cloud computing.

1. Introduction

Monitoring patients with highly contagious illnesses is extremely dangerous for medical staff as they also can be exposed to an unhealthy and harmful environment. IoT remote monitoring devices are very helpful to solve this problem by providing a remote dashboard to visualize needed medical parameters in real-time without physically contacting patients. These systems also reduce the time that has to be spent for checking each patient by the medical officer. Furthermore, these systems have a potential to enhance patient monitoring with smart functions like generation of notifications, reports, etc. [1]

Telemedicine systems with remote monitoring facilities became popular over past few decades. Some of the IoT pulse oximetry systems are designed to be used in a limited area with wireless communication. Example implementations for these methods are personal ad-hoc wireless network using Wi-Fi with station mode (STA) and point access mode (AP) [2] and wireless sensor networks (WSN) using ZigBee [3]. Some other existing efforts have been focused on real-time personal pulse oximetry monitoring with reduced data transferring time by using ISO/IEEE 11073 message format and server-side data processing in order to reduce the average response time to 251 milliseconds [4]

This paper discusses two efficient design architectures to implement IoT systems with a comparison of their capabilities. Both architectures include a cloud data server,

Citation: Lastname, F.; Lastname, F.; Lastname, F. Title. *Proceedings* **2021**, 68, x. https://doi.org/10.3390/xxxxx

Published: date

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).

dashboard, a capturing device to interface a clinical PPG sensor and a body temperature sensor as shown in the Figure 1. This paper explains the architecture of each subsection of the systems with algorithms and methodologies used to improve the proposed systems by eliminating limitation such as high bandwidth usage and accessibility limitations of existing systems. The first part of the paper discusses the main idea of the two proposed architectures in detail as shown in the Figure 2. Then the second part discusses the comparison of the results obtained from the two methods. Finally, the conclusion and the future work according to the obtained results.

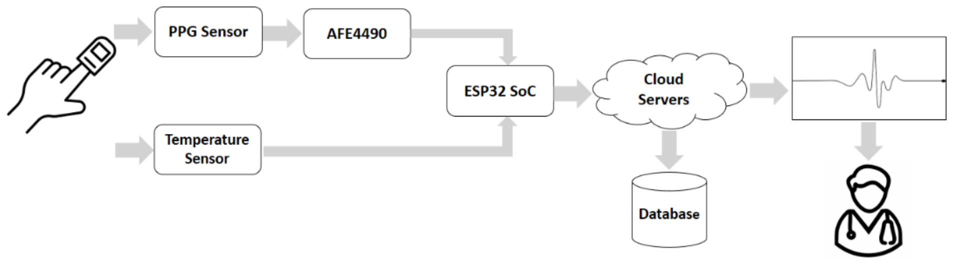


Figure 1. Structure of the proposed IoT system.

2. Methodology

2.1. System Overview

Proposed IoT system consists of loosely coupled layered architecture with five main layers. Service oriented loosely coupled architecture of the system ensures the scalability of the system as well as increase the ability to change, remove or upgrade each layer without affecting to any other layer of the system.

In the sensing layer, the wireless sensor module is used to get the readings. This hardware module is the only part that physically contacts with the patient. ESP32 has the built in Wi-Fi function and 240 MHz clock speed to become a good candidate for a IoT system with its 32-bit microcontroller and other useful features. Hence it is capable of signal processing and data encoding described in the later sections in this paper. Moreover, the proposed system enriched with wireless data transmission, optimized data transferring with minimum bandwidth usage, flow controlling as well as data encoding/decoding with a sliding window method by ensuring minimum error occurrence. Further, focused on secure sensor data communication using TCP and data layer isolation using an API platform.

Optimized data processing and transmission algorithms of the proposed system are the most important and mostly focused part in this paper. Data processing is done in a separate layer with the ability to integrate it in the cloud or in the hardware module itself according to the preferred architecture. These two architectures are discussed in detail under following sections in this paper. In the presentation layer, scalable and user-friendly web interface (dashboard) is developed to visualize obtained vital signs in real-time which accessible through the internet at anywhere.

2.2. System Architecture

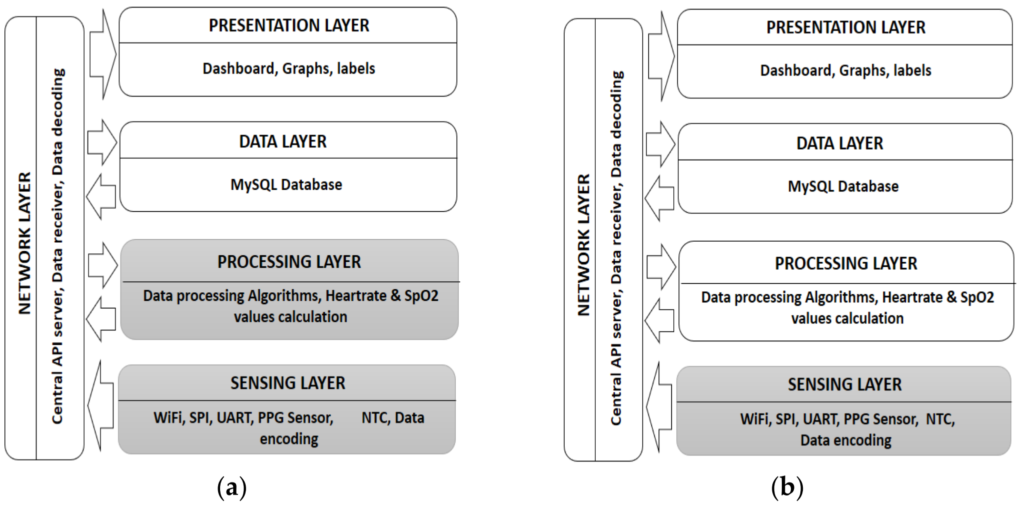

NETWORK LAYER
Central API server, Data receiver, Data decoding
PRESENTATION LAYER
Dashboard, Graphs, labels
DATA LAYER
MySQL Database
PROCESSING LAYER
Data processing Algorithms, Heartrate & SpO2 values calculation
SENSING LAYER
WiFi, SPI, UART, PPG Sensor, NTC, Data encoding
(a)
(b)

Figure 2. System architectures of two proposed methods (hardware layers represented in the dark colors while cloud layers represented by the light colors in the figure): (a) Architecture I - Describes an IoT system with data processing layer on the hardware itself. (b) Architecture II - Describe an IoT system with data processing layer on the cloud.

2.2.1. System Architecture I

In this architecture, as shown in the Figure 2(a), both Sensing layer and the Processing layer are within the hardware module. In this method, all the required measurements (SpO2, Heart Rate) are calculated within the microcontroller (ESP32) unit itself using obtained sensor values. Data from PPG sensor (IR and Red values) is being fed to SpO2 calculation algorithm and heart rate calculation algorithm. SpO2 level calculation is done using valley detection algorithms [5] and heart rate is calculated with FFT. To get a better heart rate value, valley detection algorithm also calculates heart rate again. Body temperature sensor reading, as a 10-bit Analog to Digital Converted (ADC) value is sent without processing. Calculated hart rate values, SpO2 values, and temperature values are sent to the cloud server where Data Receiver, Data Decoder are running. Decoded data then sent to the Data Layer to store in the database via API server running on Network Layer. Stored data can be retrieved whenever needed for the visualization. Then, in the presentation layer, all the measurements are visualized in the dashboard graphically in real-time. In summary this architecture sends and stores only the calculated measurements. Therefore, there can be a loss of other embedded information of the original waveform.

2.2.2. System Architecture II

Unlike in the Architecture I, in this method data does not process within the microcontroller. Instead, it sends raw data obtained from sensors as a byte stream to the Data Receiver in the cloud and stores all the data in the database without loss of any information in the original waveform. Then the data processing is done within the data processing layer which also in the cloud, and visualizes them in the same way as described in the architecture I. Thus, only the sensing layer is located within the hardware module, while all the other layers are in the cloud as shown in the Figure 2(b). Stored readings can be used for further medical studies related to PPG signal analysis as well.

2.3. Bandwith Requirement Analysis

Data transmission is done by the TCP socket server connection, which secures high Quality of Service (QoS). Error handling, Flow controlling are also handled within the TCP protocol. The following analysis discusses the payload optimization that can be obtained with the proposed architecture compared to the common practices. Note that, there we only discuss about the bandwidth requirement for the parameter data of the TCP transmission.

2.3.1. Bandwith Requirement Analysis for Architecture I

In this architecture, all the parameters are calculated within the hardware. Calculated heart rate has an integer value that normally go up to 3-digit number of beats per minute. SpO2 level is also a percentage value with an integer with maximum 3-digits.

Generally, a data frame can be defined with human readable format as follows,

H [3 bytes]	- Heart rate value: (3-digit number)
S [3 bytes]	- SpO2 level: (3-digit number)
T [4 bytes]	- Temperature reading (10-bit value number from 0 to 2^{10}): (4-digit number)
C [1 byte]	- Separation character

$$\text{Data Frame Size}_{(\text{common})} = H + C + S + C + T + C = 3 + 1 + 3 + 1 + 4 + 1 = 13 \text{ bytes} \quad (1)$$

If the database update rate is given by F_{DB}, required bandwidth needed only for the parameters is given by following equation.

$$\text{Required Bandwidth}_{(\text{common})} = F_{DB} \times 13 \text{ bytes /s} = F_{DB} \times 13 \times 8 \text{ bits /s} = 104\ F_{DB} \text{ bits/s} \quad (2)$$

If F_{DB} = 10Hz, then the expected bit rate will be: 1040bits/s (1.015625 kbits/s).

In the proposed architecture I, because of both heart rate and SpO2 values are integers, which can be represented by using only 1 byte. Thus, optimized data frame with proposed data encoder is as follows,

H_1 [8 bits]	- Heart rate value: (1 byte)
S_1 [8 bits]	- SpO2 level (1 byte)
T_1 [10 bits]	- Temperature reading: (10 bits)
FC_1 [6 bits]	- Flow control: (6 bits)

$$\text{Data Frame Size}_{(\text{optimized})} = H_1 + S_1 + T_1 + FC_1 = 8 + 8 + 10 + 6 = 32 \text{ bits} = 4 \text{ bytes} \quad (3)$$

If the database update rate is given by F_{DB}, required bandwidth needed only for the parameters is given by following equation.

$$\text{Required bandwidth}_{(\text{optimized})} = F_{DB} \times 32 \text{ bits /s} = 32\ F_{DB} \text{ bits/s} \quad (4)$$

If F_{DB} = 10Hz, then the expected bit rate only for data transmission is 320 bits/s (0.3125 kbits/s).

According to equation (1) and (3), it is clear that optimized data frame can save up to 9 bytes for each parameter set compared to the common practice which will results in considerable impact on bandwidth, by optimizing payload of TCP transmission packets.

2.3.2. Bandwith Requirement Analysis for Architecture II

In this architecture, raw sensor data send to the Data Receiver. Raw data has two 24-bit readings for each Red and IR LEDs of the PPG sensor. In addition to that 10-bit temperature reading also need be transmitted.

Generally, a data frame can be defined with human readable format as follows,

R [8 bytes] - Red LED value (24-bit value from 0 - 2^{24}): (8-digit number)
I [8 bytes] - IR LED value (24-bit value from 0 - 2^{24}): (8-digit number)
T [4 bytes] - Temperature reading (10-bit value number from 0 to 2^{10}): (4-digit number)
C [1 byte] - Separation character

$$\text{Data Frame Size}_{(\text{common})} = \quad R + C + I + C + T + C = \quad 8 + 1 + 8 + 1 + 4 + 1 = \quad 23 \text{ bytes} \tag{5}$$

If the sensor reading rate is given by F_s, required bandwidth needed only for the parameters is given by following equation.

$$\text{Required Bandwidth}_{(\text{common})} = \quad F_s \times 23 \text{ bytes /s} = \quad F_s \times 23 \times 8 \text{ bits /s} = \quad 184\ F_s \text{ bits/s} \tag{6}$$

If F_s = 400Hz, then the expected bit rate is 73600 bits/s (71.875 kbits/s).

In the proposed Architecture II, IR and Red readings are 24bit numbers which can be represented by 3 byte each. Thus, optimized data frame with proposed data encoder as follows,

R_1 [24 bits] - Red LED value: (24 bits)
I_1 [24 bits] - IR LED value: (24 bits)
T_1 [10 bits] - Temperature reading: (10 bits)
FC_1 [6 bits] - Flow control: (6 bits)

$$\text{Data Frame Size}_{(\text{optimized})} = \quad R_1 + I_1 + T_1 + FC_1 = \quad 24 + 24 + 10 + 6 = \quad 64 \text{ bits} = \quad 8 \text{ bytes} \tag{7}$$

If the sensor reading rate is given by F_s, required bandwidth needed only for the parameters is given by following equation.

$$\text{Required bandwidth}_{(\text{optimized})} = \quad F_s \times 64 \text{ bits/s} = \quad 64\ F_s \text{ bits/s} \tag{8}$$

If F = 400Hz, then the expected bit rate is 25600bits/s (25 kbits/s).

According to equation (5) and (7), optimized data frame can save up to 15 bytes of for each parameter set compared to the common practice which will results in considerable impact on bandwidth, by optimizing payload of TCP transmission packets. This can be a huge advantage for applications which uses high sampling rates, because the bandwidth usage is a function of the sensor sampling rate with a proportional relationship.

2.4. Data Encoder

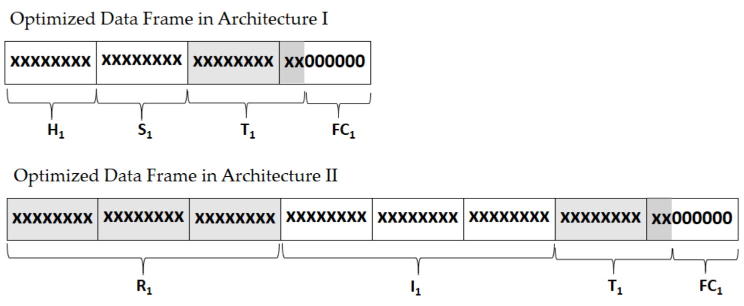
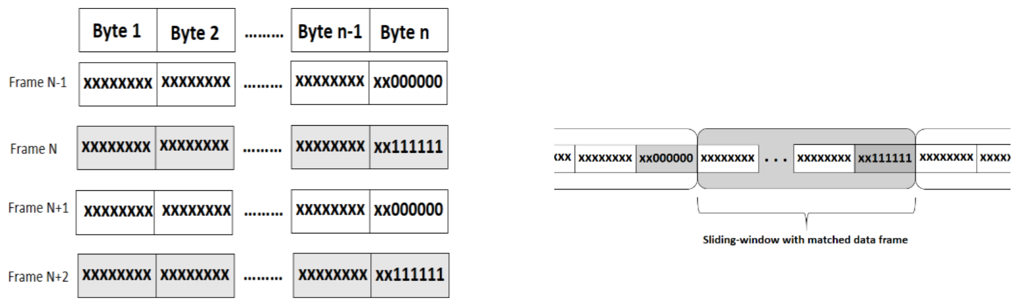


Figure 3. Optimized data frames after encoding.

Depending on the architecture used, representation of each byte of the data frame is different. The structures of the encoded data frames for each architecture are shown in the Figure 3. When it comes to flow controlling, the last 6 bits were used in such a manner that can differentiate each frame for the sliding window method in the data decoder.

This proposed encoding method use last 6 bits of each frame to represents 000000 and 111111 bit patterns to have the maximum difference between flow control values of the consecutive data frames. This method minimizes the error rate due to the facts that two consecutive sensor reading values cannot be get very large variation and the flipping pattern of the flow controlling bits. For example, in the architecture II, data frame size is 64 bits. if Nth data frame carries 111111 bit pattern as the last six bits of the data frame, the (N+1)th data frame carries 000000 bit pattern as ending bits as shown in the Figure 4(a).



(**a**) (**b**)

Figure 4. Structure of data frames used for the optimized data transmission: (**a**) Encoded data frames of n number of bytes with flow controlling bit patterns of 000000 and 111111 at last 6 bits; (**b**) sliding- window of size n with matched data frame.

2.5. Data Receiver

The Data Receiver is a TCP server located in the cloud that is listening for the data from the hardware device. For each architecture, different data receivers are used according to the size and the structure of the receiving data frames. Data receivers are responsible for four different tasks.

I) Receiving data from hardware module via TCP socket.

II) Decoding data using windowed method using Data Decoder.

III) Calculating SpO2 and HR values.

IV) Send data to the Data layer.

This uses the Central API calls for task I and IV.

2.6. Data Decoder

Data Decoder uses sliding-window method with optimization on receiving consecutive sensor reading. Every data frame received by the receiver consists of 6 bits for flow controlling at the end of the data frame as shown in the Figure 4(a). This last 6 bits (flow controlling bits) have either 000000 bit pattern or 111111 bit pattern. This flipping six-bit pattern is responsible for avoiding reading data field as flow controlling bits by mistake when sliding window method is being used. Sliding window method uses fixed window size which equals to the data frame size in each architecture. Use of flow controlling bits for decoding the receiving data frames explained in below.

The length of the sliding window is 8 bytes for architecture I and 4 bytes for architecture II. Once the data stream is receiving, the algorithm executes following steps,

Step 1: Window is shifting over the received data buffer one byte at a time until a matching combination is occurred (flipped bits of the current window's flow controlling bits should appear in the following windows' ending bits). This is shown in the Figure 4(b). When a match is occurred, algorithm calls Step 2 with the proceeding data window.

Step 2: Parameters are calculated using the data within the window. Then goes to Step 3.

Step 3: The algorithm will shift along the data buffer from single window size and validate every time with ending bits of each window to follow the expected bit pattern. If there is a mismatch occurs with flow controlling bit patterns, the algorithm goes back to Step 1. Otherwise, algorithm goes to Step 2 again.

This method guaranteed reliable delivery of the data as well as ensures the data is delivered in order.

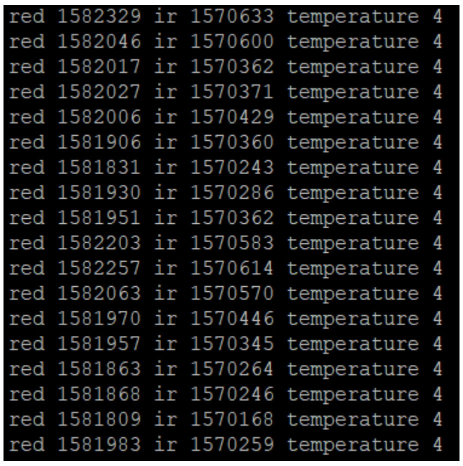
2.7. Central API

Central API is a RESTful API that consists of globally accessible functions to interact with the database. Although this Central API is globally accessible, it also handles the authentication with JSON Web Token (JWT) to limit the accessibility to ensure the network layer security. This API service consists of full functionalities including create, read, update and delete (CRUD) operations for data records and one special function to retrieve the last n number of records for visualization purposes. All the communication between database and the other layers are done only through the central APIs which ensures the security of the data layer. Moreover, changes done for the hardware layer (IoT Device) or presentation layer (Dashboard) will not affect to each other or any middle layers.

3. Results and Discussion

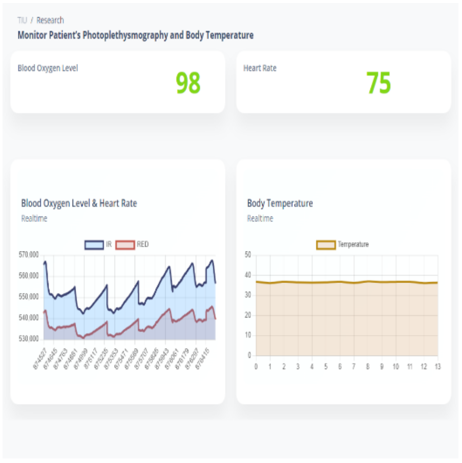
We have created the described system and tested for both design architecture I and design architecture II. We will focus more on the results of design architecture II here because unlike architecture I, architecture II has to use more bandwidth due to raw data format, real-time transmission, and its high sampling frequency. Data receiver handled continuous data streaming over several hours of testing without any connection failure. Which ensures capabilities of the data encoder running on hardware device and the data decoder running on the cloud server. Decoded message shown in Figure 5 (a). We have run the real-time data processing on the cloud to calculate heart rate and SpO2 level before visualizing it on the web dashboard (Fig 5 (b)).

MySQL database and Node.js central API server worked ideally and enable responsive dashboard to visualize all the acquired vital signs graphically using real-time updating of graphs and labels in a user-friendly way. This allows the user (medical officer) to check patients with minimum time and accurate data.



```
red 1582329 ir 1570633 temperature 4
red 1582046 ir 1570600 temperature 4
red 1582017 ir 1570362 temperature 4
red 1582027 ir 1570371 temperature 4
red 1582006 ir 1570429 temperature 4
red 1581906 ir 1570360 temperature 4
red 1581831 ir 1570243 temperature 4
red 1581930 ir 1570286 temperature 4
red 1581951 ir 1570362 temperature 4
red 1582203 ir 1570583 temperature 4
red 1582257 ir 1570614 temperature 4
red 1582063 ir 1570570 temperature 4
red 1581970 ir 1570446 temperature 4
red 1581957 ir 1570345 temperature 4
red 1581863 ir 1570264 temperature 4
red 1581868 ir 1570246 temperature 4
red 1581809 ir 1570168 temperature 4
red 1581983 ir 1570259 temperature 4
```

(**a**)



(**b**)

Figure 5. Decoded data and data visualization: (**a**) Decoded data frames in the decoder before it sends to the database for visualization; (**b**) Web Dashboard: SpO2 and Heart Rate values are displayed in the labels on the top left. Body Temperature values displayed in a different graph in the bottom left while IR and Red values are displayed in the same graph at the bottom left corner.

4. Conclusions

The proposed wireless SpO2, Heart Rate and Body Temperature monitoring system has been implemented and tested. Bandwidth optimization algorithm plays the key role in this system enabling this system to be performed as expected in case of the data transmission. Even when there is a bandwidth limitation, this optimization on data frame sizes will result in high data throughput on transmission according to the Little's law [6].

The proposed system allows monitoring real-time SpO2, Heart Rate and Body Temperature of a patient from a remote location without requiring the physician to take the measurements. Proposed bandwidth optimization algorithm enables these two-design architectures to be cost-effective long-time usage than the exiting general methods of sending data in other formats. While the proposed layered architecture enables the system to be scalable and adaptive to the future needs.

References

1. Shola Usha Rani, Antony Ignatious, Bhava Vyasa Hari, Balavishnu V J. Iot Patient Health Monitoring System. Indian Journal of Public Health Research & Development, October-December 2017, Vol.8, No. 4, in 2017
2. Adan Torralba Ayance, Hector Santiago Ramırez, Jose Miguel Rocha Perez, Carlos Gerardo Trevino Palacios. Wireless Heart Rate and Oxygen Saturation Monitor. XV Mexican Symposium on Medical Physics, AIP Conf. Proc. 2090, 040010-1–040010-4, in 2009.
3. Cristian Rotariu, Vasile Manta. Wireless system for remote monitoring of oxygen saturation and heart rate. Federated Conference on Computer Science and Information Systems (FedCSIS), Wroclaw, Poland, in 2012.
4. Ju Geon Pak1, Kee Hyun Park. Advanced Pulse Oximetry System for Remote Monitoring and Management. Journal of Biomedicine and Biotechnology Volume 2012, Article ID 930582, in 2012.
5. Hee-Kyo Joeng, Kwang-Keun Kim, Sun-Chul Hwang, Myoung-Ho Lee. A new algorithm for P-wave detection in the ECG signal. Images of the Twenty-First Century. Proceedings of the Annual International Engineering in Medicine and Biology Society, Seattle, WA, USA, in 1989.
6. J. D. Little, A proof for the queuing formula: L= λ w, Operations research, vol. 9, no. 3, pp. 383–387, in 1961.