International Journal of Applied Information Systems (IJAIS) ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 10 - No. 10, May 2016 - www.ijais.org

Design and Development of a Diabetes Management App on Android

T.J. Anande
University of Agriculture, Makurdi
Department of Electrical and Electronics Engineering
Benue State, Nigeria

S. Al-Shehri
Swansea University,Bay Campus, Swansea
Department of Electrical and Electronics Engineering
SA1 8EN, United Kingdom

T.K. Genger
University of Agriculture, Makurdi
Department of Electrical and Electronics Engineering
Benue State, Nigeria

ABSTRACT

Many individuals diagnosed with diabetes find themselves in the need to rethink their common daily routines, and alter their lifestyle and habits. Computing technology, especially ubiquitous computing applications, can help individuals maintain a high level of awareness of their health, reflect on and learn from their past experiences, and generally improve their diabetes management practices. However, to be successful, these applications require continuous engagement from their users and must be able to adapt to the changing environments of use, their users' changing health and their growing understanding of their disease. In this work we designed the graphic user interface (GUI) of an android smartphone with features such as patient's glucose level, time information is received, patient's location with the aid of global positioning system (GPS), like colour code, voice recorder and facial expression for the different glucose levels. This system also utilizes Google sheet to manage Personal Health Records (PHRs), which not only bridges the gaps between patients and different health care providers but enabling accesses to patients' PHRs anywhere and anytime by taking advantage of the universal accessibility of Google sheet. The system further integrates with GPS, Google Search and Google Map functionalities to facilitate the user to find all hospitals near to his/her current location.

General Terms:

Android, Diabetes

Keywords

Glucose level, APP, Location, Smartphone, Sugar level

1. INTRODUCTION

Diabetes is one of the most widely spread chronic diseases in the world. It is especially common among the elderly: nearly one-third of those aged 65 and older have the disease. It is suggested in a publication of the Department of Health and Human Services in the UK that in order to keep glucose at a healthy level, people with diabetes need to keep a balance between three important aspects: diet, exercise and diabetes medicine in daily routine [4][5][3]. Therefore, continuous self-monitoring of the blood glucose (blood sugar), daily diet, physical activity and medicine intake are crucial for the management of diabetes. In this mini-project, we design on an android phone a Graphic User Interface (GUI) for personalized and effective self-monitoring and managing diabetes. The system monitors the blood glucose levels, provides patient's location with the aid of global positioning system (GPS), and the time information is received. Also, we have added other features like colour code, voice recorder and facial expression for the different glucose levels. It is important for patients to monitor their own health conditions, to receive warning and guidance to adjust their behavior during their daily routine and to be able to stay connected with their physicians at all time. Also, the system is to integrate mobile-based and smart home-based health monitoring systems with Google sheet, breaking down the wall between the patient, patient's family members and different health care providers such that they can now provide collaborative care for the patient. Wireless sensor networks (WSNs) and smart phone technology has opened up new opportunities in health monitoring system. The integration of the existing specialized medical technologies with cell phone and wireless sensor networks is a very promising application in home monitoring, medical care, emergency care and disaster response. In the emergency situations, the most important thing is to provide a rapid and accurate assistance to patients with limited resources. And the real-time and continuous triage information must be distributed to health care providers. Light weight and no-intrusive biomedical sensors are easy to be deployed for continuously monitoring the vital signs of a patient and deliver the data to the first responders[2][6][1]. For example, a wireless infrastructure for emergency response. Glucose sensing for the control of diabetes is a high-volume market for biosensors.

Background Theory

This presents the story of behind android and its excellent applications. Android is a Linux based Operating System (OS) that was created basically for touch-screen mobiles devices including Smartphone and tablet PCs. It was initially developed by Android Incorporated, financially supported by Google, who later bought the company in August 2005. Android OS first went public in 2007 along with the founding of the Open Handset Alliance, a consortium of 86 software, hardware and telecommunications companies. This created a new era for the mobile phone industry. Their target in creating Android was to enhance the advancement of open standards for mobile phones. However, the first Android-powered phone was released in October 2008. By December 2010, Android had toppled Symbian as the world's leading Smartphone platform. Today, over 500 million devices run on Android OS. Today, Android has gone beyond Smartphone and Tablet PCs; it now runs on Televisions, Smart books, Cameras, etc. With so much to offer, Google has used Android to give mobile communication an entire new face. Its User Interface uses touch inputs like swiping, tapping, pinching, and reverse pinching (these loosely correspond to real-world actions) to manipulate on-screen objects. It's increasing selection of third-party applications allow users to browse, download and upload appli-

cations published by Google or third-party developers, provided they comply with Google's compatibility requirements. It also manages memory by keeping power consumption low; this is achieved whenever Android application is no longer in use, because the system will automatically suspend it in memory.

Benefits of Android

—Android Smartphone multitask; that is, they are capable of running many applications almost at the same time.

—Notifications are easily accessible and available on the blinking LED indicator on the Home Screen

—Android is available on diverse phones including Samsung, Nokia, Motorola, Sony Ericsson, HTC, etc. It is not restricted to one manufacturer.

Android Disadvantages

Despite Android's endless benefits, they come at some cost.

—Android requires continuous and consistent Internet connectivity. Meaning the user must always subscribe to stay connected.

—While Android phone applications come easily and freely, they always come with adverts which always display either at the top or bottom of the application.

—There are various security issues for the users of Google Android. Where Android identities are not secured things like thefts and hacks are possible. Issues like identity thefts and drunken dialing are common embarrassing situations.

—Its open security free business model lacking security keys and code-signing certificates allows any application to install and run, which is perhaps the biggest drawback. The magnitude of possible malware and junk ware allows the phone to be used as a tool for advertisers.

—Serious security fallout can allow for untraceable spyware. Phones can be hacked and in the result of that data can go into the wrong hands.

—Albeit Android uses Linux as its advantage. Google does not support the Linux distribution. Neither does it support the complete set of standard GNU libraries.

—The Android platform cannot be run from a SD card.

—Android only reuses Java language syntax.

—The Java type app and system cannot be installed

2. DIABETES APP DESIGN

The design sketches for the Glucose Level is fairly simple, with only one screen. In that screen, it is displayed

—the last time you received information about your glucose level,

—the measured glucose level (with some animation to understand the right meaning of the level you have) and

—your current location, which what you can know who you were by the time the glucose level information was received.

There's also a button with which you can refresh the clock and your current location.

Your Primitives

To enable us realized this design, we will used the following primitives:

—One virtual screen

—A method to display the time.

—A method to display the glucose level

—A method to display the current location

—A button to refresh the clock and the current location.

2.1 Getting Started

A new project was created titled, DiabetesManagement.

(1) The Title property of Screen1 was changed to DiabetesManagement.

(2) The Icon property of Screen1 was set with the picture you have chosen previously.

First, we will place a title on the application:

(1) Place a Label component into the Screen1.

(2) Set the Width property to Fill Parent. Change the FontSize property to 22.0.

(3) Change the text and write GLUCOSE LEVEL on. Set the TextAlignment to the center and check FontBold property.

(4) Drag and drop a label below the previous one. Rename it lblSeparation and change the Width property to Fill parent.

Now we are going to build the three main functions of the application: the timer, the glucose level. We will explain each one of this function separately. For the glucose level, we have added some extra animations.

2.2 The Timer

What we are going to achieve with this function is to display on the application screen what time you received your glucose level information. In that way, you can always know when the last time you had a check over your diabetes took place. For the screen design we will use two HorizontalArrangment components. The first of it will be to show a picture and to display the week day when you got your glucose level. The second one, it will appear the exact time (with the format hh:mmdd-Month-yyyy). For the first HorizontalArrangement:

(1) Drag and drop a HorizontalArrangement and place it below the GLUCOSE LEVEL title. Rename it HorArrangTimer1.

(2) Drag and drop an Image component and place it into the HorizontalArrangement. Change the Width property to Fill Parent.

(3) Change the image name to imgClock. Now, you can upload the picture you want. We have chosen a sand clock picture to indicate the timer. Make sure that the picture has the right size.

(4) Drag and drop a VerticalArrangement and place it to the right of the imgClock. Rename it VerArrangTimer1.

(5) Drag and drop a label, place it into the VerArragTimer1. Rename it lblLastInfoReceived.

(6) Check the FontItalic property and change the text property to Last information received:.

(7) Drag and drop another label below the previous one and rename it to lblWeekDay.

(8) Check the FontItalic property, change the FontSize into 18.0. Change the TextAlignment to center. Change the Width property to Fill Parent.

For the second HorizontalArrangement:

(1) Drag and drop a HorizontalArrangement and place it below the previous one. Rename it HorArragTimer2. Change the Width property to Fill parent.

Now, we are going to include other two Horizontal Arrangement components into the previous one, to show the time on the right format.

(1) Drag and drop a HorizontalArrangement and place it into the HowArragTimer2 arrangement. Rename it HorArrangHour. Change the AlighHorizontal to Center and the Width to Fill parent.

(2) Drag and drop a label and place it into the HorArrangHour. Rename it lblHours. Clear the text label.
(3) Drag and drop a label component and place it on the right of the lblHours label. Rename it lblColon. Change the text property to :.
(4) Drag and drop a label on the right of the lblColon label. Rename it lblMinutes. Clear the text label.
(5) Drag and drop a label on the right of the lblMinutes label. Rename it lblMinutes2. Clear the text label.
(6) Drag and drop a HorizontalArrangement and place it on the right of the HorArrangHour arrangement. Rename it HorArrangDate. Change the Width to Fill parent.
(7) Drag and drop five labels and place them into the HorArrangDate arrangement. Place the label horizontally. Rename them lblDay, lblFirstSlash, lblMonth, lblSeconSlash and lblYear, respectively. For the lblFirstSlash and lblSeconSlash labels, change the text property to /. For the other three labels, clear the texts.

With the previous step we will display the time on the right format, showing the week date, the time and the date. To use a clock:

(1) Drag and drop a Clock component from the basic palette. This is non-visible component and it will appear below the screen.

To design the glucose level display, it is necessary to separate the time arrangements of the glucose level one. To do that, we will insert an empty label:

(1) Drag and drop a label below the HorArrangTimer2 arrangement. Rename it lblSeparation1 and change the Width to Fill parent.

To display the glucose level, we only need a label to display the glucose level measure. But, we think it's better to add more animation to make easy to understand what the glucose level is and what it means.

(1) Drag and drop a label and place it below the lblSeparation1. Rename it lblCurrentGlucose.
(2) Change the FontSize property to 20.0 and change the text property to Your current Glucose Level is:.
(3) Drag and drop a horizontal arrangement component and place it below the lblCurrentGlucose label. Rename it HorArrangGlucose.
(4) Drag and drop two Image components and place it into the previous arrangement. Rename them imgFaceSad and imgFaceHappy respectively. Change the Visible property of both of them to hidden. In one of the images we will upload a happy face picture. On the other, we will upload a sad face picture. Make sure that the pictures have the right size. The images will appear depending on the glucose level. If it is good, the happy face will appear. In other case, the sad face will be shown.
(5) Drag and drop a Vertical Arrangement and place it on the right of the two images. Rename it VerArrangGlucose. Change the Width property to Fill parent.
(6) Drag and drop a label and place it into the VerArrangGlucose arrangement. Rename it lblConcentration. Check the FontBold property and change the FontSize to 35.0. Change the TextAlignment to the center and the Width to Fill parent. Clear the text label.
(7) Drag and drop another label below the previous one and rename it lblColour. Change the Width property to Fill parent. Clear the text label.

With the previous steps we will show the glucose level with a more understand format. To do it more understandable, we are going to add some sound to remember us if the glucose level is low, good, or high:

(1) Drag and drop three sound components. Rename them SoundLow, SoundGood and SoundHigh.
(2) Upload three different sounds for each case. For example, we recorded ourselves by saying "Your glucose level is low/good/high", respectively.

Finally, we are going to design the display for the current location. We will insert a picture and show the current location. To design the glucose level display, it is necessary to separate the time arrangements of the glucose level one. To do that, we will insert an empty label:

(1) Drag and drop a label below the HorArrangGlucose arrangement. Rename it lblSeparation2 and change the Width to Fill parent. Clear the text label.
(2) Drag and drop a Horizontal Arrangement below the lblSeparation2 arrangement. Rename it HorArrangGPS. Change the Width property to Fill parent.
(3) Drag and drop an Image component and place it in the HorArrangGPS. Rename it imgGPS. Upload a picture to indicate that the location information will appear. Again, make sure that the picture has the right size.
(4) Drag and drop a Vertical Arrangement on the right of the imgGPS and rename it VerArrangLocation. Change the Width property to Fill parent.
(5) Drag and drop a label into the VerArrangLocation. Check the FontBold and change the FontSize to 16.0. Change the text to Current Location:. Change the Width property to Fill parent.
(6) Drag and drop another label below the previous one and rename it lblAddress. Clear the Text label.

To show the current location we need a location sensor. It's a non-visible component:

(1) Drag and drop a LocationSensor. It will appear near the Clock component.

Once we have our three basic functions, we are going to add a button that will refresh the current time and location.

(1) Drag and drop a button and place it below the HorArrangGPS arrangement. Rename it btnReconnect. Change the text to Reconnect.

Now move on to building the blocks and procedures for your application. As soon as the application starts, there's no current information. Nevertheless, the label must show what information will appear when we press the Reconnect bottom.

(1) Typeblock the Screen1.Initialize event handler.
(2) Typeblock the lblWeekDay.Text [to] block and snap it into the Screen1.Initialize event handler. Set it with a text block. Change the text to Week day.
(3) Typeblock the lblHours.Text [to] block and snap it below the lblWeekDay.Text block. Set it with a text block. Change the text to Hour.
(4) Typeblock the lblMinutes.Text [to] block and snap it below the lblHours.Text block. Set it with a text block. Change the text to Minutes.
(5) Typeblock the lblDay.Text [to] block and snap it below the lblMinutes.Text block. Set it with a text block. Change the text to Day.
(6) Typeblock the lblMonth.Text [to] block and snap it below the lblMinute.Text block. Set it with a text block. Change the text to Month.
(7) Typeblock the lblYear.Text [to] block and snap it below the lblMonth.Text block. Set it with a text block. Change the text to Year.

(8) Typeblock the lblAddress.Text [to] block and snap it below the lblYear.Text block. Set it with a text block. Change the text to Your current address will be shown.

(9) Typeblock the lblConcentration.Text [to] block and snap it below the lblAddress.Text block. Set it with a text block. Change the text to Unknown.

(10) Typeblock the lblColour.BackgroundColor [to] block and snap it below the lblConcentration.Text block. Set it with a Gray color block. That's everything for the Screen1.Initialize event. Yours should resemble Figure 1.

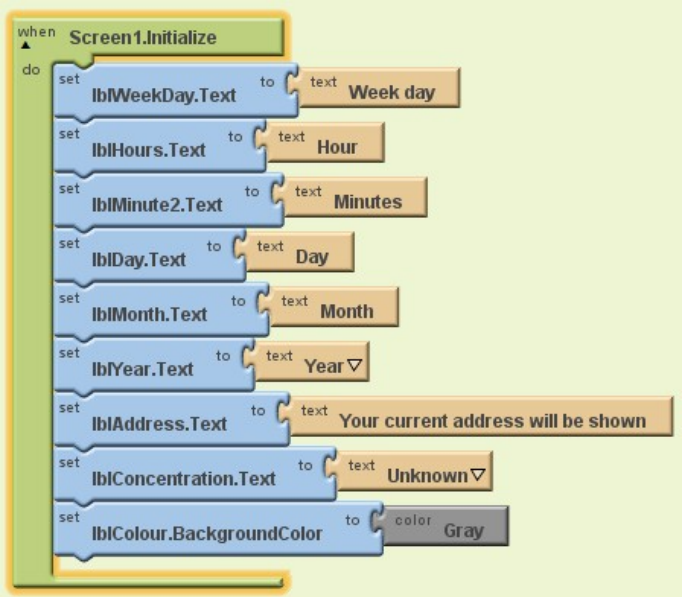

Fig. 1. Screen1.Initialize event.

To give the user control over the application, we use the Reconnect bottom. The reconnect bottom allows you to refresh the date, the location and the glucose level.

(1) Typeblock the btnReconnect.Click event handler.

To show the current time, we will use a clock method called Now method. This returns the time as an instant.

(1) Typeblock an lblWeekDay.Text [to] block and snap it inside the btnReconnect.Click event. Set it with a Clock1.WeekdayName [instant] block. Set it with a Clock1.Now block.

(2) TypeblockalblHours.Text [to] block and snap it below the lblWeekDay.Text block. Set it with a Clock1.Hour [instant] block. Set it a Clock1.Now block.

(3) Typeblock an IfEelse block and snap it below the lblHours.Text block.

(4) Typeblock a less than comparison operator and snap it into the test socket on the IfElse block.

(5) Typeblock a Clock1.Minute [instant] block and snap it into the first socket on the comparison operator. Set it with a Clock1.Now block.

(6) Typeblock a numeral 10 number block and snap it into the second socket on the less comparator.

(7) TypeblockalblMinutes.Text [to] block and snap it into the then-do socket on the IfElse block. Set it with a numeral 0 number block.

(8) TypeblockalblMinutes.Text [to] block and snap it into the else-do socket on the IfElse block. Set it with a text block. Clear the text.

(9) Typeblocka lblMinutes2.Text [to] block and snap it below the IfElse block. Set it with a Clock1.Minute [instant] block. Set it with Clock1.Now block.

(10) TypeblockalblDay.Text [to] block and snap it below the lblMinutes2.Text block. Set it with a Clock1.DayOfMonth [instant] block. Set it with a Clock1.Now block.

(11) TypeblockalblMonth.Text [to] and snap it below the lblDay.Text block. Set it with a Clock1.MonthName [instant] block. Set it with a Clock1.Now block.

(12) TypeblockalblYear.Text [to] and snap it below the lblMonth.Text block. Set it with a Clock1.Year [instant] block. Set it with a Clock1.Now block.

With the above steps, we will refresh the time each time we click on the Reconnect bottom. The Screen1.Initialize event should resemble Figure 2.

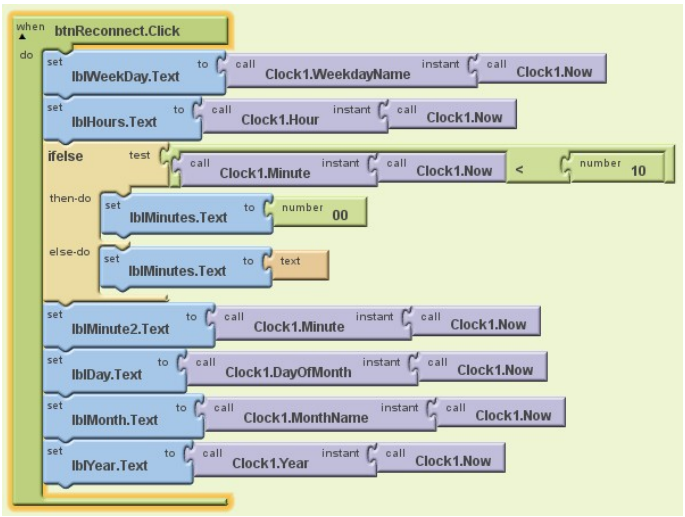

Fig. 2. btnReconnect.Click

For the glucose level, by the time Reconnect bottom is clicked, a random number will be generated. Depending on the number, the glucose level will be low (if the number is less than 70), good (if the glucose level is between 70 and 140) or high (if the glucose level is higher than 140). The glucose level will be generated between 20 and 190. In each case, a different picture will be shown: a happy face if the glucose level is good or, in other cases, a sad face will appear. In each case, a colored label will appear: yellow if the glucose level is low; green, if the glucose level is good and red if the glucose level is high. And finally, we will add the sound "Your glucose level is low/good/high" in each case, respectively.

(1) TypeblockalblConcentration.Text [to] block and snap it into the Screen1.Initialize event, below the time blocks. Set it with a random integer math block. Set a numeral 20 number and snap it into the from socket of random integer. Set a numeral 190 number and snap it into the to socket of the random integer.

(2) Typeblock an IfElse block and snap it below the lblConcentration.Text block. Set a greater than comparison and snap it into the test socket on the IfElse block.

(3) Typeblockalbl.Concentration.Text value and snap it into the first socket on the comparison operator.

(4) Typeblock a numeral 140 number block and snap it into the second socket on the comparison operator.

(5) TypeblockaimgFaceSad.Visible [to] block and snap it into the then-do socket on the IfElse block. Set it with a true block.

(6) TypeblockaimgFaceHappy.Visible [to] block and snap it below the imgFaceSad block. Set it with a false block.

(7) TypeblockalblColour.BackgroundColor [to] block and snap it below the imgFaceHappy block. Set it with a Red color block.

(8) Typeblock a SoundHigh.Play block and snap it below the lblColour.BackgroundColor block.

(9) TypeblockaIfElse block and snap it into the else-do socket of the previous IfElse block.

(10) Typeblock a less or equal than comparator and snap it into the test socket of the new IfElse block.

(11) TypeblockalblConcentration.Text block and snap it into the first socket of the less or equal than block.

(12) Typeblock a numeral 70 number block and snap it into the second socket on the comparison block.

(13) Typeblock an imgFaceSad.Visible [to] block and snap it into the then-do socket of the second IfElse block. Set it with a true block.

(14) Typeblock an imgFaceHappy.Visible [to] block and snap it below the imgFaceSad block. Set it with a false block.

(15) TypeblockalblColour.BackgroundColor [to] block and snap it below the imgFaceHappy.Visible block. Set it with a Yellow color block.

(16) Typeblock a SoundLow.Play and snap it below the lblColour.Background block.

(17) IfElse block. Set it with a false block.

(18) Typeblock another imgFaceHappy.Visible [to] block and snap it below the imgFaceSad.Visible block. Set it with a true block.

(19) TypeblockalblColour.BackgroundColor [to] block and snap it below the imgFaceHappy.Visible block. Set it with a Green color block.

(20) Typeblock a SoundGood.Play block and snap it below the lblColour.BackgroundColor block.

Now, we have already added all the block to show the glucose level with all the animations. The block editor must be like the Figure 3:
Finally, for the GPS, what we need to do is:

(1) TypeblockalblAddress.Text [to] block and snap it below the first IfElse block. Set it with a LocationSensor1.CurrentAddress block.

It will be like in the Figure 4:

3. CONCLUSION

This report presents a personal diabetes monitoring system which integrates smart home technologies and Google sheet to facilitate the management of diabetes conditions. The system further integrates with GPS, Google search and Google map functionalities to facilitate the user to find all hospitals near to his/her current location. In this min-project, we use smart phone with Android platform to collect live streaming data packages. The whole system runs for a reasonable time collecting glucose concentration, time information is received, patient location for effective monitoring. From the experiment, we find the system is robust. And collecting data, displaying data on the cell phone and pushing data into Google sheet are all done effectively. It is simple enough for the end users, especially for the elderly users to use in their daily exercise. This diabetes monitoring system not only assist with the tasks of diabetes management, but also improves the medicine and food safety by taking full advantage of features in existing subsystems in smart home and related cutting edge technologies.

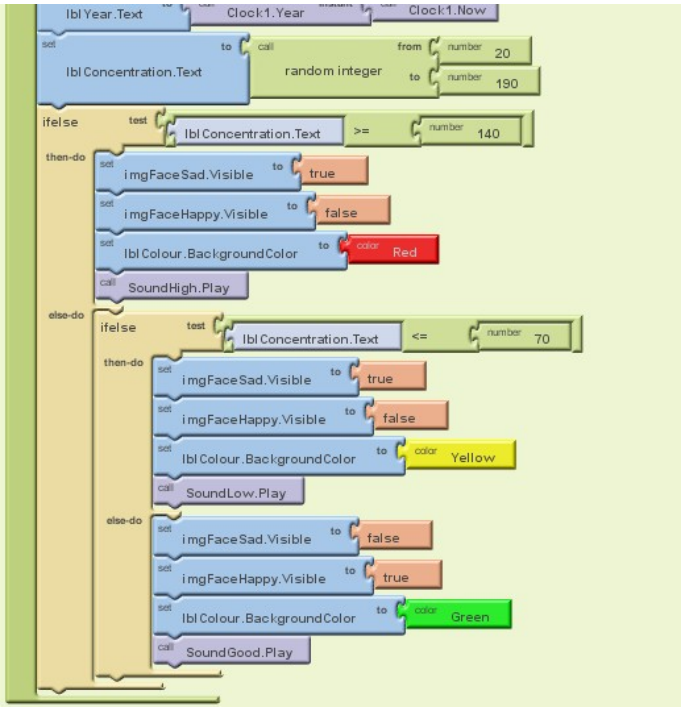

Fig. 3. Block Editor

Fig. 4. Block Editor

4. REFERENCES

[1] Stefan Brahler. Analysis of the android architecture. *Karlsruhe institute for technology*, 7, 2010.

[2] Georgios Kambourakis, Eleni Klaoudatou, and Stefanos Gritzalis. Securing medical sensor environments: the codeblue framework case. In *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, pages 637–643. IEEE, 2007.

[3] David Malan, Thaddeus Fulford-Jones, Matt Welsh, and Steve Moulton. Codeblue: An ad hoc sensor network infrastructure for emergency medical care. In *International workshop on wearable and implantable body sensor networks*, volume 5, 2004.

[4] US Department of Health and Human Services. New survey results show huge burden of diabetes. Available at `http://www.nih.gov/news-events/news-releases/new-survey-results-show-huge-burden-diabetes` (2016/03/20).

[5] US Department of Health and Human Services. Take charge of your diabetes. Available at `http://www.cdc.gov/diabetes/pubs/pdf/tctd.pdf` (2016/03/24).

[6] Huang Xuguang. An introduction to android. *Database Lab. Inha University*, 2, 2009.