Real-time environmental changes and medical risks monitoring in a context-aware system, using distributed mobile applications

Sveatoslav Vizitiu and Lazar Sidor

***Abstract*—** Personal Health Applications (PHA) are tools and services in medical informatics domain, which use information technologies to help individuals create their own personal health information. This paper introduces a personal health application as a distributed mobile application communicating with a context-aware system meant to collect environmental data, using sensors connected to a FBGA board. The physiological risk factors measurements are stored into a database running on web-server and distributed then to all the registered client applications running on mobile devices. The client applications will receive updates from the server with the new available information gathered from the context-aware system showing to the user the possible health risks identified. While a high level overview of the entire context-aware system designed for physiological risk factors measurement station is presented, this paper is focused on the data processing layer and on the software implementation of the client application running on mobile devices.

***Keywords*—** Health Application, Distributed Computing, Monitoring, Autonomic Computing, Context-aware system, APNS.

I. Introduction

Personal Health Applications are thought of as the next generation of consumer-centric information systems that help improve self-management, health care delivery and wellness by providing clear, accurate and complete information, which increases understanding, competence and awareness. Personal Health Application is now part of the Medicine 2.0 movement. eHealth (also e-health) is a relatively recent term for healthcare practice supported by electronic processes and communication, dating back to at least 1999. As Mitchell J. presents in his paper[12], e-Health can be seen as "*a new term needed to describe the combined use of electronic communication and information technology in the health sector... the use in the health sector of digital data - transmitted, stored and retrieved electronically - for clinical, educational and administrative purposes, both at the local site and at distance*".

One of the main factors blocking the use of e-Health tools from widespread acceptance is the concern about privacy issues regarding patient records, most specifically the EPR (Electronic Patient Record). This main concern has to do with the confidentiality of the data.

Sveatoslav Vizitiu is with the Electrical Engineering Department, Technical University of Cluj-Napoca, G. Baritiu street, no. 26-28, Cluj-Napoca, 400027 ROMANIA (e-mail: sveatoslav.vizitiu@gmail.com).

Lazar Sidor is with the Distributed Systems Department, Faculty of Computer Science, Technical University of Cluj-Napoca, G. Baritiu street, no. 26-28, Cluj-Napoca, 400027 ROMANIA (e-mail: lazar.sidor@gmail.com).

Health 2.0 refers to a number of related concepts including telemedicine, electronic medical records, mHealth, Connected Health, and the use of the Internet by patients themselves [2]. Of the forms of e-Health already mentioned, two types: front-end data exchange and back-end exchange. Front-end exchange typically involves the patient, while back-end exchange does not.

Mobile health, or more commonly, mHealth, is *„the use of wireless communication devices to support public health and clinical practice”* [1]. Mobile devices are handheld in nature and include mobile phones, personal digital assistants, patient monitoring devices, and other wireless devices. mHealth applications are receiving increased attention largely due to the global penetration of mobile technologies.

Mobile medical applications range from communication between individuals and health systems (such as call centers, appointment reminders, treatment compliance) to health monitoring and surveillance (including surveys, patient monitoring devices), and access to information at the point of care (health records, decision support).

As the United States has moved toward the development of a national Health IT infrastructure, over 1,500 mobile medical applications have been developed to assist both patients and their clinicians in managing care [6].

Nowadays, the mobile industry boom, combined with the rapid growth and explosion of information and the increasing necessity for having access to this information point towards also the need for computing systems capable of maintaining and managing themselves. Related to this and based on the human body autonomic nervous system, IBM proposed an approach for building autonomous computing systems, targeting the increasing complexity management, by embedding autonomous paradigms into the hardware and software components [14].

The high usage of devices capable of consuming web-services based on specific contextual information points towards also the need of providing accurate and adequate information to end users.

In order to provide such relevant information, any application and service should be aware of its context and automatically adapt to their changing contexts-known as *context-awareness.*

In this paper we describe a context-aware system capable of reacting in case of a health risk detected while performing real-time physiological measurements for a specific GPS location retrieved from a user handling a mobile device.

At the level of data processing layer, our proposed system is based on autonomic computing approach, by containing an autonomic manager for implementing autonomous features.

The relevant data read from the environment by using specific sensors are transmitted to the server in order to be distributed to all connected users.

The rest of this paper is structured as follows. Section 2 provides a short description of the main concepts used in the design and implementation of the proposed system. Then, the overview structure of the context-aware proposed system is presented and described shortly in section 3, while the mobile application design is detailed in section 4. The paper concludes with a short summary and an outlook on further research and development steps.

II. THEORETICAL STUDY

Developing applications, which communicate with a remote server, a data format has to be used for the communication protocol. For a Web Service, XML based SOAP is rife and widely accepted. While XML works fine with many scenarios, it has some drawbacks that makes it difficult to use in some scenarios. One such scenario is AJAX-style Web Services, where XML is difficult to process on the client side. Also, XML is larger in size than its corresponding JSON representation.

JSON is a lightweight data-interchange format [5] and like XML, it is human-readable, platform independent, and enjoys a wide availability of implementations. For each property, XML has two tags: opening and closing. On the other hand, JSON has the property name only once.

In order to send notifications from a central server to a list of multiple client applications, the Push Technology can be used as the publisher or the central server initiates the request for a given transaction. The notifications sent using the Push Technology are called *Push Notifications*.

APNS (Apple Push Notification Service) is a service created by Apple Inc. and it can be used to send push notifications to clients that have registered to receive updates via a configuration profile and are also using the server's mail, calendar and contacts services. Each notification has a maximum size of 256 bytes, which makes it very efficient for services with small data allowances (such as mobiles) [4], [22].

A FPGA board (Field-Programmable Gate Array) is a semiconductor device containing programmable logic components and programmable interconnections.

At the data acquisition layer our proposed system contains a FPGA board and uses digital and analog pins located on the board to communicate with temperature, pressure, humidity and dust density sensors.

The communication with the sensors is made using sensor-specific protocols and interfaces. Communication involves the transmission of commands (made up of 8 bits), waiting for a time interval for the sensors to perform the physical measurements and then read the data stored in the sensor [11].

The improvements in integrated circuit technology, in the form of low-cost, high- density FPGAs with reduced package size as digital interfacing solutions have been presented also in [16] where a biomedical monitoring system is described.

On any autonomic computing approach, an autonomic component offers at least one of the four self* or CHOP paradigms (C = configuring, H = healing, O = optimizing, P = protecting). The CHOP paradigms are implementing at the component level by the autonomic managers. Each autonomic manager enforces one or more CHOP functionalities by executing a MAPE cycle consisting of monitoring, analyzing, planning and executing the plans.

The use of the Autonomic Managers in the system development is also required by the WSDM standard [15] built on SOA design principles and a detailed overview of the basic features related to autonomic computing and its basic goal and functional demands are presented in [21].

In case of developing context-aware systems, the main element in the data acquisition process is the context meant to provide information about the current user status, location or devices in the environment. The concept of context-aware computing was introduced in 1994, as being present in systems capable to "*examine the computing environment and to react to changes to the environment*" [17]. A more complete definition was provided in [18], by taking into consideration the mobile devices with externally connected GPS, additional sensors and network access.

The lack of privacy in context sensitive systems is presented in [19] while an information system applied in the healthcare business and combined with the mobile system is described in [20].

III. SYSTEM OVERVIEW

The proposed system presented in the figure 1 consists of several components analyzed from a three-tier perspective:

1. Data acquisition layer
2. Data processing layer
3. Data visualization layer

At each layer, different hardware and software modules perform specific operations detailed in the next sections

A. Data acquisition

At this layer, the *Context Aware System* interacts with the external *Environment* through the hardware modules consisting of an FPGA board which receives real-time information from the environment, through several sensors programmed to collect relevant meteorological information related to humidity, dust and pressure.

The physical parameters measured by the sensors are converted to electrical signals transmitted to the FPGA board, which sends the data to a web-server for further processing. The messages sent to the server will stipulate the date and time the measurements were made together with the actual measurement values.

For the implementation of the system presented in this article, we will use the following sensors:

1. Compact optical dust sensor GP2Y1
2. Digital humidity sensor SHT15
3. A miniature SPI digital barometer MPL11.

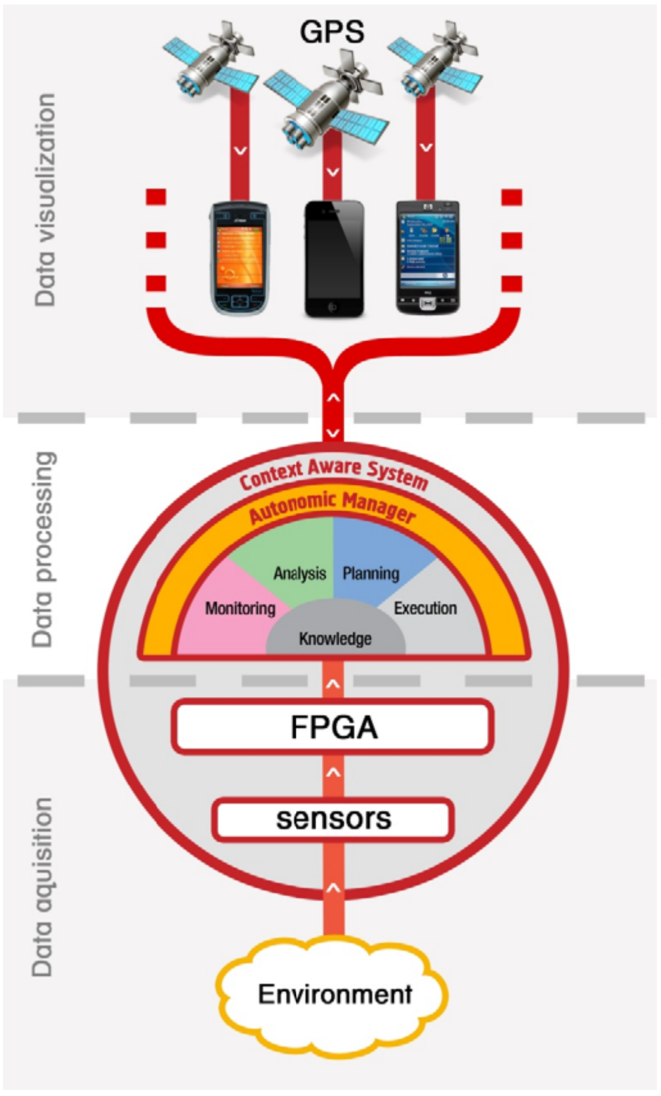


Fig. 1 Context-aware system overview diagram

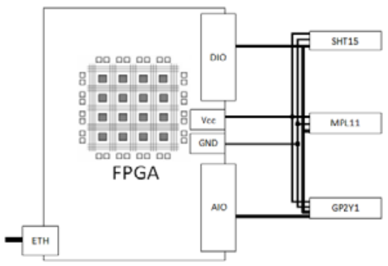


Fig. 2 Hardware architecture for data acquisition

The place where these sensors are connected to the FPGA board is presented in figure 2.

Since the SHT15 and MPL115A1 sensors only have 3.3V digital signals they can be connected only to the DIO terminals on the FPGA board. On the other hand, the GP2Y1010AU0F sensor has both digital and analogue signals, so it needs to be connected to the DIO and AIO terminals located on the FPGA board. All the sensors will get their power directly from the board.

The FPGA board will communicate with the server either through Ethernet or Wi-Fi.

The next sections describe the process of writing data to SHT15 sensor, by using LabVIEW Virtual Instrument for sensor data transmission.

In order to communicate with the SHT15 sensor the FPGA board needs to send a start sequence in order for the sensor to actually begin the data acquisition.

The SHT15 sensor must be connected to the FPGA using 4 pins: Vcc and GND, the digital signal SCK and DATA which are accessed from LabVIEW FPGA.

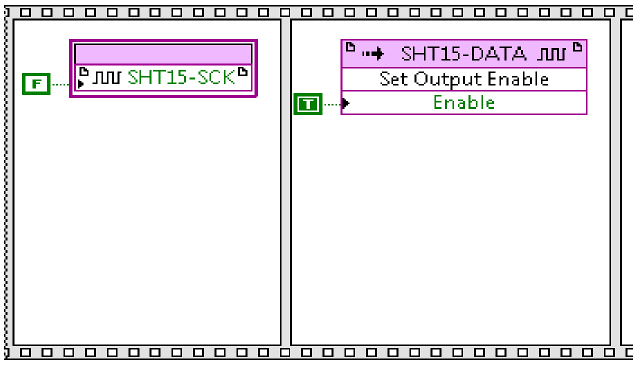


Fig.3 Send byte (part 1)

The FPGA board will need to send the necessary command which is formed of 8 bits and then wait from an acknowledge signal from the sensor.

First we must make sure that SCK is '0 'and the DATA terminal is marked for data transmission.

In a for loop, the program iterates 8 times, and in each iteration, the DATA terminal is assigned the current bit of the command that needs to be sent and then a rising edge followed by a falling edge of SCK are executed in order to actually send the data to the server. The send byte operation is illustrated in figures 4 – 6 below:

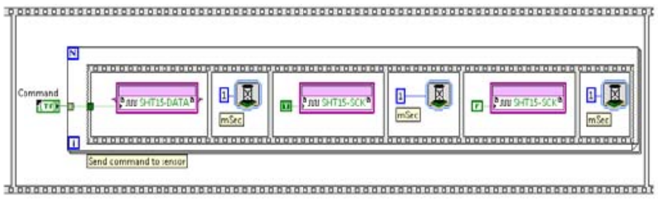


Fig. 4 Send byte (part 2)

After all the 8 bits are sent, the DATA line is released and SCK is assigned '1' to signal the sensor that the board is waiting for a confirmation for the data that was just sent. Then DATA is marked for receiving data.

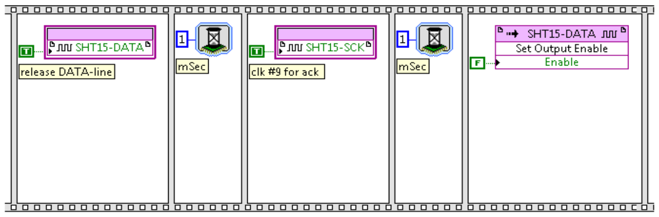


Fig. 5 Send byte (part 3)

At the end of the operation, there is a while loop in which the application waits until it receives the value '1' from the sensor, meaning that it successfully received all the data.

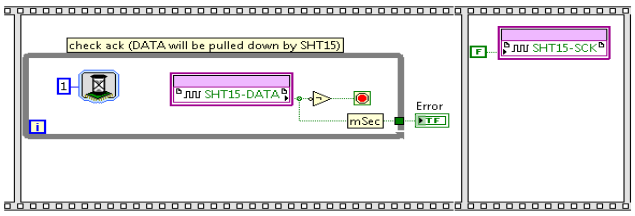


Fig.6 Send byte (part 4)

Initiating a transaction (described in figures 7 - 9) consists in lowering the DATA line while SCK isi '1', followed by a falling and then rising edge of SCK and then raising DATA while SCK is '1' - all these operation are done at a 1ms interval.

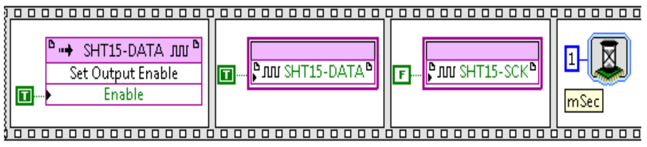


Fig.7 Transmission start (part 1)

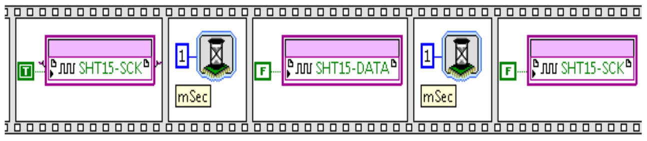


Fig.8. Transmission start (part 2)

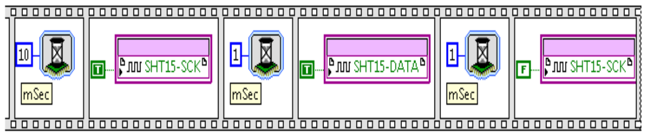


Fig. 9 Transmission start (part 3)

A measurement operation as illustrated in figure 10 is composed of resetting the connection to the sensor (sending the value '1' for 9 clock cycles, followed by a transmission start sequence), sending the command that needs to be done, waiting for a time interval (in this case 400ms) for the sensor to make the measurements and the data in the sensor to be available, reading the MSB and LSB of the data (this needs sending a confirmation to the sensor) and reading the CRC data. Because MSB and LSB are represented on 8 bits, it is necessary to convert this data into an integer so that it can easily be used later.

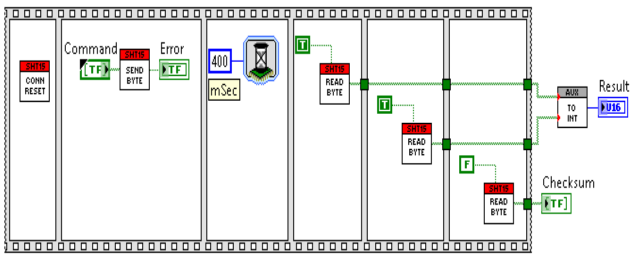


Fig.10 Measurement operation

B. *Data processing layer*

The main component of the entire context-aware system, at this layer, is the Autonomic Manager module responsible of performing M-A-P-E loops in order to process the information received from the data acquisition layer.

The autonomic manager component will be implemented on the web-server side in order to configure the system to react based on the context information and the processed measurements.

At the Knowledge level, the context-aware system will collect environmental information from the dust, pollen, temperature, humidity and pressure available sensors.

Meteorological stations running on a FPGA board and located in certain points of interest in specific locations, are directly connected to the Internet in order to send the collected information to the web-server, in form of XML formatted messages. The XML will stipulate the date and time the measurements were made and the actual measurement values.

In the Monitoring phase, the web-server will continuously check and receive updated information from the available weather stations.

The Analysis phase is responsible for processing and analyzing the XML messages and for inserting the data measurements from the sensors into the database.

In the Planning phase, based on the analyzed information received at the previous step, the risk areas are calculated and for each user a check is performed in order to detect if the last recorded GPS position points to a risk area.

The Execution phase is responsible to take action and to send notifications for each users detected as being localized near or in the risk areas calculated at the previous steps. At the same time, the web-server updates the user related information with their current position coordinates.

Whenever the server gets a message specifying that the user has changed its current position, it needs to do a series of operations such as: compute the health risk in the new area that the user is and also determine the closest pharmacy and hospital in case of an emergency.

Once a MAPE loop is completed, the server is self-configured based on the information received from the meteorological stations and from the user.

C. *Data visualization layer*

At this layer, any user can interact with the context-aware system by using a dedicated mobile application running on a smart-Phone device capable of detecting the user's GPS location. The GPS location becomes the main contextual information used to establish if the user is near a risk area, identified based on the measurements received from the server, for that specific location.

The mobile application sends the user's current GPS coordinates to the server every 30 minutes when the application is running in the background and every minute when the application is open, so the server detects the user location and checks the nearest physiological risk measurement stations to get the relevant environment information for each user.

If there are health risks, such as too much dust, the application will notify the users that they will be jeopardized if they enter in the nearby region and it is not recommended visiting that specific area. If the users still have entered in region identified as representing a health risk zone, the application will find the nearest pharmacies and hospitals in the area and displays the emergency number available for these.

In the next section we present a concrete implementation for the data visualization layer, with a mobile application implemented using Objective-C programming language and targeted to work on devices running on Apple's iOS operating system.

IV. MOBILE APPLICATION DESIGN

This section presents the high level structure of the mobile application and the main functionality flows, from the following points of view:

1. Software Architecture
2. User Interface
3. Application States
4. Push Notifications

A. *Software Architecture*

From the architectural point of view, the application relies on the MVC (Model-View-Controller) design pattern principles and is designed from a multi-tier architecture perspective, as illustrated in figure 11.

The "View" layer contains the main classes modeling the user interface and interaction screens consisting of standard and customized UI controls available in the iOS SDK. Each view is controlled by its correspondent view controller implemented at the "Controller" layer and is responsible to intercept the user gestures and to trigger actions specific to the current active screen.

At the "Model" layer all the business logic is implemented through instances of specific manager classes responsible to implement the actions requested at the controller layer and to communicate with a database for storing and fetching user's and environment related information.

The "Database" layer contains the MySQL database implementation storing information related to: users, hospitals and pharmacies as point of interests, health risk types, sensors data, notification messages and active devices. The information is stored into and fetched from database through the model layer's specific manager instances responsible of creating and populating data objects. Each data object created contains relevant information received from the operating system or from the device's hardware components (e.g: GPS location receiver).

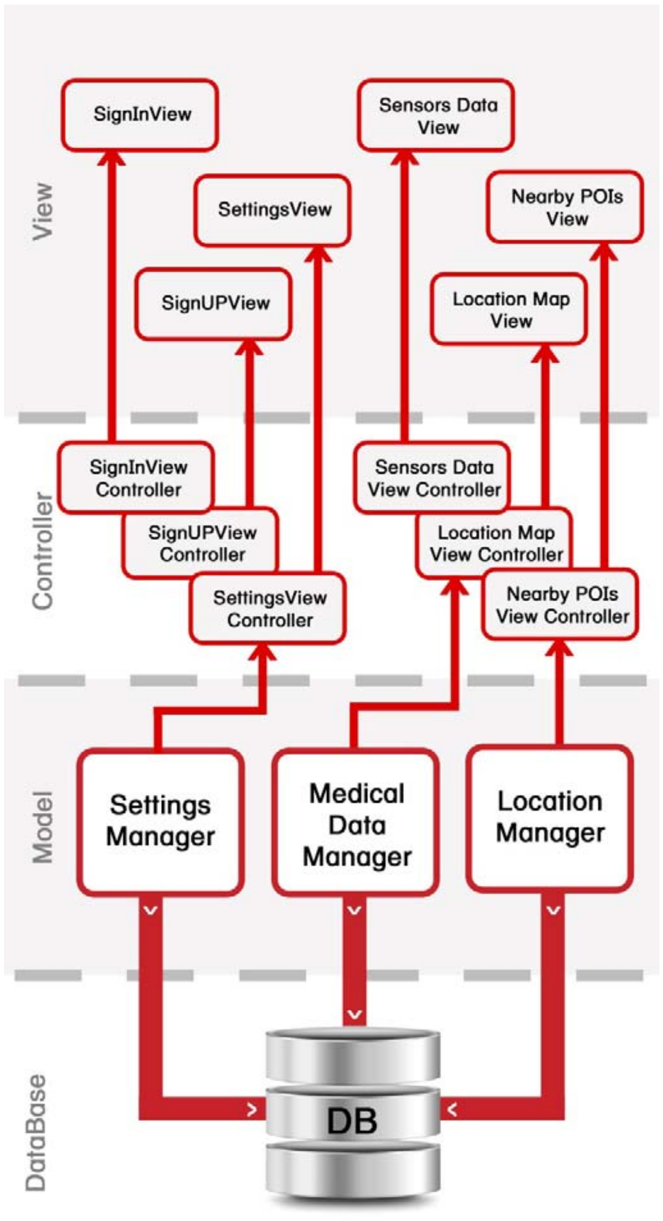
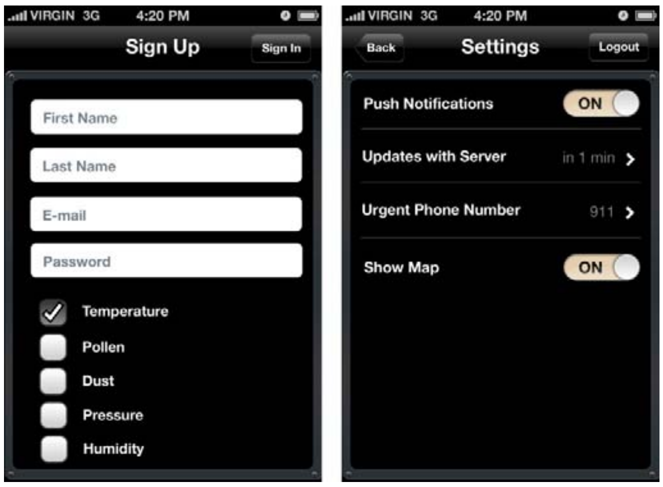


Fig.11 Mobile application class diagram

The database is located on a distributed web-server and the application connects to this via communication over WiFi or Ethernet networks.

B. User Interface

This section describes the most important user interaction views available in the application: Registration, Sign-In, Settings, Sensors Data, Map Location and Nearby POIs.



VIRGIN 3G 4:20 PM

Sign Up — Sign In

First Name

Last Name

E-mail

Password

[✓] Temperature

[ ] Pollen

[ ] Dust

[ ] Pressure

[ ] Humidity

VIRGIN 3G 4:20 PM

Back — **Settings** — Logout

Push Notifications	ON
Updates with Server	in 1 min
Urgent Phone Number	911
Show Map	ON

Fig.12 a) Sign up screen b) Settings screen

Registration view - (illustrated in figure 12. a) allows the user to enter personal information, such as: first name, last name, email address, password and to select the notification categories to receive in case of health risk detected. Each of the notification categories is related to one of the available sensors for: temperature, pollen, dust, pressure and humidity.

The information is also used to uniquely identify the user based on device identifier, at the database layer.

Sign-In view – allows the user to enter his credentials and to login into the application after a successful registration or after a previous log-out operation.

Settings view – allows the user to log-out from the application or to update state for the configurable settings (e.g.: enable or disable push notifications) as illustrated in figure 12.b.

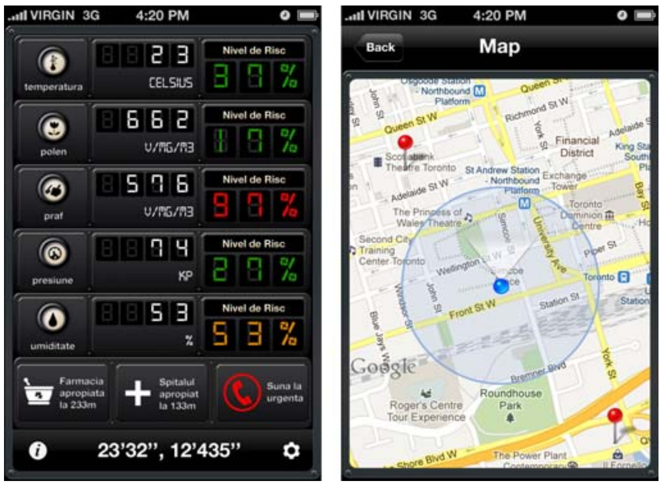


Fig.13 a) Sensor data b) Current location screens

Sensors Data view – represents the main screen of the application and it displays the sensor data from the nearest physiological risk measurement station based on the user's current location. Only data for the sensors that were selected by the user during the registration process will be shown on this screen. For each of the selected sensor, the application will also show the measurement units and the risk level, as illustrated in figure 13.a). The main screen also includes controls for accessing the other screens and functionalities, like the navigation to the nearby points of interest screen or action call of the predefined emergency phone number.

Map Location view – represents the screen where the user is be able to see his current position on the map (as illustrated in figure 13.b) and also has the possibility of checking the sensor data from other physiological risk measurement stations (not just the nearest) that can be seen on the map. All the measurement stations and correspondent risk levels registered at each of these are available as map annotations.

Nearby POIs view – represents the screen displaying hospitals and pharmacies as points of interest based on the current user's location. The hospitals and pharmacies information presented in the detailed views contains essential information as the distance to a certain institution, a link to show the information on the map and a call button that dials the phone number of that specific institution.

C. Application States

The main flow of the application is illustrated in the figure 12, in which the most important states and conditional branches are described and are detailed below:

1. Start - The application can be launched as for the first time or it can become active from the background state. In case of a "first start" launch, all the application resources are initialized together with instance of database and location managers. In case of a "transitioning from background to foreground state", the application will display the latest screen in which the user has navigated.
2. First use check - The application verifies if the user haslaunched the application for the first time. If yes, the register screen will be displayed and the user can sign up using personal information, in order to be registered for receiving notifications.
3. Login check – After a successful registration or in case the user is logged out from the application, the login screen is displayed. The login screen allows the user to enter credentials and to be able to access the main screen presenting all the possible ways of interaction with the application.
4. Load Main screen – After any of the two register and log-in operations, the application communicates with the hardware components in order to retrieve the current GPS location of the user. Once the main screen is loaded and the current GPS location is sent to the server, all the contextual information gathered from the environment is displayed on the screen. From this state, the user has the option to change personal settings or to navigate to other screens responsible of displaying relevant information

related to the current position.

5. Stop – The user can choose to terminate the application or to send it in a background state. Depending on the application state the current location is updated on a different time interval. In the "Running in background" state the application only updates the significant location changes of the user. Once the application is stopped (terminated or running in background) each time the areas of physiological risk measurement stations influence changes and if there are critical health risks in the area around the latest location sent to the server, then the user is notified about the potential risks identified

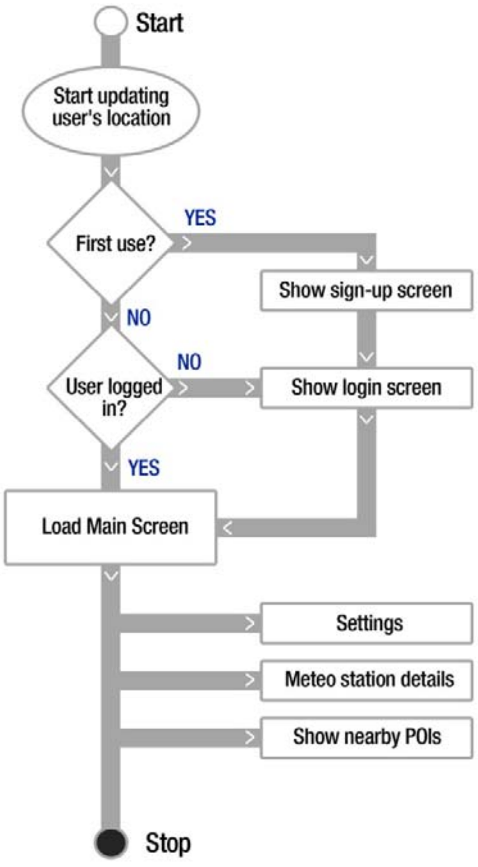


Fig.14 Application state diagram

D. Push Notifications

The Apple's push notification mechanism is used to inform the user about health risk identified for the current position received from the mobile device. No mater which application is currently running on the device and no mater in which operation state is the device: locked or unlocked, the notification alert will be displayed to the user in form of an alert containing information message and option buttons for canceling or opening the application responsible for handling the notification.

A Push Notification provider is an application written by the application's developer to send push notifications to the iPhone application through the APNs.

The APNs (Apple Push Notification Service) is a stream TCP socket that the provider can communicate using a SSL secured communication channel.

The push notification (containing the payload data) is sent as a binary stream (as described in figure 15). Once connected to the APNs, the connection should be kept alive and as many push notifications are required have to be sent within the duration of the connection.

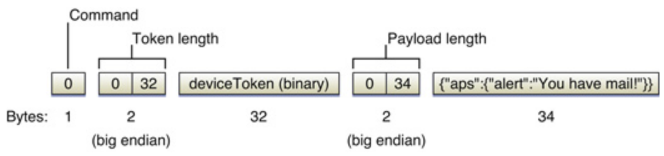


Fig.15 Format of push notification message (based on Apple's documentation [22])

The push notification flow from a provider to a client application and a sample of notification alert is displayed in figure 16 a.

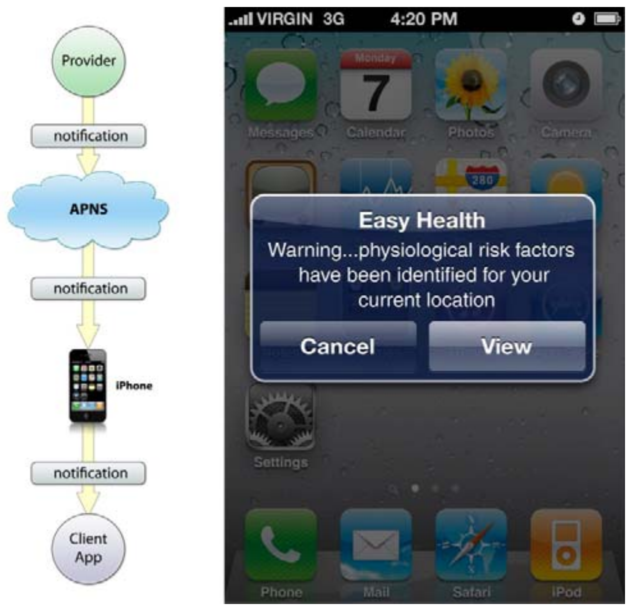


Fig.16 a) Push notification flow b) Sample notification alert

The flow of a push notification data is one-way. The provider (in our case the distributed web-server) composes a notification package that includes the device token for the mobile client application and a payload. The provider sends the notification to APNs which in turn pushes the notification to the device.

When it authenticates itself to APNs, the provider furnishes the service with its topic, which identifies the application for which it's providing data. The topic is currently the bundle identifier of the target application on an iOS device.

Each time a push notifications is required to be sent for alerting the users about a health risk identified, the following action is performed on the web-server (provider) side, in order to send the notification message to the mobile application through APNs:

1. Communicate with the APNs using SSL certificates generated and available on the server
2. Construct the payload for the message to be sent
3. Send the push notification containing the payload to the APNs

The payload is a JSON formatted string (maximum 256 bytes) carrying the information sent to the mobile application. An example of a payload is presented below:

```
{
   "aps": {
   "alert" : "Warning…physiological risk detected”,
   "badge" : 1,
   "sound" :"alert.wav"
   }
   "custom_field1" : "<sensor type>",
   "custom_field2" : "<risk level>",
}
```

On the client side, the operating system will be notified about incoming notifications sent by the APNS server. The application registered for push notifications handles the events received from the operating system and displays the message encoded in the payload stream (as exemplified in figure 16.b).

V. CONCLUSIONS AND FUTURE WORK

The purpose of this article is to present a self-configuring distributed system communicating with a mobile application that can provide useful information about a location based environment and can offer specific assistance in case of an physiological risk detected by the system, based on the sensor’s data received from the external environment.

Easy Health is an iOS application that provides medical information of different health risks to the user. Indicators for dust, humidity, pollen, pressure and temperature are available. The medical information that can be presented includes the value of the medical factor, the measurement units and the risk level. If the risk level for a medical problem is above a certain percentage (e.g. 85%) then this risk is considered critical for the user and the application alerts the user that there are critical risks for his health.

The process of notification relies heavily on the current user location. The application tracks the significant changes of the user location at all the time (also when the application runs in background mode). Combining the knowledge of the current user location and the information collected from the external environment through the physiological risk measurement stations, the system is self-configured and it is capable to determine when a user enters an area with critical health risks.

The physiological risk factors measurement stations (running on FPGA boards) are the measurement facilities that provide information about environmental factors in a certain area. The dynamics of this information are observed by the Easy Health server, which will distribute this information to all the registered users. When the areas of interest to a certain user are affected by a certain critical medical risk, then the user is notified by a push notification, displayed on the user’s device as an alert containing a description message and action buttons.

The server executes MAPE loops in order to monitor, collect, analyze and process the information received from the sensors and then it is self-configured and notifies the clients in case of health risks being detected.

The management of the full scope of the physiological risk factors information on the server and handling the user specific area of interest on a mobile device makes the application especially effective. The essential information is provided to the user all the time and the processing resources of the mobile device are used in a most optimal manner.

The mobile application described in this paper is currently implemented for Apple’s iOS platform only, but it can be easily implemented on any other platform, reusing the high level architectural design and components from the database layer.

As future work for our proposed system we intend to add support for more sensors at the data acquisition layer and to enhance the mobile application at the data visualization layer with the following new features:

1. Integration with other third-party services (e.g: public available healthcare services)
2. Extend the supported types and formats for the information displayed to the user
3. Introduce real-time advice notifications based on the user’s current location

REFERENCES

[1] Kahn JG, Yang JS, Kahn JS: ‘Mobile’ health needs and opportunities in developing countries. Health Policy 2010, 29:252-258
[2] Merrell RC, Doarn CR: Medical applications, mobility, and regulations. Telemed J E Health 2011, 17:235-236
[3] Stephen G. Kochan, Programming in Objective-C, Third Edition (Developer's Library), Addison-Wesley Professional; 3 edition (June 20, 2011)
[4] Joe Conway , Aaron Hillegass , iOS Programming: The Big Nerd Ranch Guide (2nd Edition) (Big Nerd Ranch Guides) , Addison-Wesley Professional; 2 edition (July 2, 2011)
[5] Peter MacIntyre, Brian Danchilla, Mladen Gogala and Adam MacDonald, Pro PHP Programming Apress; 1 edition (August 5, 2011)
[6] Ronald Bradford, Chris Schneider, Effective MySQL Advanced Replication Techniques, McGraw-Hill Osborne Media; 1 edition (September 22, 2012)
[7] Compact Optical Dust Sensor, SHARP Corporation, http://sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y1010au_e.pdf
[8] Humidity and Temperature Sensor, SENSIRION http://www.sensirion.com/en/pdf/product_information/Datasheet-humidity-sensor-SHT1x.pdf
[9] Miniature I2C Digital Barometer, Freescale Semiconductor Literature Distribution Center,
[10] Miniature SPI Digital Barometer, Freescale Semiconductor Literature Distribution Center, http://www.freescale.com/files/sensors/doc/data_sheet/MPL115A1.pdf
[11] Miniature SPI Digital Barometer, Freescale Semiconductor Literature Distribution Center, http://www.freescale.com/files/sensors/doc/data_sheet/MPL115A1.pdf
[12] Converting a Sensor Voltage Input to Physical Units, National Instruments Corp, http://labviewwiki.org/Converting_a_Sensor_Voltage_Input_to_Physical_Units
[13] Mitchell J. From telehealth to e-health: the unstoppable rise of e-health. Canberra, Australia: National Office for the Information Technology; 1999

[14] IBM. Autonomic Computing: IBM Perspective on the State of Information Technology, 2001.

[15] W. Robinson. Monitoring web service requirements. In Proccedings of the International Conference on Requirements Engineering, 2003.

[16]] N. Kimbelry, N. Laramie, C. Medina, "Rapid Prototyping of an FPGA based sensor system for Biomedical Monitoring", Proceedings of the 2006 WSEAS Int. Conf. on Mathematical Biology and Ecology, Miami, Florida, USA, January 18-20, 2006 (pp214-220)

[17] Schilit, Bill N., Adams, Norman I. and Want, Roy (1994): Context-Aware Computing Applications. In: Proceedings of the Workshop on Mobile Computing Systems and Applications December, 1994, Santa Cruz, CA, USA.

[18] Ryan, Nick S., Pascoe, Jason and Morse, David R. (1998): Enhanced Reality Fieldwork: the Context-aware Archaeological Assistant. In: Gaffney, V., Leusen, M. van and Exxon, S. (eds.). "Computer Applications in Archaeology - British Archaeological Reports". Oxford: Tempus Reparatum

[19] Peter Langendorfer, Krzysztof Piotrowski,"More Privacy in Context-aware Platforms: User Controlled Access Right Delegation using Kerberos", Proceedings of the 4th WSEAS Int. Conf. on Information Security, Communications and Computers, Tenerife, Spain, December 16-18, 2005 (pp542-547)

[20] S. Fernandes, V. Vieira, "Information System in healtcare: Potential of Mobile systems. The case of INEM", WSEAS Int. Conf on Sensors and Signals, Algarve, Portugal, November 3-5, 2010

[21] Goran Martinovic, Damir Filko, Miran Karic, "Analysis of Autonomic Computing Concepts in Computational Grid Based on the ACLM Model", 7th WSEAS Int. Conf. on Software Engineering, Parallel and Distributed Systems, University of Cambridge, UK, Feb 20-22, 2008.

[22] Apple Push Notification Programming Guide, http://developer.apple.com/library/ios/#documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/ApplePushService/ApplePushService.html#//apple_ref/doc/uid/TP40008194-CH100-SW9