A Scalable Framework for Healthcare Monitoring Application using the Internet of Medical Things

Venki Balasubramanian
School of Science, Information Technology and Engineering,
Federation University,
Ballarat, Victoria, Australia.
v.balasubramanian@federation.edu.au

Alireza Jolfaei
Department of Computing
Macquarie University
Sydney, NSW, Australia
alireza.jolfaei@mq.edu.au

Abstract

Internet of Things (IoT) is finding application in many areas, particularly in healthcare where an IoT can be effectively used in the form of an Internet of Medical Things (IoMT) to monitor the patients remotely. The quality of life of the patients and healthcare outcomes can be improved with the deployment of an IoMT because health care professionals can monitor conditions; access the electronic medical records and communicates with each other. This remote monitoring and consultations might reduce the traditional stressful and costly exercise of frequent hospitalisation. Also, the rising costs of healthcare in many developed countries have influenced the introduction of the Healthcare Monitoring Application (HMA) to their existing healthcare practices. To materialise the HMA concepts for successful deployment for civilian and commercial use with ease, application developers can benefit from a generic, scalable framework that provides significant components for building an HMA. In this chapter, a generic maintainable HMA is advanced by amalgamating the advantages of event-driven and the layered architecture. The proposed framework is used to establish an HMA with an end-to-end Assistive Care Loop Framework (ACLF) to provide a real-time alarm and assistance to monitor pregnant women.

1 Introduction

The growth in wireless and mobile communication has given birth to low power, low cost, multifunctional IoT such as wireless sensors. IoT is finding application in many fields, particularly in healthcare [1], this triggered the development of software known as healthcare monitoring applications (HMA) that health care providers use to access data about their patients' current state. Some data is collected directly from patients using IoMT such as Body Area Wearable Sensor Network (BAWSN) comprised of wearable sensors detecting features including heart rate, breath rate, electro-cardiograph, heart rate variability or skin conductivity and transmitting the data to a smartphone that can act as a local processing unit

(LPU). In recent years, the technologies have evolved rapidly, and now it is feasible to remotely monitor the patients either in the hospital or in the patient's residence at a low cost.

Remote patient monitoring can alleviate the emerging health care crisis [29] characterised by global shortages of health care professionals, rapidly increasing rates of chronic illnesses that need continuous monitoring rises costs of health care. In-house healthcare monitoring applications promise the delivery of healthcare to isolated residents having a variety of health conditions that require supervision and continuous monitoring. Also, the hospital infrastructure in rural areas is traditionally inadequate. Owing to these facts, many developing and developed countries have influenced the introduction of HMA into existing healthcare practices [2].

1.1 Potential challenges of Maintainability

Although HMA has been developed in numerous isolated field trials, scaling the applications up to deployments involving large numbers of patients and health care organisations is challenging. One challenge consists in handling data sets that become very large. As data from some of these devices is logged hundreds or thousands of times per second, real-time monitoring over days for many patients associated with a health care organisation leads to massive data sets that indeed challenge the internal data storage or processing capabilities of any single practice. Any significant challenge arises from the need to maintain data privacy, even diverse practitioners from different healthcare organisations seek to access a patient's data. Each HMA developer will be tempted to encode an approach for security and data management that will typically be idiosyncratic; this will lead to a plethora of applications that do not easily interoperate with each other or other systems and become challenging to maintain.

According to [4], maintainability is the ease with which a software system or component can be modified to correct faults, improve performance or other attributes or adapt to a changing environment. Many resources are spent, especially in industry, in producing software applications that are easy to maintain, reusable and able to accommodate any changes in the future, with the potential aim of saving costs. Although many tools, process and procedures were developed to enhance the maintenance of the software application, one cannot control what one cannot measure, and there is yet no universal measure of maintainability [5, 6]. However, it is shown in the literature [5-9] that developed applications that are reusable and able to accommodate changes cost less in maintenance. The business aspect of the maintainability that is cost-effective in maintaining the application is not within the scope of this chapter. Instead, it is assumed that the developed software application, which is easily configurable, reusable and flexible for the application developer, is characterised as a maintainable application. Therefore, this chapter exemplifies the technical aspects to achieve a generic maintainable HMA.

1.2 Architecture-level maintainability

One can broadly classify software application maintenance into two categories (i) architecture-level maintainability, which is based on the ease of modification of the components/functionalities in the software architecture and, (ii) software-level maintainability, which is based on the ease of change of the lines of code (LoC) in the software programs. The discussion is focused on the architecture-level maintainability to assist application developers in understanding the significant components when building an HMA. In general, the concept of modularisation in terms of the objects and the layering in terms of the functionalities is followed substantially in creating a computing application system. The objective when developing these concepts is to attain a high-quality, maintainable application. For architecture-level maintainability, the design of the application framework components should be in such a way that, with ease, the application is adaptable to the required changes and undergo repairs to correct faults and, in the case of modifications, to improve the performance and correct any failures in the existing state of the required efficiency.

In the case of the HMA, the functionality of the Healthcare Application (HA) varies owing on the type of the patient's disease or the clinician's requirements. Also, with the existence of various hardware and software for developing an application, the framework for the HMA should be generic to apply to any HMA development without compromising the required functionalities. The frameworks that have been discussed in [10 – 15] with overlapping features do not follow any defined structure. Moreover, they are dedicated solely to monitoring the patients remotely without any active data that can raise the alarm — the database of the application stores the health data of the patient for future diagnosis only. Also, the data are sent over a 4G cellular network from the user's smartphone to the internet would cause delay, which could be reduced by limiting the data transfer based on the sensed value.

Based on our studies, we established that the event-driven architecture, which is used heavily in the software development industry and has proven to be maintainable [16 – 18], could be correlated with the requirement of the HMA because the sensed values from the sensors could be termed as events in the HMA. Therefore, the initial section in this chapter describes the event-driven model, followed by its association with the HMA's event flow with the ED's event flow. Consequently, the association is used to frame the layered functions of the ACLF, because of the following known advantages of the layered function with respect to the application development in the layered approach, (a) each layer can be an abstraction of the operational view of the application, (b) it has the ability to design and develop each individual layer independent of the others, (c) it has the ability to test and verify the layers

independently, (d) it can fast-track the development, because each layer can be developed concurrently, (e) it has the ability to share the functionalities if needed and, (f) the transfer of knowledge about the features of the complete application can be achieved as layers/stages by the application developer.

2 Event-driven Paradigm

In software architecture, service orientation is a loose coupling of service or a specific task. A requestor, who needs some software service, request the service provider. The service provider is the one who is aware of its implementation while the requestor knows how to place the request [16]. The service orientation forms the basis of the Service-oriented architecture (SOA). The SOA is an architecture strategy for software service infrastructure solution. The event-driven model employs features from the SOA – one or many services will be triggered due to the occurrence of a distinguished event in the system. The triggered services may be a simple or a complex function of the system. The interaction between the event and services is generally referred to as event-driven SOA or event-driven architecture.

2.1 Event-driven architecture

An event may indicate a notable incident, future problem, deviation in reasonable value or from the threshold range. In an event-driven (ED) model, the generated event is instantly distributed to the concerned process that assesses the event, and appropriate action is taken [16]. The generated event may invoke any types of service, trigger a process, publish information or syndicate further action. In a broader event-driven architecture, the service can also be the source for many events. In general, ED architecture is dispersed and not tightly-coupled. The generator doesn't know the event's subsequent processing or the required actions but knows only the source of the event. To 'get a heads-up', it can be seen that an HMA is made up of various events. It starts with the sensing of the health data of the patient, the processing of the data based on the sensed values, that is, normal or abnormal, the triggering of an alert message locally or the updating of the server database and the diagnosis of the patient's condition at the HA. The following paragraphs describe the event flow layers in the ED architecture that can be used for building any HMA, subsequently, our ACLF.

2.2 Event Flow Layers

The most common type of ED architecture is a stream event processing (SEP) [16]. In the SEP, both regular and essential events occur. The regular events, such as the normal health data of the patient in the healthcare monitoring, are filtered for importance and fed to the information subscribers. In order to take in-time decisions, the SEP is generally used to initiate the real-time flow of information within the application system. Figure 1 shows the event flow starting with the event being generated and culminating with the execution of any

downstream (event-driven) activity in an HMA. However, the description is generalised for two reasons. First, for the readers to correlate the ED model with the HMA abstractly. Secondly, the features of the ED model are not reproduced as such in the HMA. The event flows in the ED model are segregated into four logical layers, as follows:

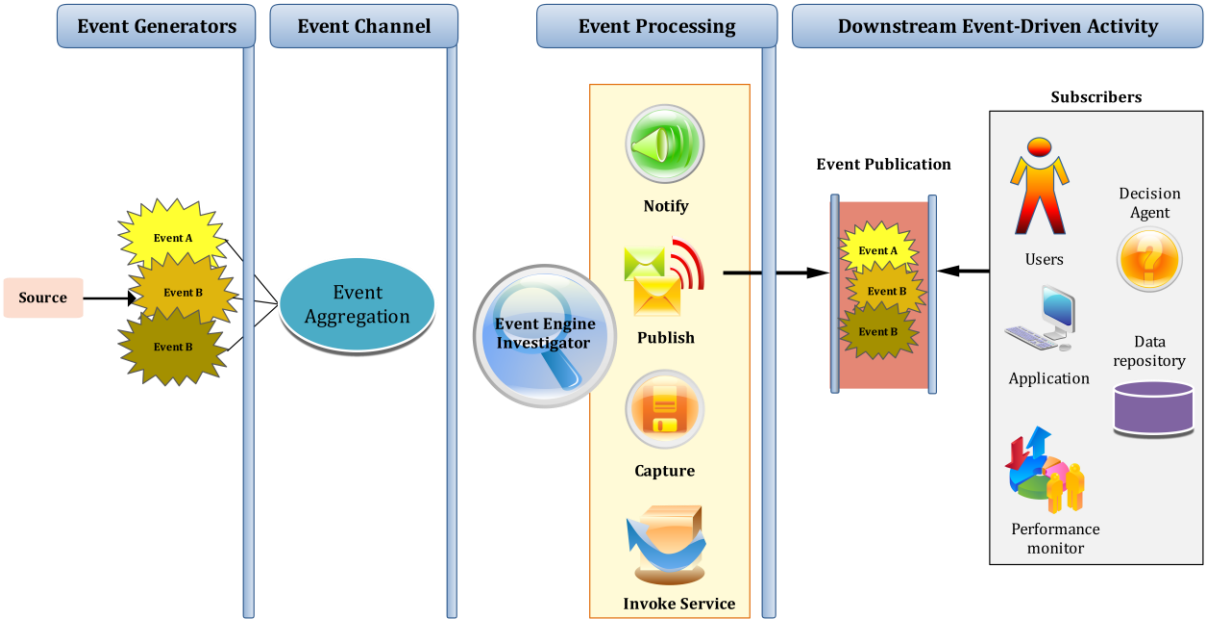


Figure 1 Event Processing Flows in Event-Driven Architecture

1. ***Event Generators*.** The source generates every event. The source might be a sensor, an application, transmitter, system process or a collaboration tool such as an instant messenger or email.
2. ***Event Channel*.** The messaging backbone is an event channel that conveyances regular clarified events between the event generators, event processing engines, and downstream subscribers.
3. ***Event Processing*.** The event-processing layer evaluates the events upon the receipt of the event from the event channel. The requirements of the application define the rules and actions for the event-processing. Based on the specified rule, the event-processing layer can induce response such as notifying the user, publish the event description or invoke the required service.
4. ***Downstream event-driven activity*.** The downstream activities are initiated either by a single event or event correlation. The downstream operations are invoked either by push or pull by the subscribers of the event publication.

The above-described functionalities of the four layers in the ED model can be employed in constructing the layered functions of the ACLF. The following section describes the features of each component in the ACLF and their mapping with the logical layers in the ED model.

3 ACLF Architecture

The ACLF is constructed with the primary focus of creating a framework that can exploit the IoT infrastructure in the healthcare field to cope with the intricate needs of the aged or patients. The design of the framework is not only used to send the alarm message when the

patient's health condition deteriorate, but it also satisfies the complex needs of the existing healthcare system requirement as mentioned in the paragraph below.

Paper-based patient-held health records are still primarily used in most hospitals in Australia. Most importantly, there is no interconnection between the patient's records between hospitals, unless the records are maintained by the patients. These non-standardized records with the known disadvantages (such as fragility, occupy more space and maintenance) of using paper-based records can be overcome by establishing an electronic record, such as the electronic Medical Record (eMR) in the Assistive Maternity Care (AMC) application [28]. Inherently, electronic records about the patients can be interlinked easily, which can enhance the existing clinical diagnosis and avoid the costs involved in repeated clinical examinations. Other facilities, for instance, the availability of the recommended specialist for the related disease and the collaboration of the doctors (general practitioner and specialist) about the patient's condition, can be achieved 'on the fly'. Using online forums, the patients can discuss their health with their fellow patients and also with their doctors. Because the design and implementation of the ACLF can achieve the above needs in the existing healthcare system, it would pave the way for deploying a civilian and commercial state-of-the-art HMA in the coming years. The following section describes the general and layer functions of the ACLF.

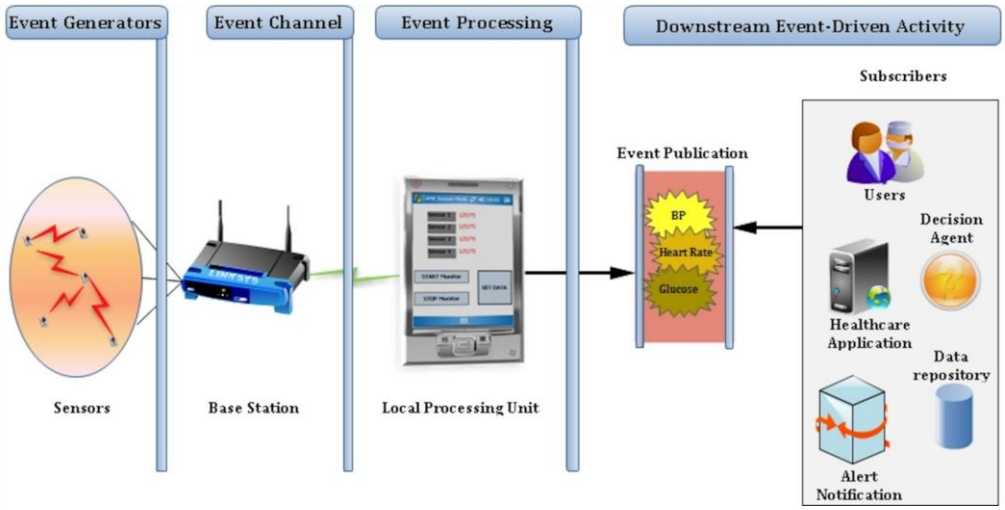
3.1 Functioning of ACLF

Based on the needs of the clinician's requirement or disease of the patient, the health condition of the patients or aged are monitored using the wearable sensors. The wearable sensors can be multi-modal and can monitor vital signs of the patient such as Blood Pressure, SpO2 (oxygen saturation), Respiratory Rate, Heart rate and ECG at once. The sensors collect the vital signs data and send the aggregated data to the smartphone for pre-processing. In ACLF, the aggregated data are pulled using a smartphone-controlled by the patient, where the data is pre-possessed and saved in the application database, thus updating and/or creating an electronic record of the patient via the amalgamations of the various networking protocols. Additionally, subject to the seriousness of the patient's vital signs, the database notifies the Health Server System (HSS), which in turn sends an alert message with the health condition of the patient to the clinicians using either a short messaging service (SMS) or email. The functions of the ACLF are layered to let the vital signs value to transfer from one discrete stage of processing to another. The movement of vital signs value provides control not only for the user but also for the application developer to exploit the framework with the various available technologies. The ACLF layer functionalities are described below.

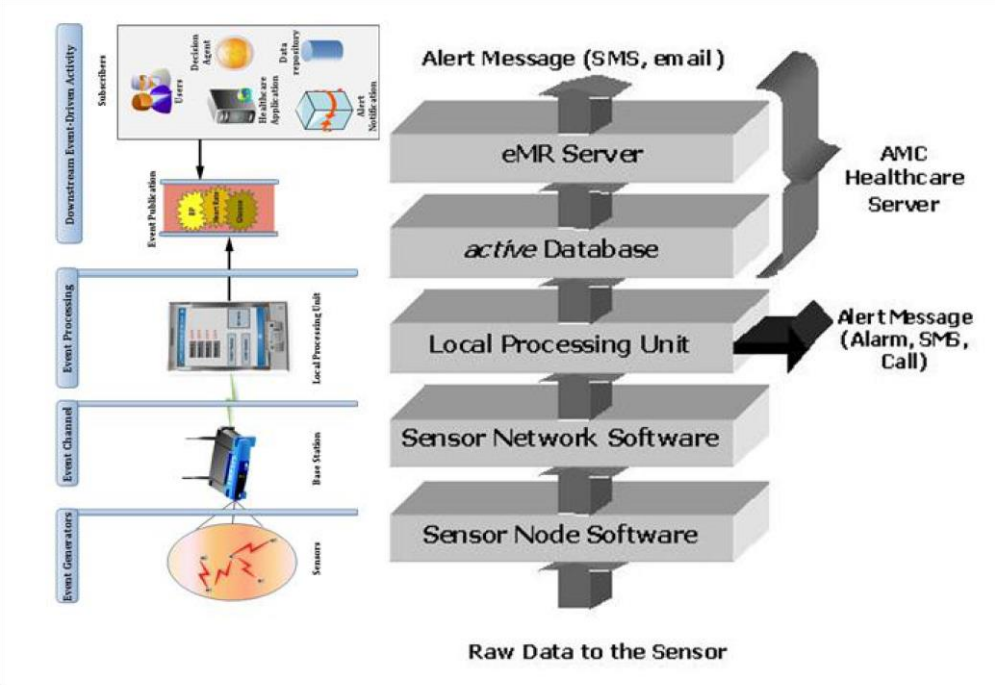
3.2 ACLF Layer Functions

The use of IoMT to monitor the patients remotely and continuously is clinically significant and can save lives [10–12, 19–21]. The IoMT based monitoring application is built using multi-modal wearable sensors and smartphone for a given Healthcare Application (HA) [22]. The advanced framework encompasses our previous work on the AMC [28], and it can be functional for any application that uses the IoMT with wearable sensors. Based on the requirements for the next generation (nextG) healthcare system, we layered our ACLF into five levels, as shown in Figure 2(b) (i) Sensor Node Software (SNoS), (ii) Sensor Network Software (SNeS), (iii) Local Processing Unit (LPU), (iv) Active Database (AcDa) System and, (v) eHR System. The functionalities of each layer are described below.

Sensor Node Software: The SNoS are programmed within the wearable sensors. Since the sensors are multi-modal and capable of transmitting different health data of the patients such as Respiratory Rate, ECG, Blood Pressure, Heart Rate and SpO2, the SNoS aggregates the sensor data. When generating the event (data) in the ACLF, the functions of the SNoS are programmed into the sensor depending on the necessity of the monitoring application, such as the size of the data (or packet size), packet structure; secure key for sensor node identification and the timing instruction to the sensor to sense/send data.



(a)



(b)

Figure 2 (a) Event-Driven Architecture for Healthcare Monitoring Application (b) Comparison of Assistive Care Loop Framework Layers with Event-Driven Architecture

Sensor Network Software: The SNeS is programmed in the base station to obtain all the health date of the patient from the sensors on the body of the patient. The base station acts as the event (data) channel between the sensors and the LPU. The primary function of the SNeS is to aggregate the data from the sensors and send it to LPU upon its request. Apart from this, the SNeS can also be programmed to authenticate the sensors.

***Local Processing Unit*.** The LPU is the event (data) processor in the ACLF. Therefore, the application in the LPU should be capable of receiving the authenticated sensor data through the base station. Recall that, in the ED model, the event processor initiates the actions and the processing rules upon the receipt of the event, and the application provider sets the rules and the action rather than the event generators. In the same manner, the Local Processing Unit in the ACLF pre-process the received data from the wearable sensors and sends only those vital data that are acute and sensible to the application database. The LPU also checks the gravity of the received data, in case of any critically alarming vital signs the LPU is programmed to raise the alarm to the clinicians for timely assistance for the patients. The primary role of the LPU is to create an eMR in the database and acts as a bridge between the wearable sensors and application database.

***Active Database*.** The Assistive Maternity Care (AMC) Application consists of a database, a web-driven application known as HSS, the client system and the communication infrastructure. The web-based HSS application is made up of an eMR server (application logic), an SMS gateway to send alert messages and an *active* database. The database in the AMC application is termed as an *active* database because it contains specific or selected data value that is clinically significant and needs continuously monitoring, any changes in the specified or selected data value should be notified to the system. The type of database that contains active data values are known as the Active Database (AcDa). The requirement of the specific and selected *active* data value depends on the need of the application developer as well as the clinical significance. In our AMC application, the clinically significant value is blood pressure (BP) of the pregnant woman.

***eMR Server*.** The eMR server is used to monitor the BP of the pregnant woman. However, the eMR server is the application logic in the AMC application can be used for any IoMT based application that uses wearable sensors. The application logic in the eMR server needs to be updated with the new features based on the type of IoMT application. There are many types of IoMT applications, such as knee surgery recovery, elderly homecare and cognitive disorders that need continuous monitoring [15]. The eMR server that connects to an email and an SMS gateway can send sophisticated alert messages to the nurses and the doctors. The eMR server can be updated to send more information, including ECG recording and a short report of the patient condition. The LPU in the AMC application is only used as a pre-processing unit or an edge device, while the eMR server is powerful and capable of handling processes intensive tasks such as ECG and patients reports.

The combined functionalities of the AcDa and eMR system layers form a complete HA. In turn, their functionalities also create the downstream event-driven activity for the ED model. It can be noted that some of the statements about the functionalities of the AcDa and eMR systems are biased towards our AMC application rather than generically towards our framework. The purpose of those statements is for the readers to understand the basic functioning of these layers that might also be common for other HA*s*. Moreover, the reader would gain further understanding of those layers' functionalities upon reading the detailed implementation of our ACLF.

4 Implementation of the ACLF

Our ACLF, as shown in Figure 3, is implemented using different types of components and features. The functionality and features of the components in the ACLF differ in their storage capacity, processing power in a vastly diverse platform. As a research-oriented development, this framework uses open source software wherever possible.

The ACLF consists of a BAWSN, Health Sever System, alert notification logic and SMS Gateway as a framework component. The details of the framework components are given in the following sections.

The sections below are arranged by the flow of patient's health data from the wearable sensor to the SMS gateway, that is, the end-to-end process in the ACLF. In so doing, it is necessary to exemplify the components in the BAWSN, the monitoring part followed by the components in the AMC application, the HA. Also, the arranged section can be correlated with the above-mentioned ACLF layers.

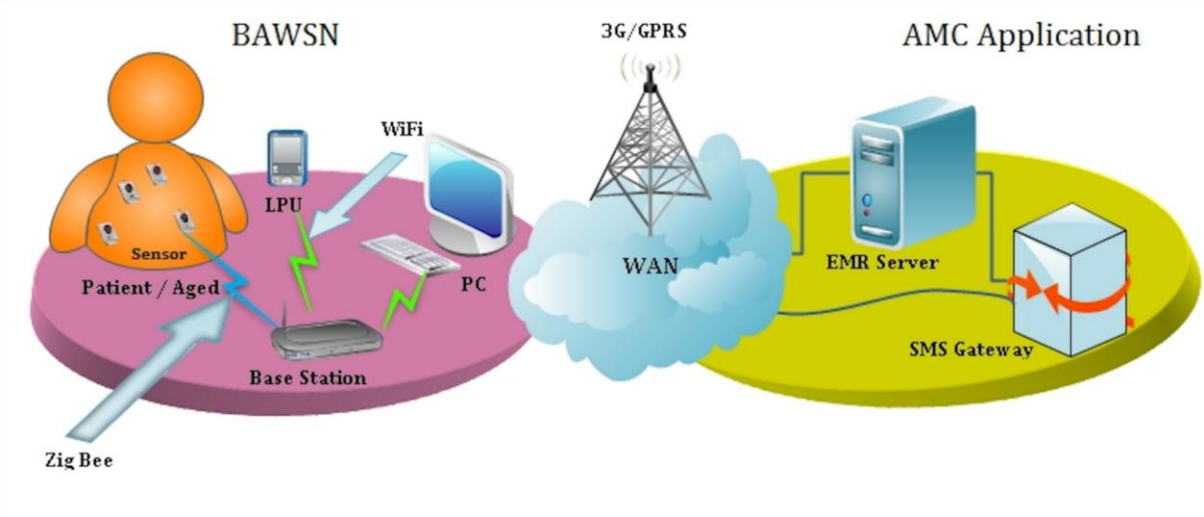


Figure 3 Assistive Care Loop Framework

4.1 BAWSN

The BAWSN is made up of three different types of devices, sensors, base station and the smartphone (see Figure 4). The processing power, storage capability, software platforms and the communication protocols used by these devices vary considerably. The IoMT in ACLF application consists of TinyOS based wearable sensors. The wearable sensors used in the ACLF application consists of mote sensors made using crossbow-based technology [23, 24]. The wearable sensors form a body area network connected to the Stargate base station, which in turn connects with the smartphone that acts as gateway to internet. The Stargate base station is used to connect the ZigBee based mote sensors with the IEEE 802.11 based smartphones.

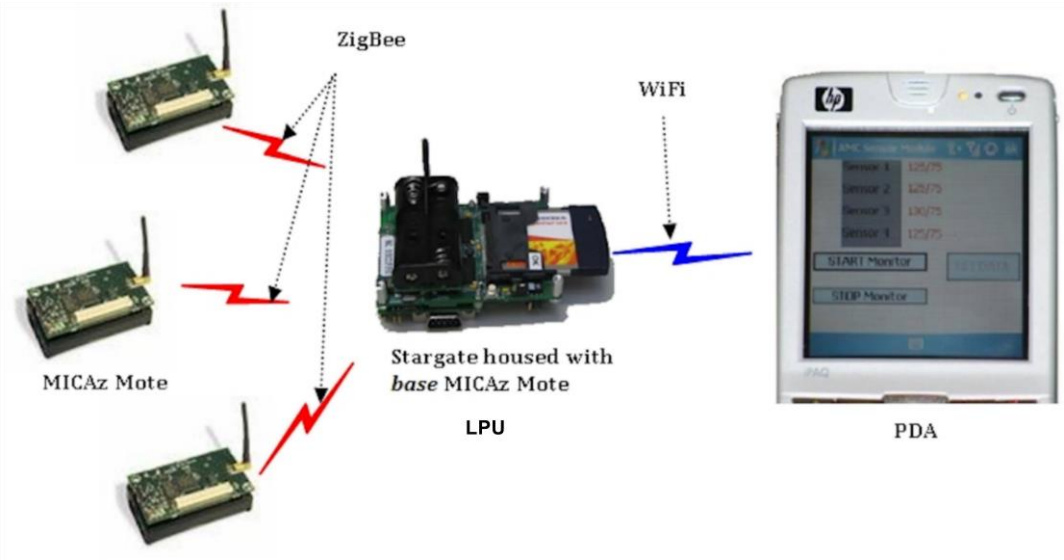
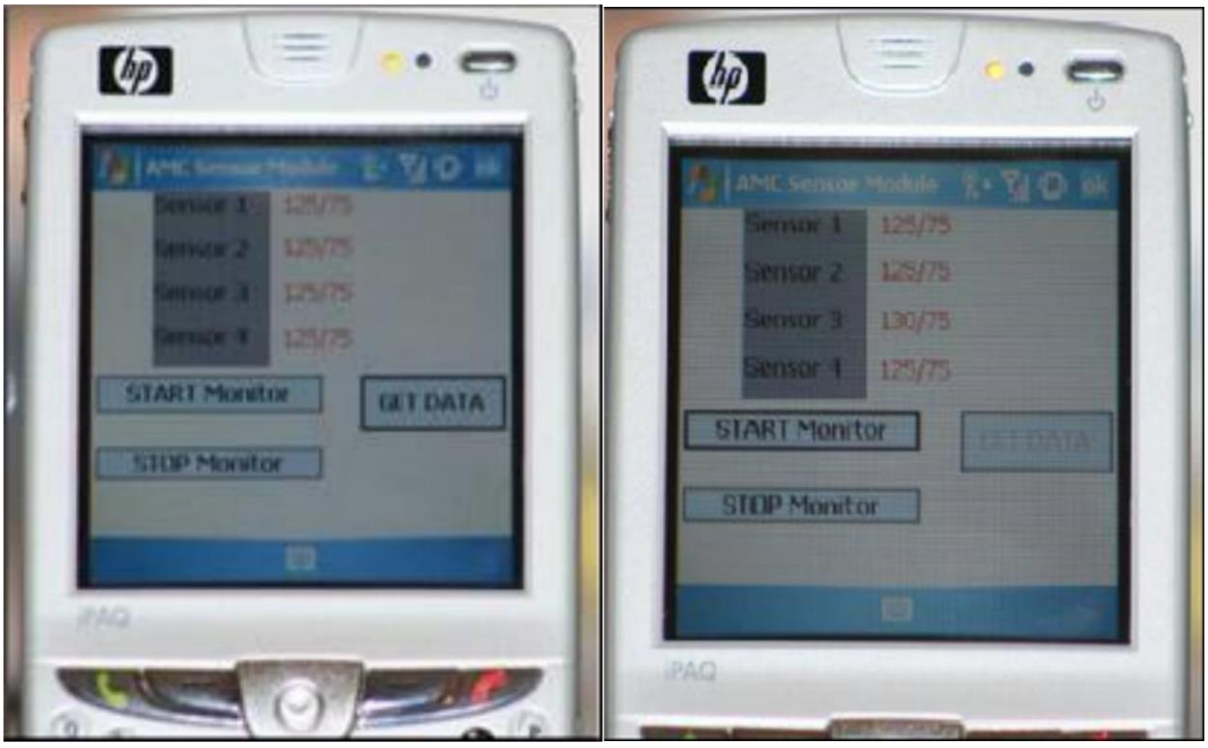


Figure 4 Implemented Hardware Components in Body Area Wearable Sensor Network

The mobile application *Obi-MATE 2*, as shown in Figure 5, is used to control the monitoring of the wearable sensors. The user can use *Obi-MATE 2* to start/stop monitoring (see Figure 5b), while the application uses the Simple Object Access Protocol (SOAP) [27] to integrate the *Obi-MATE 2* with the AMC application.



(a) (b)

Figure 5 The Graphical User Interface for the Obi-MATE 2 (a) continuous monitoring, (b) to get current data

4.2 Health Server System

The HSS in the AMC application consists of an eMR sever, an SMS gateway and a database, which contains active data that is critical to be monitored. The eMR server that has the actual implementation of the eMR application provides all the functionalities for the maternity care with two primary interfaces, the nurse interface and the pregnant woman interface. The nurse interface consists of features that form the major of the system. In order to take care of pregnant women the nurse will register them as midwife in the HSS. After registration the nurse create the electronic Medical Record of a particular pregnant woman. The created eMR can be used to view the progress of the pregnancy. The registered pregnant woman can view and modify the details using the pregnant woman interface. The users, pregnant woman, nurse and doctors, of the HSS can access the same set of screens. However, the pregnant woman has limited access rights to particular data fields and screens.

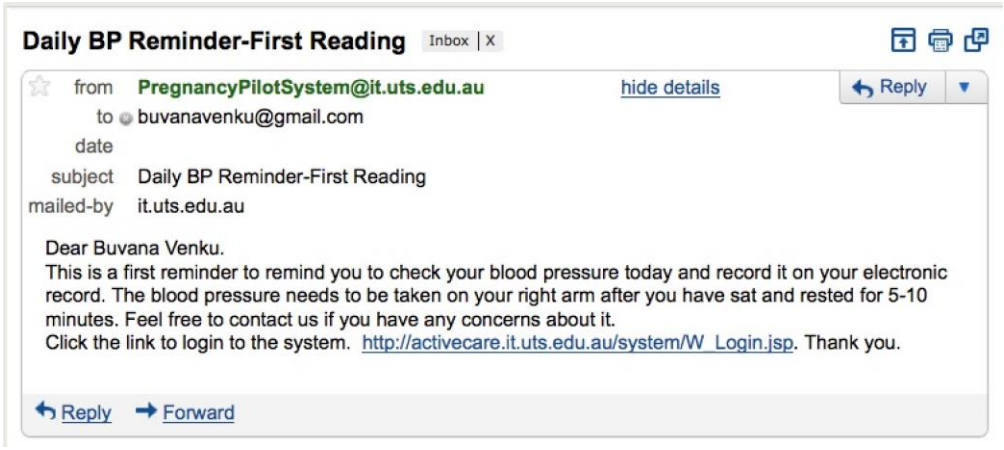
The eMR application is developed using the Model, View and Controller (MVC) design pattern [28]. The perfect MVC is achieved by using Java Beans, Java Server Pages and Java Servlet for the Model, View and Controller respectively. In detail, the controller is deployed using a Java Servlet. In the eMR application, every functional request goes through the controller that retrieves the essential model object by using Java Beans. The controller gets the output back from the model object and forwards a redirect request to the Java Server Page (JSP) view.

The developed eMR application is placed under an Apache Tomcat, the web application container, for execution. A real-time-like application was built using Internet Information Services (IIS) 6.0 web server. A focus group of nurses and pregnant women was conducted to demonstrate the application [21, 28]. The visit[1] table that contains BP (Systolic and Diastolic) value of the pregnant woman is used as an active data in the database.

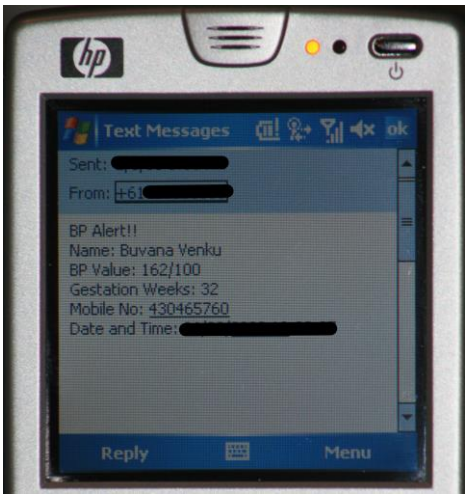
4.2.1 Alert Notification and SMS Gateway

The application uses two types of alert notification – by SMS or by email. The type of notification is selected based on the gravity of the situation. For any non-critical case, an email notification is sent to the nurses and the pregnant women, while the SMS notification is sent if the gravity of the case is critical. The application provides an opportunity for the users to choose their preference for receiving the notification. One of the advantages of sending alerts through email is that one can attach additional medical information and images to the alert. However, alerts through email may not be very useful unless the users check their email box frequently. In case of any attachment is needed, the eMR application sends the attachment using an email – an example of remainder email for the pregnant woman is shown in Figure 6. In the case of mobility, the best effective medium for sending alerts notification is SMS. An alert message sent via SMS must be short and concise. If further details of a patient's condition are required, the clinicians need to access the system. A telephone conversation may follow between the pregnant woman and her clinician if desired.

[1] The visit table in the AMC application records the regular check-up information of the pregnant woman's visit.



(a)



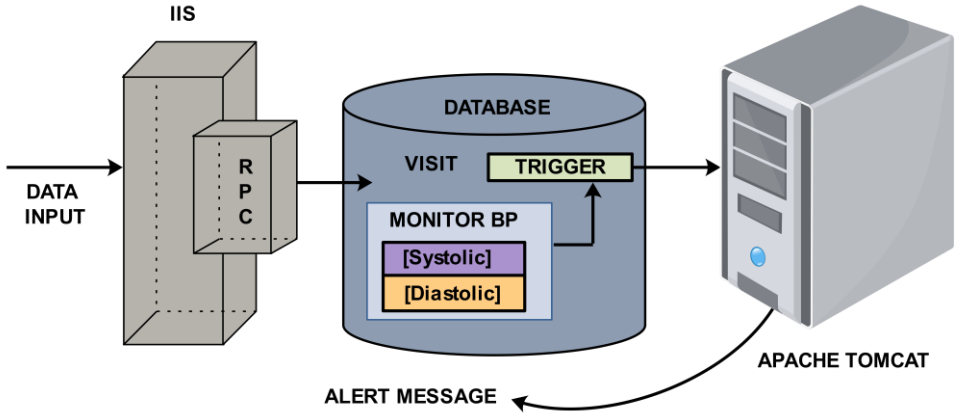
(b)

Figure 6 Alert Message using (a) notification email (b) an SMS

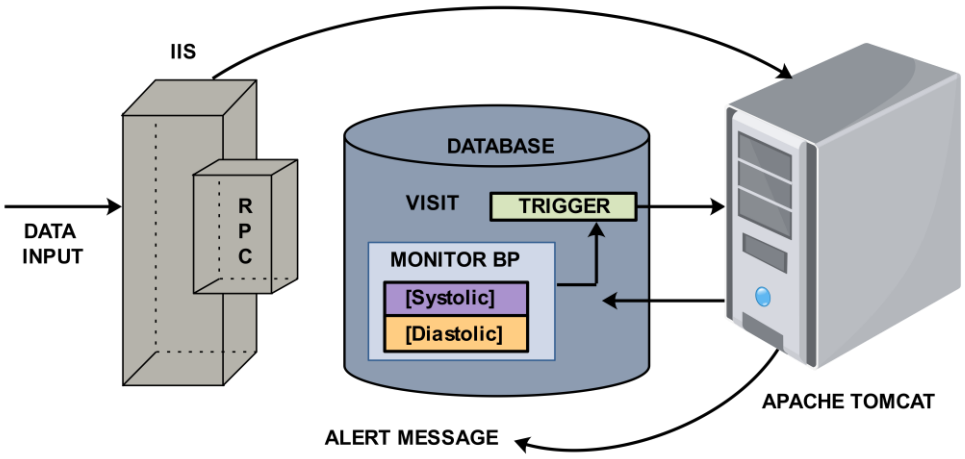
For sending an SMS, an SMS gateway is established by connecting a GSM modem to the eMR server using the serial communication port. A stand-alone Java program is used to send the SMS by using the AT communication commands – the Java program is triggered by a Java Servlet from the eMR application. Figure 6(b) shows an example of the output for sending the SMS. In sending the SMS, the gravity of the situation is taken care by sending SMS to all the nominated clinicians for that particular pregnant woman.

4.2.2 Mode of Monitoring

In our implementation, we use a trigger for monitoring the BP. A trigger that checks the systolic and diastolic values is created in the database. The mode of monitoring in AMC application can be active or passive. In active mode, the application evaluates the monitored data, if the gravity of the stored BP reading is abnormal an alert message is sent to the clinicians or to the woman advising her to pursue a follow-up procedure. In passive mode, the entered BP data by the pregnant woman will send an alert message to the clinicians to summon assistance.



(a)



(b)

Figure 7 Assistive Maternity Care application behaviour upon receiving Blood Pressure value through, (a) wearable sensor (b) manual entry

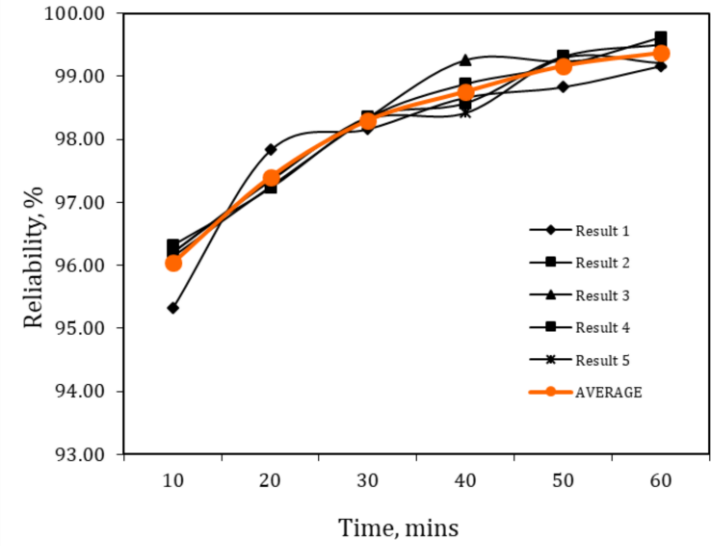
When the pregnant woman's health data are entered into the AMC application database, an event is generated. The generated event will invoke a process in the application logic that will evaluate the BP values - both systolic and diastolic. If the BP value is within the prescribed threshold for the pregnant woman, the value is stored in the eMR of the pregnant woman. If it is not within the reasonable limit an alert is raised to the clinicians or the midwives. The alert will enable the clinicians to take further action on the pregnant woman.

We used a stored procedure and trigger in the MySQL database for monitoring the parameters; this trigger will induce the relevant Java Servlet program in the eMR application and, in turn, this will induce the stand-alone Java program. The web interface is used to enter the values mainly using the JSP that updates the database through the Java Servlets, as shown in Figure 7(b). However, in our ACLF with the BAWSN, for the monitoring part, the pregnant women are not required to enter the data manually. Each can use the *Obi-MATE 2* from her smartphone to collect the BP data from the sensor and send it to the AMC application automatically. When the AMC application receives the data from the wearable sensors via smartphone, it executes a remote procedure call (RPC) through web method that is housed in

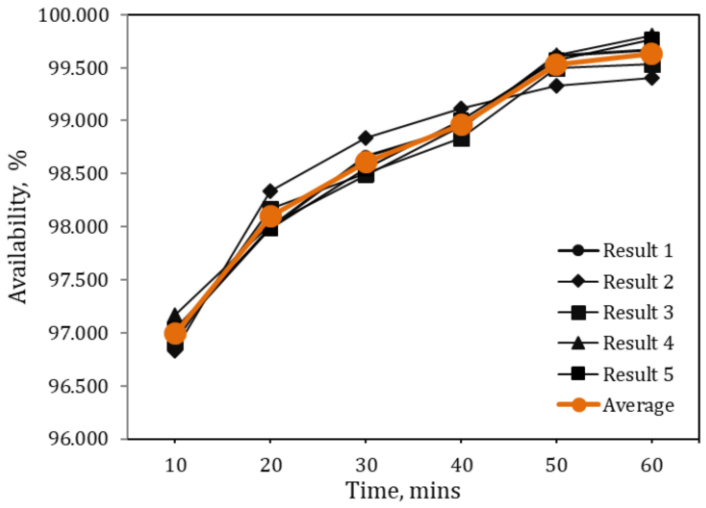
the IIS to update the active database, as shown in Figure 7(a). The database triggers the eMR server to send an alert message depending on the gravity of data. In either scenario, the database executes a trigger that then notifies the application to send the alert message, if necessary.

5 Performance Evaluation of ACLF

The main focus of this chapter is to emphasise the necessity of maintainability for IoT-based applications. One can realise the property of maintainability only by assessing their flexibility in the design during its development and their adaptability to the system during its real-time deployment because specific properties of the dependability such as maintainability and safety cannot be measured numerically [30]. However, one can quantify the performance of the maintainable application by using the other properties of the dependability, such as availability and reliability of the application [30].



(a)



(b)

Figure 8 Performance of Operation Properties in Assistive Care Loop Framework (a) Reliability and (b) Availability

The performance evaluation of the IoT-based applications, in our case, the implemented ACLF, by conducting both the real-time test-bed experiment and the simulation. The availability and the reliability of the IoT application are measured using the non-device-centric formulation such as critical time and space [30]. Technically, the frequency and the accuracy of the essential health data needs of the continuous HMA determine the reliability and availability. The experiments for the operational properties in the IoT are conducted based on the arbitrary frequency for a continuous HMA. The IoT operates most of the time as an in-house healthcare monitoring system, mainly using the lightweight security scheme [31].

The Figure 4 shows the deployment of the IoT-based application. The data packets in the experiments were unencrypted in the initial phase to study the impact of the operational properties of the IoT-based application. The duration of each experiment was 60 minutes, and a series of five experiments were carried out to find out the average of the operational properties such as reliability and availability. The Figure 8 shows the average of the operational properties for every 10-minute interval within the actual duration of the experiment.

6 Conclusion

The maintainability of the Healthcare Monitoring Application was achieved by harnessing the advantage of using a layered architecture by amalgamating the Event-Driven architecture. The amalgamated architecture facilitated the development in various ways, such as in the initial software development for the Assistive Maternity Care application, in extending the same application to have monitoring components, for testing each and every part in the ACLF separately, to fix various bugs in different components synchronously, to resolving any failures during the mode of operation and, most importantly, to identify the failures more quickly. For instance, if anyone of the sensors runs out of battery power, it will be determined by the smartphone monitoring software more or less immediately, and also in the case of a base station failure. The framework was also used to develop more sophisticated functionalities in the smartphone, such as generating log files of sensed data and artistic user interfaces with the pregnant women and their clinician details for the *Obi-MATE* 2, without modifying any of the other layers in the framework. Finally, the generic framework was sufficiently flexible to harness the newly developed generic Healthcare Application, the Health Server System that can be used to emulate a cloud-based healthcare framework in future with ease.

References

1. Mahmoud MM, Rodrigues JJ, and Saleem K. Cloud of Things for Healthcare: A Survey from Energy Efficiency Perspective. In 2019 International Conference on Computer and Information Sciences (ICCIS), pp. 1-7. IEEE, April, 2019.
2. Islam, Riazul SM. The internet of things for health care: a comprehensive survey. IEEE Access 3, 678-708, 2015.
3. IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.12- 1990, IEEE, pp.1, 1990
4. Land R. Measurements of Software Maintainability, In Proceedings of ARTES Graduate Student Conference, ARTES, 2002.
5. Land R. Software Deterioration and Maintainability – A Model Proposal, Second Conference on Software Engineering Research and Practice, Sweden, 2002.
6. Gilb T. Designing Maintainability in Software Engineering: a Qualified Approach, Result Planning Limited, INCOSE, 2008.

7. Fernley EH, Design Metrics as an Aid to Software Maintenance: An Empirical Study, Journal of Software Maintenance: Research and Practice, vol.11, issue. 1, pp. 55-72, January.1999.
8. Jaktman CB, Leaney J, Liu M. Structural Analysis of the Software Architecture - A Maintenance Assessment Case Study, In Proceedings of The First Working IFIP Conference on Software Architecture (WICSA1), Kluwer Academic Publishers, 1999.
9. Jovanov E, Raskovic D, Price J, Chapman J, Moore A, Krishnamurthy A. Patient monitoring using personal area networks of wireless intelligent sensors, Biomedical Sciences Instrumentation, vol. 37, pp. 373–378, 2001.
10. Wilson LS, Gill RW, Sharp IF, Joseph J, Heitmann SA, Chen CF, Dadd MJ, Kajan A, Collings AF, Gunaratnam M. Building the hospital without walls – A CSIRO home telecare initiative, Telemedicine Journal, 6(2) (Jun), pp. 275–281, 2000.
11. Stranieri A, Balasubramanian V. Remote Patient Monitoring for Healthcare: A Big Challenge for Big Data. In Managerial Perspectives on Intelligent Big Data Analytics, ed. Zhaohao Sun, 163-179, 2019.
12. Uddin MA, Stranieri A, Gondal I, Balasubramanian V. A Patient Agent to Manage Blockchain for Remote Patient Monitoring. Studies in health technology and informatics, 254, 105-115, 2018.
13. Benjemmaa A, Ltifi H, Ayed MB. Design of Remote Heart Monitoring System for Cardiac Patients. In International Conference on Advanced Information Networking and Applications, Springer, Cham. (pp. 963-976), 2019.
14. Arora J, Yomsi PM. Wearable Sensors Based Remote Patient Monitoring using IoT and Data Analytics. U. Porto Journal of Engineering, 5(1), 34-45, 2019.
15. Michelson BM, Event-Driven Architecture Overview: Event-Driven SOA Is Just Part of the EDA Story, Technical Report from Patricia Seybold Group, Feb 2006, Available: www.omg.org/soa/Uploaded%20Docs/EDA/bda2-2-06cc.pdf, accessed 14 June 2019.
16. Michelson BM. The Evolution of Service-Oriented Architecture: From Integration to Business Scenario Development, Technical Report from Patricia Seybold Group, January 2005.
17. Sriraman B, Radhakrishnan R. Event Driven Architecture Augmenting Service-Oriented Architecture, Technical Report from Sun Microsystems, January 2005, Available : http://www.omg.org/soa/Uploaded%20Docs/EDA/edamdasoa.pdf, accessed 14 June 2019.
18. Benjemmaa A, Ltifi H, Ayed MB. Design of Remote Heart Monitoring System for Cardiac Patients. In International Conference on Advanced Information Networking and Applications, pp. 963-976. Springer, 2019.
19. Hasan M, Shahjalal M, Chowdhury MZ, Jang YM. Real-Time Healthcare Data Transmission for Remote Patient Monitoring in Patch-Based Hybrid OCC/BLE Networks. Sensors, 19(5), 1208, 2019.
20. Mardini MT, Iraqi Y, Agoulmine N. A Survey of Healthcare Monitoring Systems for Chronically Ill Patients and Elderly. Journal of medical systems, 43(3), 50. 2019.
21. Sharma S, Balasubramanian V. A biometric based authentication and encryption Framework for Sensor Health Data in Cloud. In Proceedings of the 6th International Conference on Information Technology and Multimedia, pp. 49-54. IEEE, 2014.
22. Crossbow Technology Inc., Motes, Wearable Sensors, Available: http://www.xbow.com/, accessed 24 January 2010.
23. TinyOS Embedded Operating System, Available: http://www.tinyos.net, accessed 24 January 2010.

24. Gay D, Levis P, Behren RV, Welsh M, Brewer E, Culler D. The nesC Language: A Hololistic Approach to Networked Embedded Systems, In Proceedings of the ACM SIGPLAN conference on Programming Language Design and Implementation, San Diego, California, 2003, pp. 1–11.
25. Barbar´an J, D´ıaz MD, Esteve I, Rubio B. RadMote: A Mobile Framework for Radiation Monitoring in Nuclear Power Plants, International Journal of Electronics, Circuits and Systems, vol. 1, 2007.
26. Tidwell D, Snell J, Kulchenko P. Programming Web Services with SOAP, O'Reilly, 2001.
27. Gamma E, Helm R, Johnson R, Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Publication, pp. 395, 1994.
28. Hoang DB, Lawrence EM, Ahmad N, Balasubramanian V, Homer CS, Foureur M, Leap N. Assistive Care Loop with Electronic Maternity Records, In Proceedings of the 10th IEEE International Conference on e-Health Networking, Applications and Services, Singapore, pp. 118–123, 2008.
29. Alexander KP, Newby LK, Armstrong PW, Cannon CP, Gibler WB, Rich MW, Van de Werf F, White HD, Weaver WD, Naylor MD, Gore JM. Acute coronary care in the elderly, part II: ST-segment–elevation myocardial infarction: A scientific statement for healthcare professionals from the American Heart Association Council on Clinical Cardiology: In collaboration with the Society of Geriatric Cardiology. Circulation, 115(19), pp.2570-2589, 2007.
30. Balasubramanian V, Hoang DB. Reliability measure model for assistive care loop framework using wireless sensor networks. Journal of Healthcare Engineering, 1(2), pp.239-254, 2010.
31. Balasubramanian V, Hoang DB, Zia TA. Addressing the confidentiality and integrity of assistive care loop framework using wireless sensor networks. In 21st International Conference on Systems Engineering (pp. 416-421). IEEE, August 2011.