

# Cell phone notification via Bluetooth for Web 2.0 applications

Muyowa Mutemwa, William D. Tucker and Michael Norman

Department of Computer Science, University of the Western Cape, Bellville, South Africa  
Telephone: +(27) 21 959-2461, Fax: +(27) 21 959-3006, Email: {2550606, btucker}@uwc.ac.za

**Abstract—** This paper discusses how an Instant Messaging application on a computer can use Bluetooth in order to provide vibration notification on a mobile phone. The initial motivation was to aid Deaf<sup>1</sup> office workers to know when events happened on the computer on their desks. Deaf people with access to modern technology have become accustomed to using Instant Messaging, email and video conferencing. However, most of these applications are designed for hearing users and often use audible notification. Cell phone vibration offers a way to convey similar notifications because many Deaf users have a cell phone. The use of SMS has also become widespread among Deaf users because they cannot hear or speak, even though they might be somewhat text illiterate. Vibration notification in addition to aural notification is common on most cell phones and Deaf users can use the former. This paper describes a Bluetooth notification system to notify a Deaf user with vibration on a cell phone whenever a new Instant Message is received on a given computer. A design goal was to provide an application programming interface to the notification system so that it can be used with any form of Web 2.0 desktop communication tool.

**Index Terms—**Cell phone, vibration, application programming interface, Bluetooth, Deaf users.

## I. INTRODUCTION

THIS paper describes prototypes that enables an Instant Messaging (IM) client running on a desktop computer to use Bluetooth to notify a Deaf user of an event with vibration on a mobile phone. The application domain concerns Deaf office workers with computers on their desks. Today IM systems exist on cell phones, computers, and person digital assistants. Along with the Short Message Service (SMS) and email, IM has greatly increased text communication among people, especially for Deaf users [1]. A Deaf user requires at least some level of text literacy to use SMS or IM, providing that the user first has access to Information and Communication Technology (ICT) [2]. Even with such access, IM systems in general are not designed for Deaf users. Notification of events can be aural or visual, with an emphasis on the former. IM systems such as Google talk, Skype, Yahoo Messenger and Windows Messenger give visual and sound notifications to inform the user that a new event or message has arrived for the client application.

If Deaf users are facing away from their computers, as is customary when communicating in sign language with one another, a different form of notification is required. This paper

describes how we made Instant Messaging notification available on a mobile phone with Bluetooth. The idea is to connect the cell phone to the computer using Bluetooth so that the phone vibrates whenever a new message arrives for the IM client running on the computer. Bluetooth makes the approach cost free. This concept is similar to other solutions of using aural and visual notification, both of which are also free. The advantage is that the target users are Deaf and carry their phones on them at all times because they use them to communicate using SMS. A Deaf user knows when a new SMS has arrived by feeling the vibration of the phone. In this way the system described in this paper can make the phone vibrate to tell a Deaf user that a new IM has arrived on the computer.

Section II gives a brief background on the Deaf users we worked with, and their familiarity with technology. Section III describes user requirements and how the research problem arose. Section IV analyses the user requirements to formulate the research question. Section V presents an initial prototype design along with its implementation and limitations. Section VI describes a second prototype that provides an alternative solution to achieve notification with cell phone vibration. That section also discusses the second prototype's advantages and limitations. Section VII discusses user feedback from field tests with the second prototype. Section VIII discusses the overall results. Section IX holds the conclusion and Section X has suggestions for future work.

## II. BACKGROUND

A hearing impaired person cannot fully hear sound but can use a combination of the other four senses, or some kind of aid, to compensate for not being able to hear. The word deaf, as used in this paper, includes three categories of hearing impaired people. The categories are deaf, hard of hearing and Deaf. Firstly, deaf people are people who lose their hearing due to an accident or some progressive physiological disorder. Many deaf people can read and write. Similarly, hard of hearing people lose their hearing capabilities, usually due to advanced age. Both categories tend to make use of external or internal implanted hearing devices, meaning they can still fit in with the hearing population. Deaf people, on the other hand, are often born Deaf and their main language for communication is sign language. Deaf people are therefore a minority in the overall deaf populace (roughly 10% of the South African population [2]). Deaf people tend to be textually illiterate as well as computer illiterate. However, these days, many of them are cell phone literate.

<sup>1</sup> Deaf with a capital 'D' is different from deaf or hard of hearing in that Deaf people primarily use sign language to communicate and define their sense of culture, as opposed to the other groups that use textual languages like English or Xhosa.

We are primarily concerned with Deaf users who are staff members of the Deaf Community of Cape Town (DCCT), a non-governmental organization (NGO) based in Cape Town, South Africa. They help the overall Deaf membership mostly with social services. Most DCCT staff members have a moderate degree of text (English) literacy, e.g. they can read and write. However, none of them have completed high school. Because they are Deaf, they mainly use South African sign language to communicate with each other.

Since 2004, various Deaf telephony prototypes have been tested out with DCCT staff and members [2]. One positive result was an increase in computer usage and literacy amongst DCCT members associated with the centre, mainly due to the establishment of a small computer lab on their premises, the Bastion of the Deaf (herein referred to as the Bastion). When the ICT projects started, few Deaf users had cell phones. Currently, most Deaf users have a low-end mobile phone. All of these phones support SMS and can vibrate to alert a Deaf user of an incoming SMS.

Most DCCT users have become accustomed to using SMS, which is similar to IM, except that the communication is asynchronous and drastically more expensive. When they receive an SMS the phone vibrates and because they keep their phones on them all the time. It is therefore easy to respond to the vibration [1]. To repeat, most of the staff members at the DCCT are Deaf. That means they have to constantly move around the office to make use of face-to-face sign language communication. With computers on their desks, the staff at the DCCT make use of IM systems such as Google talk and Skype. A result of them moving around the office quite often means they have no way of knowing that they have received a new IM. The challenge to deal with this problem arose from one DCCT staff member who works at the Bastion. He suggested that we find some way to vibrate a cell phone when something happens on his computer.

### III. USER REQUIREMENTS

Although this idea was introduced to the research team by one of the DCCT staff members, he could not participate in the design process. However, he was able to provide feedback on the exploratory prototypes that are described below. He explained that because Deaf people at the DCCT can make use of sign language to communicate with each other, they must move around the office to use face-to-face communication. When they do so, they always carry their cell phone with them. He thought it would be a good idea to somehow integrate Web 2.0 applications with cell phones to ease Deaf people's use of such technologies. An interview was arranged with him to listen to his idea and consider if it was programmable. Table 1 shows the results of the interview. Other points obtained from the interview were that the DCCT's main concern is to educate and introduce Deaf people belonging to DCCT community to computer and cell phone related technologies in order to improve their individual lifestyles. The suggested cell phone notification can be used by Deaf people who have to constantly move around an office environment instead of just being stationary at a desk, and can

even be adopted by hearing users as well. The user requirements are summarized in Figure 1.

Criteria	Result
What is the level of written language literacy among the DCCT members?	They can read and write enough to use SMS
How many make use of mobile phones?	Most
How many have mobile phones?	Most have them
The type of phones they have	Low end phones, but a few good phones
And their normal or most usage?	SMS
Do the phones have Bluetooth?	No, only a few have Bluetooth
Do they use Instant Messengers?	Yes
How often do they use Instant Messengers?	Frequently
What do they use it for?	Chatting

Table 1: The table shows the interview questions and answers used to obtain the user requirements.

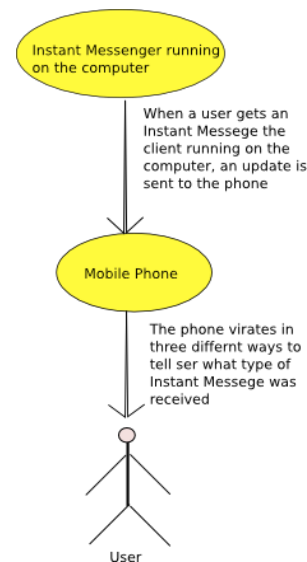


Figure 1: The user requirements can be summarized as the need to connect a Web 2.0 application with notification on a mobile phone.

### IV. REQUIREMENTS ANALYSIS

From the user requirements we obtained the following. First, the target users are Deaf staff members at the DCCT. They have to constantly move around the office to use face-to-face sign language communication. This means they have to leave their desk in order to communicate with another Deaf person. Their offices are within a radius of about 10 meters. All of them have cell phones and are language literate enough to read and write to use SMS and IM. They all have computers on their desks. They normally use their cell phones for sending and receiving SMS. This means they are used to responding to

an SMS when they feel the cell phone vibrate. We devised a technical solution to address the user requirements in the following way, as illustrated in Figure 2:

#### A. Receiving the instant message

When one user sends an IM message to another user, the IM client puts the information about the received IM in a log file making it easy to read the file by using code. That means that as a new message is received by the IM client, the information regarding the IM such as the sender, time, and type is added to the log file found in the IM systems installation directory.

#### B. The WIN32 application sees a new IM

This is a C++ application that will check for new information added to the log file. That means as soon as the information about the IM is added to the log file, the application can react.

#### C. WIN32 application runs Bluetooth code

There will be Bluetooth code within the win32 application. The Bluetooth code will always have the port open for sending and receiving information. This Bluetooth code will be able to pair with the phone, and send a message to the phone to vibrate.

#### D. Receiving the update on the phone

A Symbian mobile application will always be running on the phone. This application will receive updates from the computer without the need to give authorization, reading the update to determine the type of IM message received on the computer and then vibrate in three different ways based on the type of IM (voice, text and video). The user will then be able to understand what type of IM that has arrived on the computer and respond accordingly.

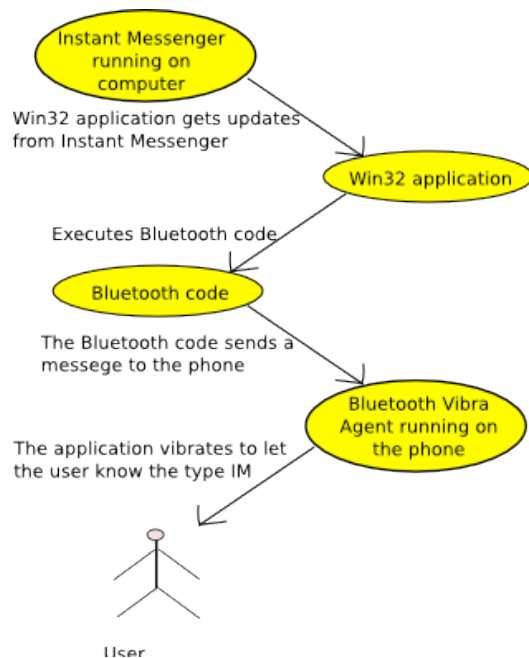


Figure 2: The figure shows how the user would interact with the system in order to respond to an IM message while being away from the computer.

## V. INITIAL PROTOTYPE

An initial prototype was designed for the original concept described in Section IV. The prototype had an application running on the mobile phone that would vibrate in one of three ways depending on the type of IM received. This section describes its high level design, implementation and limitations.

#### A. High level design

Figure 3 shows the design of the win32 application and the various attributes associated with it. This is the class for an application that runs continually on the host machine checking for new messages in the IM system's log file. Figure 4 shows the design for the Bluetooth code that is executed when a new IM message arrives on the computer. It is responsible for connecting and sending updates to the phone. Figure 5 shows the design of the Symbian application that is responsible for pairing with the computer through Bluetooth. It listens for updates through the Bluetooth port on the phone, displays a user interface for the user and vibrates accordingly when a an IM arrives on the host machine.

Win32_Application
<i>Attributes</i>
private String type_of_im
private String name
private int time
<i>Operations</i>
public Win32_Application( )
public void check_IM_message( )
public void run_bluetooth_code( )
public String getType_of_im( )
public void setType_of_im( String val )
public String getName( )
public void setName( String val )
public int getTime( )
public void setTime( int val )

Figure 3: The win32 application is responsible for connecting the IM system with the rest of the solution. Its functions include: extract the important details from the IM system, determine if an IM was received, e.g. a text message and execute the Bluetooth code.

BLuetooth_code
<i>Attributes</i>
private int time
private String name
private String type_of_im
<i>Operations</i>
public BLuetooth_code( )
public void establish_connection( )
public void send_message( )
public int getTime( )
public void setTime( int val )
public String getName( )
public void setName( String val )
public String getType_of_im( )
public void setType_of_im( String val )

Figure 4: The Bluetooth code is set in motion by the win32 application. It is responsible for connecting the PC and the phone via Bluetooth, and sending the message through to the phone.

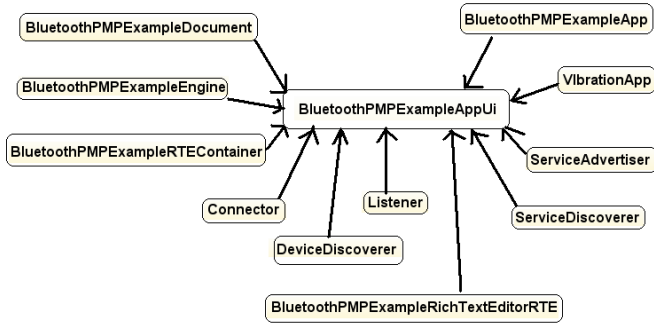


Figure 5: High-level design for the application running on the phone.

The Symbian application is responsible also for establishing the connection between the phone and the PC, vibrating the phone in three different ways, storing messages and providing an interface for viewing the messages sent from the PC to the phone. Figure 5 shows the classes needed to construct the Symbian application. All of the classes are called as instances in the main class, which is **BluetoothPMPEExampleAppUi**. This class controls the user interface inputs and handles the user interface of the application as commands are received from the user. The required class is called to handle the command, for example, to initiate the application the **connect** class is called to check if the Bluetooth is switched on, if there is a free port available for receiving communication from the computer, and connects to the computer [5].

### B. Implementation

The main aspect of the phone application is the vibration. The Vibration API on the Symbian phones is called CHWRMvibra. This API can be used in two modes:

1. **Without notify handling** – when an instance of the class is just created without having to derive the class. The instance is created if the client application will require up-to-date status information.
2. **With notify handling** – when an instance of the class is created without a callback pointer. The CHWRMvibra class has to be derived.

After an instance is created, vibration can be directly controlled via the provided class methods as follows [5]:

**iVibra->StartVibraL(5000, 50)** start vibration for five seconds with intensity of fifty.

**iVibra->StopVibraL()** stop the vibration.

**iVibra->ReleaseVibra()** release the resources used during the vibration.

**CleanupStack::PopAndDestroy(vibra)** destroy instances created at the top of the file.

We created an application on the phone that could pair with the computer by running Bluetooth code on the PC and the application on the mobile phone. The connection was

successful. On the computer we used the **connect (host Bluetooth IP, host Bluetooth friendly name, UUID (universally Unique identifier), authorization (an integer))** method found on the MSDN API for Windows networking applications. On the phone we used the **connect (host Bluetooth IP, host Bluetooth friendly name, UUID, authorization)** method phone in the Carbide C++ S60 API for networking application. Then we used the **send (host Bluetooth IP, host Bluetooth friendly name, data (this is sent as a string), length/size (the number of characters or the size of the string))** on the computer and the **listen()** Carbide C++ S60 API [6].

The application could connect to the computer itself. Also, two mobile devices running the application could be connected to each other and the messages could be sent through. The vibration could be controlled by altering either the time or intensity on the **startvibra()** method. This was easy because the two applications had the same Bluetooth stack used for communication.

### C. Limitations

The **send ()** method on the computer can send the string but the **listen ()** method on the application on the phone could not receive the message although an indication from the phone's operating system showed that a Bluetooth message was about to be received. This is the main reason why the two devices could not send messages to each other and the motivation to design a second prototype.

There are two different types of host controller interfaces: (HCI) transport layers used are the **USB** (on the PC) and the second is **UART** (on the mobile phone). Each uses a different hardware interface (stack) to send and receive the same data. The HCI transport layer allows the host stack and the controller to swap data with minimal adaptation, but the algorithms are sometimes manufacturer-specific for transmitting and receiving. This way information cannot be transferred from one device to the next very easily [4].

## VI. SECOND PROTOTYPE

### A. High level design

The second prototype was built as an alternative solution because the first prototype had the computer to mobile phone connectivity problem discussed above. Therefore an alternative prototype had to be designed that could achieve the same goals: cell phone vibration and notification. The alternative second approach is to send a text file from the computer to the phone. The design of the Bluetooth code and win32 application remained the same because the problem was on the cell phone application side not the computer design (see Figures 3 and 4), but the **send()** method was changed to send a text file instead of a string.

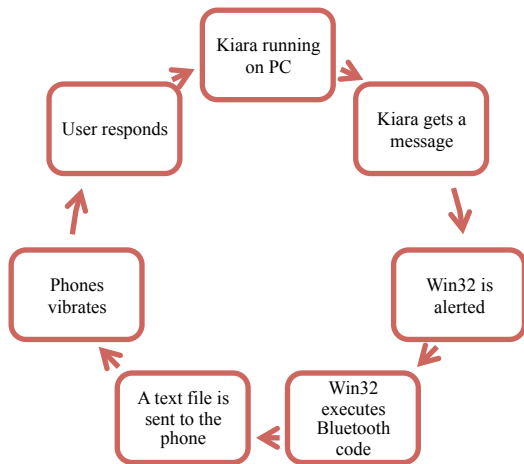


Figure 6: The design of the alternative solution, there was no coding done on the phone because the phone's Operating System can read text files like SMSs.

### B. Implementation

A cell phones receives a text file via Bluetooth in a similar way it receives an SMS. This makes it easy to vibrate the phone whenever a Bluetooth message arrives on the phone by setting the phone to vibrate every time an SMS is received. Symbian phones can read text files because the operating system does cater for this. So the alternative solution is for the win32 application to look for a new IM in the log file, create a text file for sending the information about the IM, and then activate the Bluetooth code that sends the new text file to the phone. On the phone the text would be treated like an SMS and can be found in the message inbox. The user can read it like an SMS but it has a Bluetooth icon differentiating it from a normal SMS [7].

### C. Advantages

The main advantages of this approach are that the text file can be received on the mobile phone as an SMS, and it can be found in the inbox like any other message. When the text file arrives on the mobile phone, it causes the phone to vibrate. This solves the primary goal of cell phone notification via vibration. The text file can be edited by the win32 code on the computer to read like an SMS on the phone. The text can identify the IM type from the host machine. The message looks different to a normal SMS because it has a Bluetooth icon.

### D. Limitations

The method's main disadvantage is that the arrival of the text file vibrates like a normal SMS. This means that the user cannot feel the difference between a voice, text or video IM that arrived on the computer. The user must look at the contents of the message to see what has transpired.

## VII. USER FEEDBACK

The second prototype was designed before the user testing was done, and this was the prototype that was tested at the Bastion. Three members of the DCCT staff tested the prototype. No incentives were given to them as they

volunteered to test the prototype. Each person who evaluated the prototype had to sign a consent form. An interpreter was used during the test. Each evaluation with a user lasted for less than five minutes. More information from the evaluation was obtained using questionnaires and also from their facial expression as they used the system. The feedback obtained from the questionnaire is summarized in Table 2.

Criteria	Result
Number of participants	3 male, Deaf participants
Type of phone	E51 & Samsung SGH – D900i
Do their phones have Bluetooth?	Yes
Use of Instant Messenger	Moderate, usage.
Is the product easy to use?	Moderate
Is the product friendly?	Moderate
Does it meet the necessary requirements?	Primary requirement is met, to give notification & vibrate but they expected the vibration to differ with each IM.
Is the notification understandable?	Yes
Is it efficient and effective?	Moderate

Table 2: User feedback through questionnaires. The table summarizes the user feedback obtained during testing.

## VIII. DISCUSSION

The prototypes are not portable across different platforms. For example, the second prototype can only be used on Microsoft Windows XP and on Microsoft Vista. This is because the win32 application makes use of the Microsoft Platform SDK for Windows Server 2003 R2. This library can only be used for Windows-related operating systems. The system design requires the use of a supported Bluetooth dongle. This is mandatory. An onboard Bluetooth device would work best because some Bluetooth dongles do not interoperate [4] [6]. The mobile phone has to be able to read text files. A phone with Symbian OS is preferred because it can read text files. The system is currently restricted to Deaf personnel working in an office at the Bastion. The solution is ideal for an office environment as there is little signal disruption that may interfere with the Bluetooth signal. As long as the user uses the system within a radius of ten metres as specified for class 2 Bluetooth devices, the system should function properly [7]. Although the system is intended for Deaf people, it can also be used by hearing people who wish to integrate IM on a computer with a cell phone for IM updates.

## IX. CONCLUSION

The mobile phone notification via Bluetooth for Web 2.0 PC-based applications is a system that allows the computer based application to send notification to a mobile phone to indicate that a voice, text or video Instant Message has arrived on the computer running an IM system. The system described in this paper was integrated with a home grown open source IM system. It could notify a Deaf user when an Instant

Message has arrived on a computer. The notification obtained on the mobile phone resembles that of a normal SMS thereby making it user friendly for anyone who has used SMS before. The Bluetooth notification system can give notification to any phone that has Bluetooth capability and can open text files. This system is ideal for the office environment as Bluetooth can transmit the notification without object obstruction and cell phones gives mobility to its users especially Deaf users as they constantly have to move around the office to use sign language with one another.

#### X. FUTURE WORK

We used the second prototype for user testing, with text files being sent to the phone's inbox instead of the mobile vibration application on the cell phone. This is because the computer's Bluetooth stack is different to that of the mobile phone in the current Bluetooth specifications [7]. We could only connect the mobile vibration application and the computer but were unable to send a message from the computer to the application on the phone under the first prototype, which was the ideal solution. This however might change under the new Bluetooth specification as new features are introduced and Symbian application development in Bluetooth for communication between the phone and the computer might be made more programmer-friendly. The system was only implemented with a homegrown IM system. The prototype could be encapsulated with an API so that it can also be used with any IM system. Future work includes integrating the notification code with another open source IM system.

There is also a need to extend the system from the mobile phone application side to be able to handle message reception and cause the system to vibrate in different ways depending on the type of message, or event, on the computer. Also we could integrate the notification tool into other Web 2.0 tools, e.g. Facebook. Our solution was originally intended to be used by Deaf people but it can also be used by any person who works in an office, has a Bluetooth-enabled phone, and wants remote alerts for Web 2.0 events.

#### ACKNOWLEDGMENTS

The authors thank the staff and members of Deaf Community of Cape Town (DCCT) for their participation in the project. We also thank Telkom, Cisco and THRIP for financial support via the Telkom Centre of Excellence (CoE) programme. This project is also financially supported by SANPAD, the South Africa-Netherlands Research Programme on Alternatives in Development.

#### REFERENCES

- [1] Power MR and Power D (2004). *Everyone Here Speaks TXT: Deaf People Using SMS in Australia and the Rest of the World*. *Journal of Deaf Studies and Deaf Education*, 9(3), 333-343.
- [2] Glaser, M., & Tucker, W. D. (2004). *Telecommunications bridging between Deaf and hearing users in South Africa*. In *Proc. Conference*
- [3] Bruce Hopkins & Ranjith Antony: *Bluetooth for Java*. Apress 20003
- [4] S60 Vibra API Specification: Using Vibra API. Retrieved May 13, 2009, from

- [http://www.forum.nokia.com/document/Cpp\\_Developers\\_Library/GUID-759FBC7F-5384-4487-8457-A8D4B76F6AA6/html/S60\\_Vibra\\_API\\_Specification4.html](http://www.forum.nokia.com/document/Cpp_Developers_Library/GUID-759FBC7F-5384-4487-8457-A8D4B76F6AA6/html/S60_Vibra_API_Specification4.html).
- [5] QH Mahmoud. *Part II: The Java APIs for Bluetooth Wireless Technology*. 2003 - fivedots.coe.psu.ac.th
  - [6] Bluetooth API. Retrieved May 13, 2009, from [http://wiki.forum.nokia.com/index.php/Bluetooth\\_API](http://wiki.forum.nokia.com/index.php/Bluetooth_API)

**Muyowa Mutemwa** holds an Honours degree in computer science from the University of the Western Cape (UWC). The author is currently studying for a Masters degree at UWC with the Bridging Applications and Networks Group (BANG).

**William D. Tucker** is a senior lecturer in computer science at UWC and leads BANG research there. He recently submitted a PhD thesis on design and evaluation abstractions for information and communication technology for development. One of his field studies concerned technology for Deaf users.

**Michael Norman** is a senior lecturer in computer science at UWC. His teaching and research interests are in software engineering.