# A Bluetooth Solution for Public Information Systems

Nicolae Radu MÂRŞANU, Radu Gabriel CIOBANU
Academy of Economic Studies, Bucharest, Romania
radu.marsanu@ie.ase.ro, radu.gabriel.ciobanu@gmail.com

*This paper gives insights into the opportunities offered by the Bluetooth technology. Bluetooth advertising proves to be a cheap and strong tool for enriching and improving the experience offered by a public transport system, by delivering dense and essential information about topics of interest. Alongside the Java platform, new applications can be designed and implemented to make use of the already available Bluetooth technology incorporated in devices in the target public's custody. The paper sets focus on the segment of ready to be made available content regarding general information about the routes and timetables of the vehicles integrated in a public transport system.*
***Keywords:*** *Public Transport, Bluetooth, Java, Mobile Device*

## 1 Introduction

In today's fast moving environment people tend to choose the way they're moving from one location to another after considering such aspects as speed and reliability of available means of transport. A good way of improving a public transport system from the customers' point of view is adding predictability as a main feature. This, alongside with the implementation of solutions regarding the improvement of the way other resources are used, can result in a competitive and efficient public transport system [12].

One area that has profited greatly from the rise of Bluetooth technology is that of Marketing, which gave rise to the concept of Bluetooth Advertising. The latter is included in the category of proximity marketing and relates to a method of promoting information about products and services offered by a company / institution by transmission via Bluetooth. Potential customers or the beneficiaries of services can be informed of company offerings in areas relevant to them. This method of promotion proved to be successful and has been widely implemented due to the two main features of Bluetooth technology: low cost of implementation and overall operation and interoperability [14].

## 2 Problem formulation

The practical theme addressed by this paper is represented by the implementation of a way of transmitting relevant information about the transport network, to its potential users. In order to cover all requirements, the following aspects should be addressed: firstly, the application must be easily accessible to all users. From the point of view of the necessary software suite and the availability of services, the application should be made available according to needs of the users, by their location and the time they access it. The software requirements of the application must be in line with the available hardware on the devices already held by customers. Furthermore, the application must provide a safe and dynamic way of accessing constantly updated information for the users, which requires a low bandwidth Internet connection on the server side of application. Additionally, the application must provide multi-client support; therefore it is necessary for the application to solve client requests simultaneously. Finally, the application must provide a simple and easy to understand interface for all users, regardless of their experience with computer systems and the IT field; the interface must also provide access to the desired information in a short amount of clicks and time.

The following network technologies have been found suitable to address the needs imposed by the application: Bluetooth, Ultra-Wideband (UWB), Certified Wireless USB, Wi-Fi (IEEE 802.11), WiMAX (Worldwide Interoperability for Microwave Access and IEEE 802.16), WiBro (Wireless Broadband),

Infrared (IrDA), Radio Frequency Identification (RFID), Near Field Communication (NFC), HiperLAN, HIPERMAN, 802.20, ZigBee (IEEE 802.15.4) [6].
Also, from the programming language's point of view, a solution in accordance with the imposed specifications could be constructed with the help of Python, Java Micro Edition (J2ME) and C# .NET .

## 3 How Bluetooth technology works

Once defined in its final form, the technology began to show its true potential. In addition to its status as a method of communication without cables between mobile devices and accessories, Bluetooth gave rise to the concept of Personal Area Network (PAN). Due to the way it is constructed, a strong point of this technology is that devices incorporating it have the capacity to create ad hoc networks. Thus, one can create connections involving a wide range of mobile devices, printers, access-points and personal computers for the implementation of various voice and data services, all at minimal cost [1]. Resulting from the cooperation between member companies of Bluetooth Special Interest Group (SIG), technology specifications have been defined so as to ensure full compatibility between devices that implement it, regardless of their manufacturer. In order to achieve this, the necessity to impose transmission methods, the way data is encoded and the protocol stack that is used arose. The concept of Bluetooth Profiles, which represents a series of implementation models of this technology, was also introduced, as each model implements a specific combination of the available protocols. The 3.0 version of the technology, which was launched in April 2009, stands as proof for the improvements made to the original specification and to the continuously evolving user profiles.
According to [2], the Bluetooth protocol stack can be broadly divided into two components: the Bluetooth host and the Bluetooth controller (or Bluetooth radio module). The Host Controller Interface (HCI) provides a standardized interface between the Bluetooth host and the Bluetooth controller. Figure 1 illustrates the Bluetooth host and Bluetooth device classification. The Bluetooth host is also known as the upper-layer stack and is usually implemented in software [15]. It is generally integrated within the system software or host operating system. Bluetooth profiles are built on top of the protocols. They are generally integrated in software applications and run on the host device hardware [11]. For example, a laptop computer or a phone would be the host device. The Bluetooth host would be integrated with the operating system on the laptop or the phone.

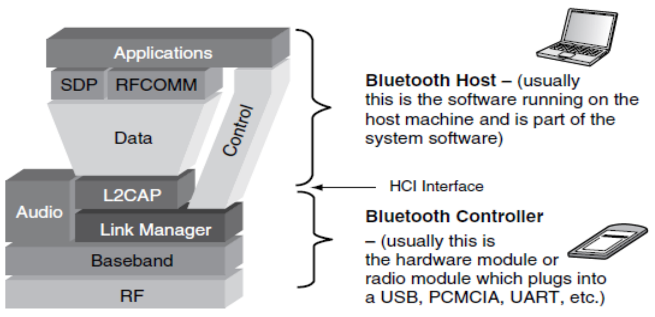

Applications
SDP
RFCOMM
Data
Control
Audio
L2CAP
Link Manager
Baseband
RF
Bluetooth Host – (usually this is the software running on the host machine and is part of the system software)
HCI Interface
Bluetooth Controller – (usually this is the hardware module or radio module which plugs into a USB, PCMCIA, UART, etc.)

**Fig. 1.** Bluetooth host and device specification [2]

As set out in [3], Bluetooth communications are realized by spread-spectrum transmissions using the frequency-hopping technique (Frequency Hopping Spread Spectrum - FHSS). The raw binary rate is 1 Mbps, using a simple frequency modulation, i.e. GFSK (Gaussian Frequency Shift Keying) to minimize the complexity of the radio part. During normal operation, a radio channel is shared by a group of devices that are synchronized by a common clock scheme and meet the same jump-in frequency. A single device called the master device can provide reference for synchronization. All other devices are considered slave devices. A group of synchronized devices form a so called *piconet*, this being the fundamental principle of communication using Bluetooth technology. Thus the Bluetooth connection is very stable, as interferences with other devices do not lead to the discontinuation of a connection, but lower data transmission speed. Bluetooth technology supports point to point connections and point to multipoint connections which can be established automatically and dynamically. Bluetooth devices can be configured to operate in a single role, but most can assume any role, depending on the model of use that is involved. Thus a master may communicate with multiple slave devices (up to seven active slave devices or up to 255 slave devices in parked state) (Figure 2).

The range of a Bluetooth equipped device depends on the power class of the radio transmitter. Most devices are equipped with Class 2 transmitters, having a nominal transmit power of 0 dBm, resulting in a range of up to 10 m in an environment without obstacles. This range is sufficient for applications designed to replace cable connections. When a higher range is needed, a more powerful radio transmitter can be used, e.g. a class 1 device, or a signal repeater, via the *Scatternet* method (Figure 2). Higher power consumption is not a problem if the transmitter is part of a fixed device. In mobile devices such as mobile phones, power consumption is very important and fitting them with a Class 2 radio transmitter to be used by Bluetooth is the only viable option [3]. Link speed, range and power level emission were chosen so that, using current manufacturing technologies, it is possible to implement a solution at low cost with low power consumption in a single chip.

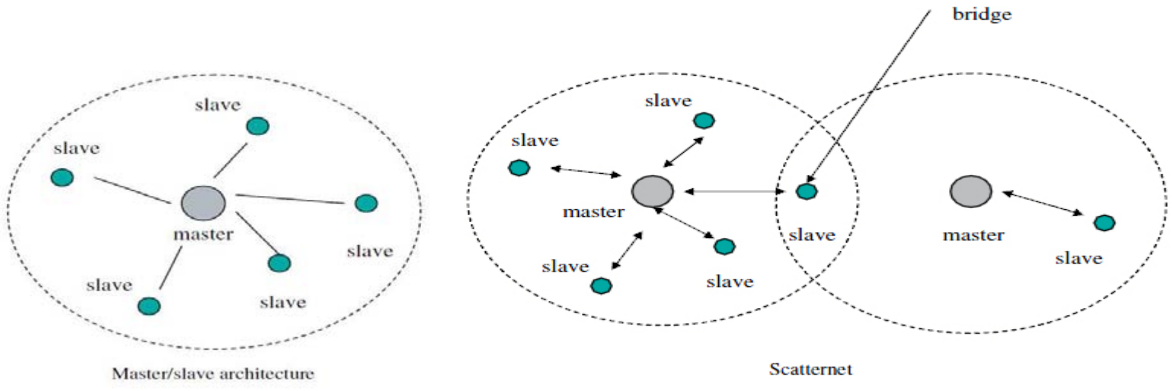


**Fig. 2.** Master/Slave and Scatternet models [3]

In addition to the technology specifications, the Bluetooth Special Interest Group defined the concept of "Bluetooth Profiles", which represents standard ways of using sets of protocols and facilities to perform certain functions. Sources [4] and [5] describe these models. A Bluetooth device may be compatible with one or more profiles. The four basic profiles are the Generic Access Profile (GAP), Serial Port Profile (SPP), Service Discovery Application Profile (SDAP) and Generic Object Exchange Profile (GOEP). The main profile, on which all other profiles are based, is GAP. It defines the general procedures needed to establish connections between devices, including discovery of nearby devices, configuring them and establishing links with security policies [13]. SDAP describes basic operations for finding services of interest found in nearby devices. SPP specifies requirements for establishing emulated serial connections between devices using RFCOMM. GOEP is an abstract profile based on other existing protocols which have been customized for Bluetooth, such as the previously defined IrDA OBEX. Therefore, Bluetooth profiles are organized hierarchically, each being built from another and bringing additional features.

- Bluetooth Advantages: easy scalability, low cost (under $ 3 per transmitter), compatible with both data and voice transmissions, low power consumption, the possibility of forming ad hoc networks, large

scale integration within the users' devices, extensive software support by many programming languages;
- Disadvantages of this technology: the relatively small range compared to other technologies, the relatively slow connection speed, shared wireless ISM bandwidth could lead to interferences with other devices, low mobility potential for users, the cost increases significantly with the increase of coverage;

## 4 Design of the application

The analysis of the technological possibilities that are suitable for the development of applications with the described features shows that implementing Bluetooth is optimal. The main arguments for choosing this technology are that it is widespread and integrated in the vast majority of mobile phones held by users. This information, coupled with the fact that the mobile phone penetration rate in Romania is 130% (according to the National Regulatory Authority for Communications) leads to the conclusion that we are dealing with a target public properly equipped in terms of the chosen technology. Thus, the need for users to purchase devices compatible with the service provided is eliminated. Secondly, in terms of coverage, given the current infrastructure provided by the stations in transport system the implementation of the solution can be done easily, in an organized and optimal way. Technical requirements are represented by a mobile device equipped with a Bluetooth transmitter and, optionally, an antenna for signal amplification, at each station. Thirdly, Bluetooth architecture offers high level security to ensure the safety and confidentiality of transmissions [13]. Also, once implemented, Bluetooth does not present additional requirements for qualification of additional staff. Applications that are easy to use by anyone anywhere can be easily made based on this technology. The need for an Internet connection for the specified application's server side is modest and can be assured through the mobile device that acts as a server, through inexpensive technologies like GPRS in the GSM network. Also, low cost and low power consumption are a plus for Bluetooth technology. Finally, the technology is receiving significant support from the software perspective, as there are many programming languages which can implement the Bluetooth protocol stack, among which C#.NET, Java ME and Python.

All three programming languages that were mentioned implement the Bluetooth protocol stack at a high, easy to use level. Writing a program for a Bluetooth device is reduced to using a series of methods that help with handling profiles and functionalities that are already included in them. However, the main criterion that should be taken into account when determining which language to use is interoperability, i.e. the ability to run the final application on a large set of heterogeneous mobile devices. The large number of potential users which own many types of mobile phones with different operating systems and hardware configurations lead to a large number of combinations with which the application must cope in terms of compatibility.

Python is supported by most operating systems for smart phones, supporting, inter alia, Windows Mobile and Symbian OS. In order to implement Bluetooth capabilities using Python, BlueZ is the tool most often used. This tool provides organized and easy to understand methods to access the Bluetooth protocol stack modules, according to profiles.

The C # programming language, when implementing the Bluetooth stack, only offers support for the Windows Mobile operating system. Its main advantage is the ability to use complex components, leading to expanded opportunities in terms of functionality and GUI appearance of programs. However, the percentage of smart phones running Windows Mobile is, in Romania, below 15%.

The Java programming language, with its version dedicated to mobile devices, Java Micro Edition, presents the user with the opportunities of an object-oriented, multiplatform programming environment. The main purpose of this high level programming language is to allow devices to dynamically load programs that use locally available resources, which may vary significantly from one de-

vice to another. As a mode of operation, a Java virtual machine installed locally interprets the code (called bytecode) generated by the compiler and previously stored in a resource archive file (of type .Jar). After the installation of an application, the Java virtual machine translates the instructions contained in the resource file into a code that is compatible with the operating system and that can be understood by the local hardware configuration. The Java implementation for mobile devices uses two sets of standard configurations: the Connected, Limited Device Configuration (CLDC) and Connected Device Configuration (CDC). Alongside the Java virtual machine, these configurations define the functionalities and functional requirements of applications written for this environment, for devices in this category. Also, the presence in a large proportion of Java-enabled phones within the market, that have at least a KVM (Kilobyte Virtual Machine), is an important asset for Java2ME [7].
Therefore, the analysis of opportunities presented by these three programming languages leads to the conclusion that the programming language most appropriate for this situation is Java2ME. As noted above, interoperability is the most important criterion, and Java provides the greatest level of platform compatibility. Because of the mechanisms of adaptation to locally available resources provided by the operating system, Java applications can run even on the simplest of the phones in terms of memory and processing power, as long as they have a version of KVM (Kilobyte Virtual Machine) installed.

## 5 The way the application is implemented

The application designed to solve the problem described above is structured in two parts, as is the functionality offered. The first part, called BlueTransportInfo, aims to provide information about the route and timetables of different means of transport. The second part, called BlueMap, allows users to navigate on a city map on which the routes of transportation means were outlined, depending on the options chosen.
The whole concept is based upon a network of uniformly distributed static Hotspots which will provide information to the client devices held by the users. All the information that can be provided to the clients is managed and initially stored on a central server. Each individual hotspot will periodically synchronize itself with the central managed server by the means of a low-bandwidth Internet connection. The integrity and accuracy of the data stored locally on the Hotspot device will be assured by these periodic synchronizations with the central server. Revision numbers and Hashed Message Authentication Codes will be used by this process in order for it to happen efficiently and reliably. From a hardware point of view, the requirements for each Hotspot are a Java enabled mobile device and a low bandwidth Internet connection.
Thus, the solution found is structured along the lines of a server – client model. The "server" side of the architecture is represented by the central data server and the mesh of Hotspots replicating that data and presenting it to the clients, while the "client" side of the architecture is represented by the application installed on the client's mobile devices. The company operating the transport system has to install at least one Hotspot per station, in order to provide an efficient mesh of information points, offering full coverage for the clients. Every time the server part of the application is started, a user must insert a certain set of credentials in order to get the application to run.
The first part of the application, called BlueTransportInfo, allows users to request general information about the route of a means of surface transport, identified by *line* number (Figure 4). The information provided is represented by a list of stations on the route and an estimated time until the next vehicle will arrive at the station area from which the request was sent. In order to be considered operational, the application's server needs to be distributed to the stations included in the transport network and the hardware devices must be installed in accordance with the requirements specified above. Also, the client must be installed on customers' mobile

phones.

In order for the server to be running the application, and employee must use his/her own set of unique credentials to authenticate (Figure 8). These credentials are managed and distributed by the company who implements and administers the application. Once the user has been authenticated, the server is operational and customers can connect to it. Statistics relating to the customers who have requested information from the server are invisible to the user, but are saved in a text log, after having been shown only on the management console. Also, as the Bluetooth transmitter only accepts 8 simultaneous connections, the application has to queue, in the *parked* mode, other devices requesting data once 8 concurrent connections have been reached. After the number of active connections drops under 8, the application "wakes up" the next device in the queue, allowing it to request data.

The client side of the application, once run, will allow customers to connect to a server through a pre-known URL, or retrieve a list of all discovered servers within range. Once these servers are shown, customers can choose to start the service of interest from the retrieved services list of each server. This implementation was chosen to allow the modular addition of new services, if necessary, on any of the servers involved. Once connected to a server, customers can send requests for information about each means of transport, identified by line number. Data received from the server is automatically stored to be reread at any time if necessary. Application menu interface is independent of the local device's menu and can be made available in three languages: French, English and Romanian. This is made possible through local resources (Figure 5).

The BlueMap application allows users to navigate on a city map on which routes of transportation means were highlighted, depending on the options that were chosen (Figure 3). The solution found is structured along the lines of a server – client model. The server allows the application to present the passengers with starting points on the map, called "Hotspots" where they can begin to navigate the map, depending on their needs. The client side allows users to browse the map of the means of transport, starting from any of the locations published by servers within their range. To be considered operational, the server application must be distributed to the stations of the transport network and hardware devices must be installed in accordance with the requirements specified above. The client side must be installed on customers' mobile phones (Figure 6).

The server side of the BlueMap application, once launched, requires the user to authenticate by entering the credentials that were distributed to them by the company operating the transport system (Figure 8). Once authenticated, the user can select from a list the sites that he/she wishes to make available for the passengers. Changing the status of a location (published / unpublished) may be done with the help of the contextual menu, which is accessible via dedicated keys on the device that is running the application. The interactions with the customers are invisible to the user, as they are displayed only on the management console and are stored in a text log, being accessible only when needed. Menu options allow publication of a list of preconfigured points of interest, adapted to each location where a server will be installed through the use of customized local resources that were deployed alongside with the application.

Once launched, the client side of the application allows users to search all information sources (servers) within range. By selecting "Find" from the phone menu, a search will begin for all sites (hotspots) listed by servers in the area (Figure 6). Once displayed, the user can choose the starting point on the map from this list, depending on the desired mode of transport. Once the map segment corresponding to the selected location has been properly received, the client can navigate in whichever direction he wants by using the contextual menu (Figure 7).

The deployment method chosen for the two parts of the application is via a web server, so that users can download the client side software by following a URL which will be published, alongside with details about the application and suggestion to turn on their Bluetooth transmitters, in all stations where the server was installed.

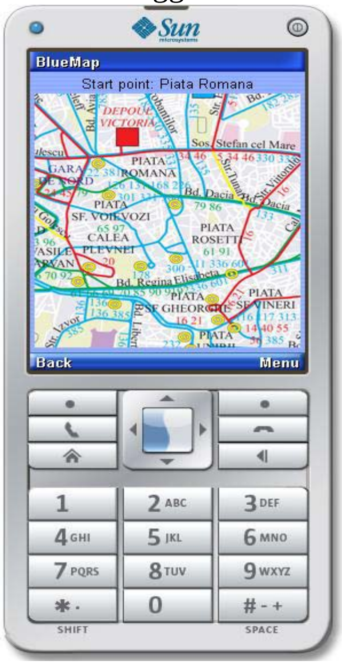
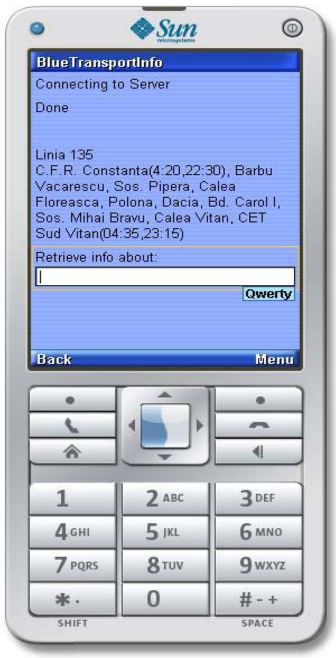

BlueMap
Start point: Piata Romana

**Fig. 3.** BlueMap content


BlueTransportInfo
Connecting to Server
Done

Linia 135
C.F.R. Constanta(4:20,22:30), Barbu Vacarescu, Sos. Pipera, Calea Floreasca, Polona, Dacia, Bd. Carol I, Sos. Mihai Bravu, Calea Vitan, CET Sud Vitan(04:35,23:15)
Retrieve info about:

**Fig. 4.** BlueTransportInfo content

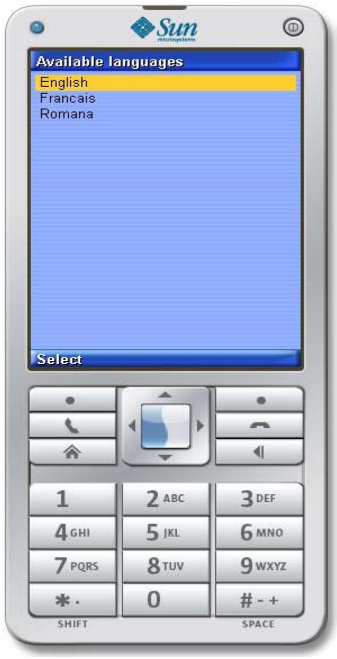

Available languages
English
Francais
Romana

**Fig. 5.** Language resources

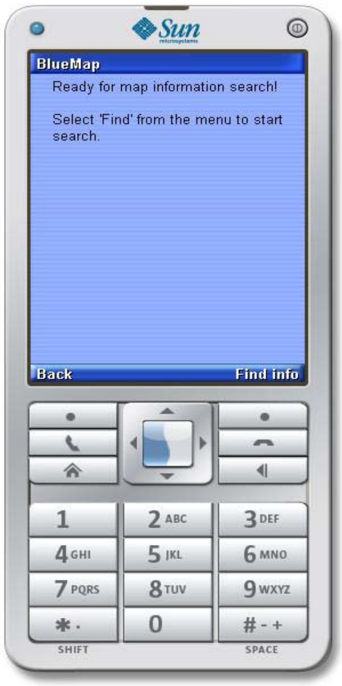

BlueMap
Ready for map information search!
Select 'Find' from the menu to start search.

**Fig. 6.** BlueMap client

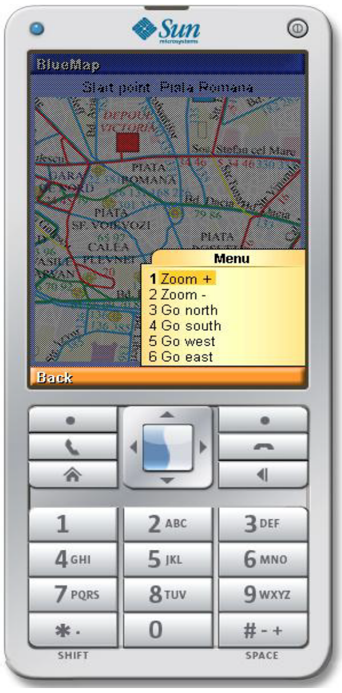

BlueMap
Start point: Piata Romana

**Fig. 7.** BlueMap client context menu

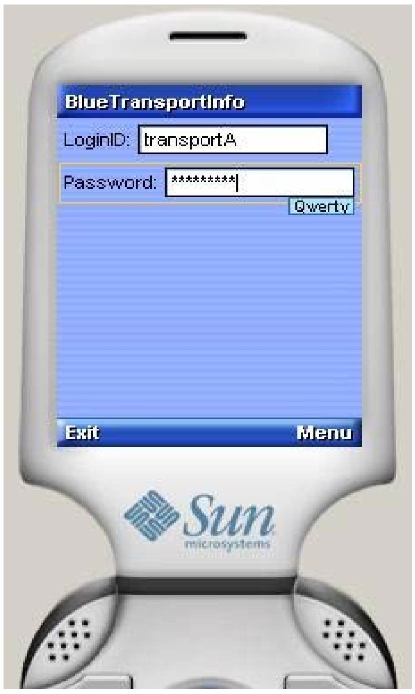

BlueTransportInfo
LoginID: transportA
Password: *********

**Fig. 8.** Application login

Hardware requirements of the application:

**Table 1.** MIDP 2.0 specifications [7]

| | |
|---|---|
| **DISPLAY:** | Pixels: 96x54 |
| **DISPLAY DEPTH:** | 8-bit |
| **PIXEL SHAPE (ASPECT RATIO):** | approximately 1:1 |
| **INPUT:** | one- or two-handed keyboard or touch screen |
| **MEMORY:** | 256 KB of non-volatile memory for the MIDP components<br>8 KB of non-volatile memory for application-created persistent data<br>128 KB of volatile memory for the Java runtime environment |
| **NETWORKING:** | Two-way, wireless, possibly intermittent, with limited bandwidth |
| **SOUND:** | The ability to play tones, either via dedicated hardware or via software algorithm |

Besides the requirements that were previously described by the structure of CLDC and MIDP (Table 1), the application requires from the server side about 4 MB of non-volatile storage space for any static resources.
Software requirements, besides the hardware requirements specified earlier on, are:
JSR 30 - Connected Limited Device Configuration (CLDC) 1.0
JSR 118 - Mobile Information Device Profile (MIDP) 2.1
JSR 82 - Java APIs for Bluetooth
In the server – client model, a Bluetooth service is an application that acts as a server, providing assistance to client devices through Bluetooth channels. This support is reflected by specific processing that cannot be performed on the client host device. All services provided by a server are included in the database of descriptors (SDDB), created and managed locally using the SDP protocol, which will be made public by the server once it isn't empty anymore and will be transmitted in the Service Discovery operation started by a client. This database consists of *ServiceRecord* elements, which store information necessary for a client to connect to the server and the processing categories provided by the service.
The exchange of information between the client and the server is done with SPP Protocol (Serial Port Profile), the Bluetooth connectivity being achieved by using RFCOMM. The latter comes as a general emulation of a serial connection through a RS-232 port, sending data in the form of a stream. The server is identified by its unique single UUID. To add an additional level of security, in addition to those provided by the Bluetooth technology and administered by the Bluetooth Control Center (BCC) on the host device, the application sends encrypted data by using the Data Encryption Standard (DES) symmetric key algorithm, with 56 bit key on 64 bit blocks. The application uses local resources available on the mobile device at the graphical interface level, but dynamic resources were introduced to provide greater versatility in key points. Thus, the language interface can be implemented independently of that required by the device menu and the content offered by the application can be interpreted and customized dynamically, without the need for intervention (Figure 5). This leads to a high level of scalability, given a large number of devices.
Methods *startApp()*, *pauseApp()* and *destroyApp()* are essential for any MIDlet, providing entry and exit points of application in the context of the running environment. The *CommandAction()* method, derived from *CommandListener* interface, helps by intercepting and interpreting user commands (the events generated by them).
During the process of discovering other de-

vices in the area and the services provided by them, the arising specific events will be interpreted by the DiscoveryListener object. The *DeviceDiscovered()*, *inquiryCompleted()*, *serviceSearchCompleted()* and *servicesDiscovered()* methods are used to interpret the triggered events. The search for devices in the area is initiated by the *LocalDevice* and *DiscoveryAgent* objects. The devices covered by the search are those that have the GIAC parameter as an asset (are generally discoverable). Once the devices are found or the search operation ends (the default timeout of 30 seconds) specific events will be triggered in order to be processed. After ending the search for devices within range, the application will target actions to recover the list of services from each device found earlier. The methods of interest are *servicesDiscovered()* and *serviceSearchCompleted()*. The information received after analysis of specific events will be filtered by criteria imposed by application, such as the existence of certain services, identified by hexadecimal numbers. The result of these actions is represented by a list of devices, of interest for the user due to the services they offer.

The actual handling of data used by application's specific processing is done using serial connections, in this case emulated by the SPP protocol (Serial Port Profile). Relevant to the operation and emulated by SPP, is the function *Connector.open()* from the RFCOMM protocol. Through the returned notifier object (StreamConnectionNotifier object type) the *acceptAndOpen()* function can be launched, which enables the server to wait for connection requests from the clients. Once it has observed a connection request from a client, this function creates an object of type *StreamConnection* through which the server will direct the flow of information to the client who initiated the connection. All the application related content (section map or text) is represented as arrays of bytes. The server-side application establishes a new session by launching a new thread (process) for each connection request by a new client. This ensures the multiclient capabilities required by the application. Content requests that were launched by each client are processed in separate threads simultaneously and the content is stored on the server and shared by the running sessions. Once the client closes the connection, the server is automatically notified and it closes the session with that client.

## 6 Conclusions

In this paper, the Bluetooth wireless technology was described in the context of small area networks. In response to the problem addressed by the application, Bluetooth has proven its superiority over other technologies examined. Beside the Java2ME programming language, this technology offers a multiplatform solution with the highest degree of interoperability and versatility. The large scale availability of Bluetooth-enabled devices in the target audience's custody, alongside with the versatility of the Java platform leads to a simple and adequate solution to the problem.

The scope of the BlueTransportInfo / BlueMap suite of applications is to inform the users of a transport system about vehicle routes and timetables.

The BlueMap application allows users to navigate on a city map, on which the routes of means of transport were outlined, depending on the selected options. The solution found is structured along the lines of a server – client model. The server needs to allow the application to publish a series of starting points on the map, called "Hotspots" where the users can begin to navigate the map, depending on their needs. The client side of the application allows users to browse the map of available means of transport, starting from any of the hotspots published by the servers in their range. Simultaneously, the BlueTransportInfo application allows users to request general information about the route of a means of surface transport, identified by the line number. The information provided is represented by a list of stations on the route and an estimated time until the next vehicle will arrive at the station from near which the request was sent. To be considered operational, the application server needs to be distributed to the stations within the transport network, and

hardware devices need to be installed in accordance with the defined requirements; the client application must also be installed on customers' mobile phones. The implementation of this application, from a technical point of view, can be observed using the attached Java documentation in HTML format, called Javadoc.
The content provided by the BlueTransportInfo / BlueMap suite is simple and easily accessible to users, regardless of their knowledge regarding the IT sector. The data presented to the normal user is the key data about transport system, complementing the traditional means of informing users (posters, billboards, etc.). The presented system is dynamic, as the data to be made available to passengers is constantly updated in order to be accurate. In the context of applications for distribution of public information, the presented program comes as a complement to already implemented solutions for transportation systems. Along with methods of measuring traffic and monitoring the performance of the vehicles involved, this application leads to a series of mechanisms that can ensure a more reliable transport, predictable and easily adaptable to different situations. From the user's point of view, having a dependable application that can provide information about routes and schedule at any time is superior to any traditional mean of gathering such information (like a map, for example). Considering how widespread the technologies required by this application are, most users don't have to do anything more than download the .jar file from the website of the company implementing the solution.
The deployment method chosen for the BlueTransportInfo / BlueMap pair of applications is easily accessible for anyone who has an Internet connection. Because of the limited necessity of hardware and software type resources, a very high percentage of mobile devices held by potential users are covered. The modular structure and easy to access components delivered via a multiplexed local Bluetooth controller ensure that the submitted application will not interfere with other applications run by users and create discomfort for them.
Thanks to Bluetooth technology and the considerable support that is provided for the Java platform (in particular for the segment dedicated to Bluetooth - JABWT) the application can be easily updated, in order to keep it in tune with trends in the mobile phone market. With the trend of global integration of the WiFi (802.11) standard the 3.0 version of Bluetooth was released in April 2009. This version integrates support for this standard and has improved technical performance and characteristics [8]. The software support level is also constantly evolving, keeping up with the opportunities offered by the hardware it runs on. In the near future, improved versions of CLDC and CDC configurations and profiles such as MIDP are expected, addressing the shortcomings presented by the current versions. As the processing power is becoming more powerful on mobile devices and as the volumes of volatile and non-volatile memory increase as they become cheaper, the trend leads to the creation of new types of applications, which combine more complex profiles, providing a series of solutions to problems that become increasingly diverse.
In terms of security, the implemented solution offers high level security due to mechanisms specific to data transmissions via Bluetooth (Frequency Hopping Spread Spectrum). Also, the three levels of software security requirements of this technology, each with its own mechanisms, ensure a stable and secure connection between any two devices. Additionally, the Java2ME programming language improves the level of security, due to the way the software is installed and run. Also, general traffic is secured by using the DES encryption algorithm.

## References

[1] ***The Bluetooth technology official website http://bluetooth.com/English/SIG/Pages/default.aspx

[2] T. J. Thompson, P. J. Kline, C. Bala Kumar, *Bluetooth Application Programming with the Java APIs Essentials Edition*, Morgan Kaufmann, 2008, pp 8-9

[3] H. Labiod, H. Afifi, C. De Santis, *WI-FI, BLUETOOTH, ZIGBEE AND WIMAX,* Springer Verlag, 2007, pp 75-108
[4] *** Bluetooth Specification – Profiles http://www.bluetooth.com/English/Technology/Works/Pages/Profiles_Overview.aspx
[5] STMicroelectronics, *Bluetooth profiles overview*, 2002, pp. 2-19
[6] *** Bluetooth compared to other technologies http://www.bluetooth.com/English/Technology/Works/Pages/Compare.aspx#1
[7] A. N. Klingsheim, *J2ME Bluetooth Programming,* Department of Informatics, University of Bergen, 2005, pp. 21-33
[8] *** Wikipedia - Bluetooth 3.0 http://en.wikipedia.org/wiki/Bluetooth#Bluetooth_v3.0_.2B_HS
[9] R. Mârşanu Ş.a., *Calculatoare,* Ed. Tribuna Economică, Dec. 2009, pp. 302-305
[10] R. Mârşanu, „Tehnologii wireless. Bluetooth", *Revista Tribuna Economică,* No. 27, 2010, pp. 22,23,34
[11] R. Mârşanu, „Aplicaţii ale tehnologiei Bluetooth utilizând protocolul OBEX", *Revista de Studii şi Cercetări de Calcul Economic şi Cibernetică Economică*, vol. 44, nr.1-2 din iulie 2010, pp 69 – 73
[12] J. LeBrun, C. Chen-Nee, *Bluetooth Content Distribution Stations on Public Transit*, 2008, Dept. of Electrical and Computer Engineering University of California
[13] C. Gehrmann, J. Persson, B. Smeets, *Bluetooth Security,* 2004, Artech House, pp 19-42
[14] S. Scott, *The Wireless Future: An Overview of Potential Outcomes And Trends,* 2010, FT Press, pp. 3-18
[15] R. Mârşanu, „Descrierea şi securitatea standardului BLUETOOTH", *Revista Tribuna Economică*, No. 29, 21 July 2010, pp. 22 -24

**Radu MÂRŞANU** is professor at Computer Science Department, Faculty of Cybernetics, Statistics and Economic Informatics from the Academy of Economic Studies Bucharest. He has graduated Faculty of Cybernetics, Statistics and Economic Informatics in 1978 and he holds PhD diploma in 1993. He is the author of 42 books and university courses in the domain of informatics. In 2007, he received from graduates of the Faculty of Cybernetics, Statistics and Economic Informatics, diploma of excellence in teaching activities for exceptional specialized knowledge. His scientific research activity include over 43 articles and studies published in the journals of international scientific conferences or in professional journals rated by CNCSIS, indexed in international databases, among them two articles are ISI rated. He participated in 38 research projects as director (two projects) or as team member. He is member of the Informatic National Commission of the Ministry of Education Research Youth and Sports from 1995, member of INFOREC and UPI professional associations. Areas of professional competence are: Computer Science and Networks, Operating Systems, Economic Informatics, Technology of Databases, Technology of Web applications, Management information systems, E-learning, E-commerce.

**Radu Gabriel CIOBANU** has graduated the Faculty of Cybernetics, Statistics and Economic Informatics, Economic Informatics specialization, within the Academy of Economic Studies of Bucharest in 2010. As of 2010, he completed the Cisco Certified Network Associate course, and he is currently working to obtain the CCNA Certification. His fields of interest include Computer Networks and High-level Programming Languages.