Traffic Control by Bluetooth Enabled Mobile Phone

G. Manikandan and S. Srinivasan

Abstract—**This work confers an application, which makes possible to use a Bluetooth enabled mobile phone to remote control Traffic Signals connected to the personal computer. It can also be used to control other computer applications such as Emergency and Road Map. It is a client – server based application, which makes use of powerful Java technology to achieve its end. The program works on a variety of Bluetooth enabled mobile phone and is compatible with majority of Bluetooth stacks.**

Index Terms—**Bluetooth, mobile phones, home networking, PAN, wireless networks, J2ME, java.**

I. Introduction

An electronic device, which has intruded our life, more than any other device are mobile phone. It keeps us connected to every part of the world all the time. It contains some of our most guarded secrets. It is our notepad, guide, alarm and what not. It looks as if sometime in the future everything would be possible to do by a mobile phone. Our work also is a step to empower this small device. It enables a mobile phone with Bluetooth to control a computer system and all the hardware attached to it.

II. Motivation Behind the Work

Simply speaking, Bluetooth enabled remote control is a remote control for a personal computer. The interesting thing is that the remote control is nothing but your mobile phone. Your mobile phone acts as a remote control for the applications present in your computer. For instance, you can know the details about road map and emergency services present in your computer with the help of your mobile phone.

Infrared is normally used by a remote control of television. But Bluetooth enabled remote control uses Bluetooth. Why? Because many mobile phones today contain Bluetooth and similarly many personal laptop and computer are also coming equipped with Bluetooth. Thus instead of using extra hardware required by infrared remote control, we can use already available Bluetooth hardware. So this gives us a major advantage i.e. No extra Hardware Cost. Bluetooth Enabled Remote Control is a software program. Part of it resides in a computer and a part in mobile phone. Each Bluetooth hardware requires a program called Bluetooth stack to be installed before use. Different vendors have different Bluetooth stacks and most of the time they are incompatible with each other. We have tried to make Bluetooth enabled Remote Control generic and successfully achieved it.

Manuscript received February 24, 2012; revised april 27, 2012.

G. Manikandan is Research Scholar, Sathyabama University, and Chennai. (e-mail: mani4876@gmail.com)

S. Srinivasan is with Director of Affiliation, Anna University of Technology Madurai, and Madurai.

III. Operational Overview

The system basically involves communication between a mobile phone and computer application.

Fig. 1. Data exchange

A computer and a mobile phone can communicate with each other using ports. Similar to two islands which communicate with each other using ship ports or airports. Technically we call them COM ports. Data can be exchanged using ports i.e. port of mobile phone and port of computer. This data is known as commands. When the commands are received from the user's Mobile Phone the server accepts and produces the desired operation on the Personal Computer. The server has been implemented in J2SE and operations are produced in a variety of ways such as by producing virtual key presses, calling different executable files.

IV. System Components

The "Bluetooth enabled Remote Control" has two components, namely

1) Client Program
2) Server Program

V. Client Program

Client program uses Bluetooth API to create Bluetooth connection with the computer. The "Client Program" has been implemented in J2ME [1]. The jar file has to be transferred and installed on the Mobile Phone. After opening the application the user has to select "Search" from the menu so that all the nearby Bluetooth devices appear on the mobile screen.

This operation takes 20 to 40 seconds under normal circumstances. This time for searching can be eradicated if the same Personal Computer is used again and again by storing the serial port service URL and using it directly to open connection. When the list of operations is displayed, user can select the appropriate operations (Personal

Computer on which Server Program is installed). Then pairing of the operations occur and using the 48 bit Bluetooth address of the Personal Computer a serial port connection is established between the Personal Computer and the Mobile Phone.

After this a list of operations are appear on the screen;

- Traffic Signals
- Emergency
- Road Map

On selecting "Traffic Signals" the program displays the following list;

- Stop
- Go
- Etc.

On selecting "Emergency" the following menu appears;

- Ambulance
- Fire
- Police
- Exit

Now we will discuss the internal operations, which are performed by the "Client Program". Firstly the program discovers all the nearby devices using *startInquiry*( ) [2].

Once the device is discovered a RFCOMM Stream connection or a virtual serial port connection is established with the Personal Computer and the Interface three is displayed to the user. On the selection of an option from a sub menu a command is sent to the "Server Program".

This command is actually a 5-character code. The right most character is identified as character 0.

Character 4,3,2,1 defines the application on which the operation has to be done. Character 0 defines the operation, which has to be done on the application specified by the character 4,3,2,1. E.g. If the user selects Previous from the Interface "Road Map" then the following code is generated: aaabc Here *aaab* is for Road map and *c* stands for *Previous*.

VI. Server Program

The "Server Program" has been implemented using J2SE.

Interfaces of Server Program

The server has only one interface, which is to be used by the user to select the virtual COM port where the Bluetooth serial port server is running.

The interface is shown below.

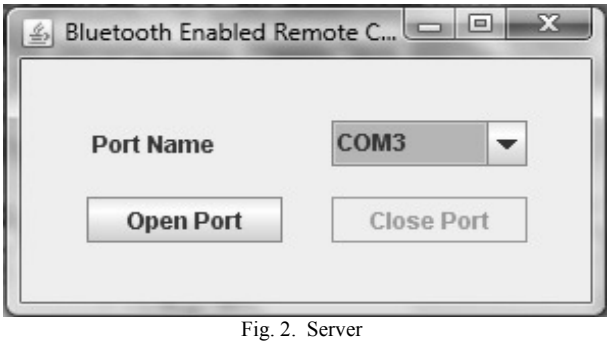

Fig. 2. Server

As soon as the user selects a COM port and presses the Open Port button the server will start and will run in the system tray hereafter. The system tray interface of server is shown in Fig. 3.

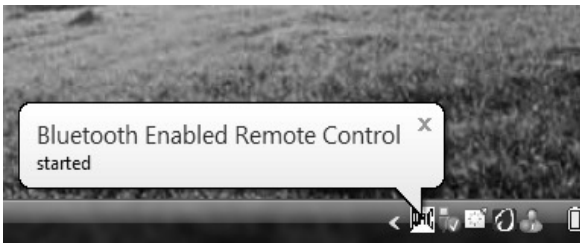

Fig. 3. Server in windows tray

VII. Operations

The commands received from the client will be decoded and the corresponding action will be performed by the server. The server has four classes namely Blue, *TrayInterface*, *W32Util*, *AlertDialog*.

Blue class is the main class of server. This class uses driver "com.sun.comm.Win32Driver" to access various functionality of the Virtual serial ports. The available COM port list is then read using a separate module of the Blue class. To open a port for reading, on selected COM port an independent module is executed. In this module a module is used for reading command on this port. In order to avoid errors exception-handling [3] has been very efficiently used. Errors are reported to the user using class *AlertDialog*.

To start reading commands in a stream an object of *InputStream* [4] class is build by calling *getInpuStream* [5] on an object of *SerialPort* class. The parameters are set on serial port using *setSerialPortParams*. The string received on serial port will be passed on to decode module of W32Util class.

In the decode module of W32Util class, command is decoded and the desired action is performed. The characters 4,3,2,1 of command are used to set the focus on the specified application window utilizing open source package *org.jawin*. The character 0 will specify the task to be performed on the application window by performing virtual key press using Robot [6] class. To open executable file the exec module of Runtime class is used [7]. To map the window handles with their title, the Map data structure of java.util library is used [8]. Taking an example of "Road map", when user selects the Road map in Remote Control (in interface three), the client will send command "aaaaz" through Bluetooth to the server.

When the server will receive the command, it will decode it and upon decoding it will open Google Map using URL command of class Runtime. When the server will receive

the command

aaaaa it will decode it and recognize that it is the search command for Google map. The server will activate the window and bring it on top of all other windows. Then it will generate Ctrl+P using keyPress module of Robot class.

VIII. Hardware Interfacing

It is also possible to control the devices connected to the LPT parallel port using Bluetooth enabled Remote Control. The data pins are turned low or high through an application made in Visual Basic.

"Inp" and "Out" are used to receive and output signals on the LPT.

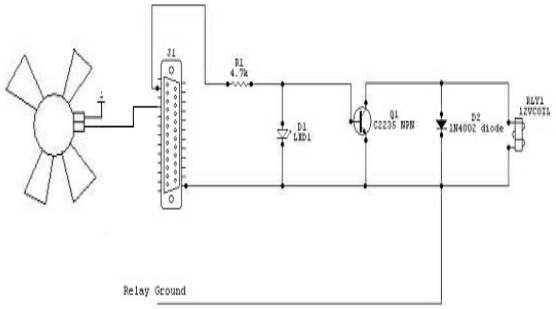

Fig. 4. Circuit to control traffic signal devices

Server calls the executables to send high and low signals to parallel port. Circuit to glow LED is shown in figure 5 [8]. The circuit to control Traffic Signals such as "Emergency" is shown in figure 4 [9]. The load will be connected to Communication Devices.

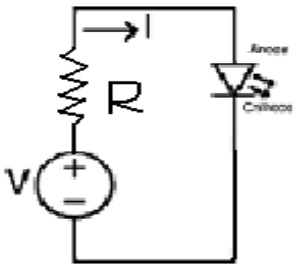

Fig. 5. Simple LED circuit diagram.

IX. Conclusions

This problem can be solved in two ways COM based solution: The COM-based solution is based on virtual COM ports created by both communicating devices. From the programmer's point of view, using a virtual COM port is similar to using a normal COM port [3].

JSR-82 based solution: This solution is based on JSR-82[10], which is the Java API defined for using Bluetooth in Java ™ ME devices. J2SE™ [11] used in Personal Computers does not support the JSR-82 API by default. There are third-party implementations for the JSR-82 API to be used with J2SE. These third party JSR-82 implementations are based on existing Bluetooth stacks in computers and usually only implement the API over Bluetooth stacks [14] services. One example of JSR-82 implementation for J2SE is BlueCove (http://bluecove.sourceforge.net/). BlueCove works with the Windows XP Service Pack 2 Bluetooth stack (and all Bluetooth devices it supports). The COM-based solution is not Bluetooth stack specific and hence could be used on a variety of Bluetooth stacks such as IVT Bluesoleil, Microsoft Bluetooth Stack (Windows XP Service Pack 2 Bluetooth stack), and Widcomm [15] Bluetooth Stack etc. This was the reason for choosing it as the solution to problem one.

We have used NOKIA 6600 as the Mobile Phone to run client. The server was tested over IVT Bluesoleil and Microsoft Bluetooth stacks. Both the components of the Bluetooth enabled Remote Control worked as expected. The operations were executed in real time. All the work was tested on Microsoft Windows 2000 and Microsoft Windows XP SP2 operating systems.

The following is list of phones which are expected to support the Client Program: Nokia 6600, 6681, 6682, 6680, 9500, 9300, 6620, 7610, 6630, 6260, 6670, 3230, 6230, 6255, Motorola A1000, Sendo X2, Siemens SK 65, Siemens S65 / S66, Panasonic X700, Sony Ericsson P900, Sony Ericsson P910, BenQ P30, BenQ P31.

X. Future Work

In the future, Bluetooth enabled Remote Control could include the capability to control the serial port [12] and USB [13] of the computer system. It could also display the Personal Computer screen on the Mobile Phone using object push service.

References

[1] M. A. Mazlan, "Stress Test on J2ME Compatible Mobile Device," *Innovations in Information Technology*, pp. 1 – 5, 2006.

[2] H. Schildt, "The Complete Reference J2SE 5," *Tata McGraw Hill*, 2005.

[3] P. A. Buhr and W. Y. R. Mok, "Advanced exception handling mechanisms," in *IEEE Transactions on Software Engineering*, Volume 26, pp. 820 – 836, 2000.

[4] S. Microsystems, "InputStream (Java 2 Platform SE v1.4.2)," modified on June 21, 2005, Retrieved on December 26, 2006, from ttp://java.sun.com/j2se/1.4.2/docs/api/java/io/InputStream.html.

[5] Sun Microsystems, Inc, "CommPort," Modified on July 02, 2005, Retrieved on December 26, 2006, from http://java.sun.com/products/javacomm/ reference/api/javax/comm/CommPort.html

[6] Sun Microsystems, Inc (2003), "Robot (Java 2 Platform SE v1.4.2)", Modified on June 21, 2005, Retrieved on January10,2007,from,http://java.sun.com/j2se/1.4.2/docs/api/java/awt/ Robot.html

[7] Sun Microsystems, Inc (2003), "java.util(Java 2 Platform SE v1.4.2)", Modified on June 21, 2005, Retrieved on January 10, 2007, from http://java.sun.com/j2se/1.4.2/docs/api/java/util/package-summary.html

[8] S. Nathan, N. Shammas, and S. Grainger, "The future of high-power conventional semiconductor based Light Emitting Diodes (LEDS) against Organic Light Emitting Diodes (OLEDS)," in *42nd International Universities Power Engineering Conference*, UPEC 2007, pp. 697 – 700.2007.

[9] R. Li ,"Home- Electrical- Control.pdf," Modified on June 10, 2004.

[10] P. D. Garner, "Mobile Bluetooth networking: technical considerations and applications" in *4th International Conference on 3G Mobile Communication Technologies*, pp. 274 – 276.2003.

[11] S. Vinoski, "Java Business Integration," in *IEEE Internet Computing*, Vol. 9, pp. 89 – 91.2005.

[12] M. R. Samady, M. R. Movahedin, M. Fakhraie, A. Zakeri, and G. Dezfuli, "A Implementation of serial port interconnections for

integrated circuits," T*he Eleventh International Conference on Microelectronics*, pp. 291 – 294, 1999.
[13] X. K. Zhu; L. X. Xu, and H. J. Yong, "USB Interface Data Acquisition System Hardware Design," in *Chinese Control Conference,* pp. 1405 – 1410, 2006.
[14] D. Groten and J. R. Schmidt, "Bluetooth-based mobile ad hoc networks: opportunities and challenges for a telecommunications operator," *IEEE VTS 53rd Vehicular Technology Conference*, Vol. 2, pp. 1134 – 1138, 2001.
[15] C. Wang; Z. Shao, and M. Fujise, "Design of upper-layer protocol emulator for SDR prototype of IEEE 802.11g and Bluetooth," in *IEEE International Symposium on Communications and Information Technology*, Vol. 2, pp. 1118 - 1121 , 2004.