Linux software and Bluetooth: a formula to improve accessibility by using Interactive Voice Response systems

Nuno T. Almeida, Emanuel Ribeiro
Institute for Systems and Computer Engineering of Porto - INESC Porto
Department of Electrical and Computer Engineering, Faculty of Engineering, University of Porto
Rua Dr. Roberto Frias, 4200-465 Porto, Portugal
Email: nalmeida@fe.up.pt, ee99083@fe.up.pt

***Abstract*— When accessing and interacting with regular electronic devices or informatics systems, users frequently depend on the sight sense because the information is, consistently, presented in a visual format, despite the feasibility to use additional formats, such as audio. It also happens that the access to the information systems is triggered by the action of the users, hence requiring an awareness of the surrounding environment. These two facts lead to an exclusion of a considerable number of citizens, such as the visually impaired ones, from a fair access to the so-called Information Society.**

The development of automatic user-aware Interactive Voice Response (IVR) systems can be an interesting solution to this issue, particularly, if the communications are done in a private manner and a low-cost implementation is achieved. In a reply to this challenge, it was formulated a pervasive user detection and communication concept, based on Bluetooth technology and Linux software modules, allowing improved interaction between personal area information systems and mobile user devices. To demonstrate the validity of the concept, it was developed a personal area IVR system, based on Linux, being the private interaction with the users made by means of the Bluetooth technology.

Keywords- Interactive Voice Response (IVR) systems; Bluetooth; Linux; "btsco" software; BlueZ protocol stack; ALSA sound system.

I. INTRODUCTION

Nowadays, in what concerns the interaction of general users with large/medium size organizations (corporations, enterprises, etc.), Interactive Voice Response (IVR) systems have a very important role, since they operate as the first line answer to the broad user/consumer questions. The conception and the implementation of a good IVR system improve the satisfaction of final users, particularly in terms of permanent availability and data discretion. From the point of view of the organizations, besides some savings in man power, it is a way of showing commitment to modern technological solutions and in giving an answer to different user needs [1].

The implementation of IVR systems become even more important when considering that almost all information is presented in a visual format, hence excluding relevant groups of citizens, such as the visually impaired ones. As so, it is rather important to increase the number of systems providing information in an audio format, additionally to a visual one [2] [3]. In this aspect, the use of Linux open platforms can greatly help the deployment of such systems.

In addition to the use of audio formats, and in order to assure information access handling equality, it is also important to perform the communication in a private and a discreet manner, along with the use of fairly accessible electronic devices. This last condition can be fulfilled using devices equipped with the Bluetooth radio technology [4].

In fact, Bluetooth is a very well implemented personal area network technology and, besides conventional cable replacement, it allows new forms in the process of exchanging or accessing information. Effectively, the attribute of mobility, of almost all devices equipped with Bluetooth, has stimulated the emerging of other types of service, of more dynamic nature. One of the features, where this dynamic is more noticeable, is the possibility to use Bluetooth devices, initially conceived for one determined application, in contexts related with the current location of the user, e.g., within an airport, a train station, an organization premises or a museum. Consequently, new informative services and new forms of access can now emerge.

The novel concept, herein presented, deals with situations where is intended the creation of spontaneous communication channels, between generalized infrastructures of data and the Bluetooth mobile user devices. Accordingly, such spontaneous communications involve an automatic user-aware context-based exchange of information, and occur when the mobile devices approach a duly prepared data infrastructure. The goal is, therefore, the establishment of an interactive environment providing contextualized information to nearby Bluetooth systems and devices.

To demonstrate the above proposed concept, in a context particularly helpful to the visually impaired citizens, it was defined a framework involving the implementation of an IVR system personal area, using machines with Linux, as servers, and plain Bluetooth headsets, as terminals. Fig. 1 shows the scenario example of the IVR system personal area, using Bluetooth as the main communication technology.

After this introductory section, the remaining paper is organized into five more sections. The section two describes the Bluetooth profiles which are particularly relevant to the accomplishment of the defined work. In the section three is exposed the Bluetooth software included with Windows XP

and Linux distributions, together with some Bluetooth hardware devices. The section four is focused in the software modules development, being the system demonstration explained in the section five. The paper ends with the conclusions and the references.

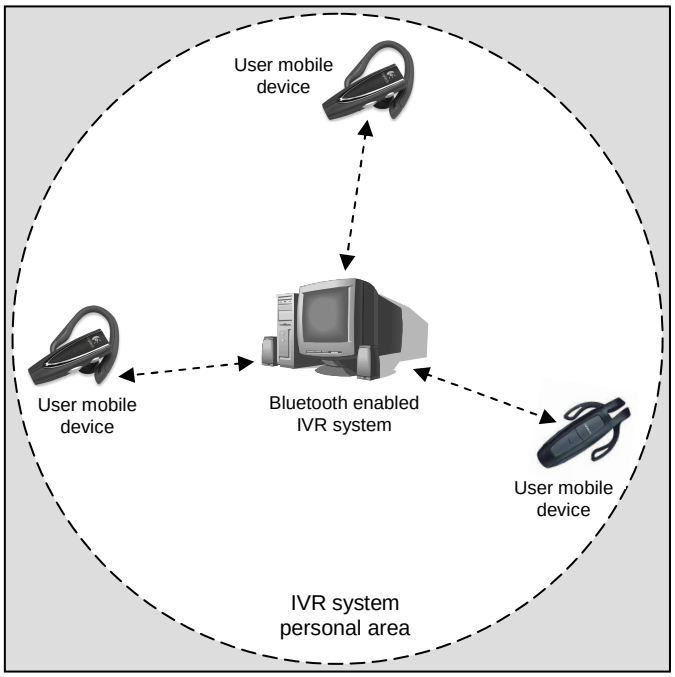

Figure 1. Scenario example of the Interactive Voice Response (IVR) system personal area, using Bluetooth communications technology.

II. WORK RELEVANT BLUETOOTH PROFILES

To enable improved compliance, among applications and devices, Bluetooth standard sets up various kinds of *profiles.* These profiles define the protocols and the procedures to be executed by the applications and the devices.

For the purpose of this work, the relevant Bluetooth profiles are the *Service Discovery Application Profile* and the *Headset Profile*. The Fig. 2 shows the related structure for these and some other important Bluetooth profiles.

A. Service Discovery Application Profile

In the Bluetooth standard is defined a Service Discovery Protocol (SDP) which is used to locate the available services, in the neighborhood of a Bluetooth enabled system or device. After the identification of these services, the Bluetooth system or device can start the request of intended ones.

Although SDP is not directly involved in the access process to the services, returned information is very important in order to select correct Bluetooth protocol stack segments.

Objectively, SDP offers direct support for the following type of inquiries:

- search services by class type,
- search services by attributes,
- and, services browsing.

Inquiries can be carried out in particular chosen devices or in all devices found in the neighborhood. In both situations, Bluetooth devices have first to be connected and just then inquired for supported services.

B. Headset Profile

Within the proposed framework, the most important Bluetooth profile to take into account is the Headset Profile. This profile defines the protocols and the procedures that must be implemented by the so-called “Ultimate Headset” devices, enabling full-duplex audio. Besides headsets, such profile implementation can be found in Bluetooth enabled mobile phones and personal computers. The Headset Profile depends on the Serial Port Profile and this one depends on the Generic Access Profile, as depicted in the Fig. 2.

In the Fig. 3 it is shown the protocol stack and the entities defined by the Headset Profile for the Bluetooth communication devices. The *Baseband*, the *LMP* (Link Manager Protocol) and the *L2CAP* (Logical Link Control and Adaptation Protocol) correspond, basically, to the layer 2 of the OSI model. The *RFCOMM* (Radio Frequency Communication) emulates a serial line interface. The SDP is the already referred protocol.

In the upper layers of the Headset Profile it can be found a Headset Control entity, which is responsible by the headset signaling. Most of this signaling is composed by *Attention Sequence* (AT) commands. Although not visible in the Fig. 3, it is assumed that the Headset Control has access to some low-level procedures (e.g., establishment of Baseband links). The protocol stack is completed with suitable applications in the Audio Gateway side (audio port emulation) and in the Headset side (audio driver).

In the context of the undertaken work, the Audio Gateway side stands in a PC with a Bluetooth USB dongle, communicating with a normal Bluetooth headset, in the Headset side.

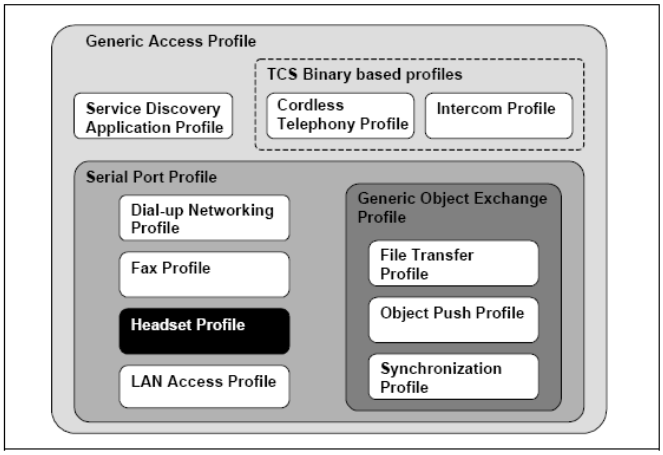

Figure 2. Bluetooth profiles related structure.

Audio communications rely on the establishment of full-duplex *Synchronous Connection Oriented* (SCO) baseband links. Audio Gateway controls these audio communications, having the Headset the opportunity to send related AT

commands to the Audio Gateway. If valid, those AT commands are properly recognized and the respective sub-actions executed. In the Fig. 4 it is represented a SCO link establishment between an Audio Gateway and a Headset.

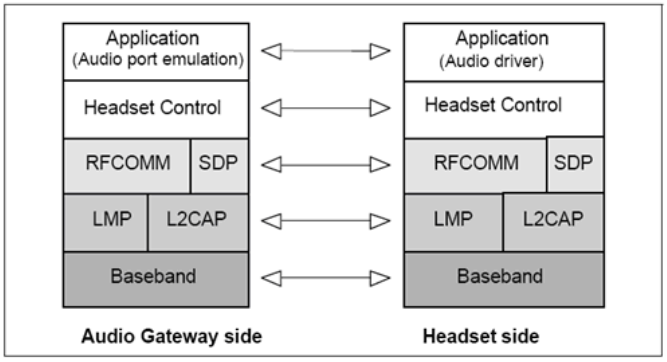

Figure 3. Bluetooth Headset Profile: protocol stack and entities defined for the communication devices.

III. BLUETOOTH SOFTWARE AND HARDWARE DEVICES

A. Windows XP related Bluetooth software

A Bluetooth protocol stack is a software module which enables the interaction with other Bluetooth devices. In the Windows operating system two protocol stacks are currently available: the Bluetooth protocol stack from Microsoft, which is supplied with Windows XP Service Pack 2, and the Bluetooth protocol stack from Widcomm.

The protocol stack of Windows XP SP2 has some limitations due to the reduced number of supported profiles. In particular, it does not support the Headset Profile. Thus, a PC with the original Windows XP Bluetooth protocol stack is unable to communicate with Bluetooth headsets.

On the other hand, Widcomm (acquired later by Broadcom) licenses its Bluetooth software to the great majority of Bluetooth device manufacturers, and it includes the Headset Profile. Therefore, it is possible to install the drivers of Widcomm, which come along with many Bluetooth devices, thus enabling the use of Bluetooth headsets with Windows.

This solution is feasible from user's point of view but, concerning programming and the development of software solutions, the access to Widcomm's protocol stack is rather problematic. This fact leads us to investigate Linux as a possible alternative.

B. Linux related Bluetooth software

1) BlueZ protocol stack

The official Bluetooth protocol stack of Linux operating system distributions is named *BlueZ* [5]. The BlueZ stack provides support to the Bluetooth core layers and protocols and is included in Linux series 2.6. Unfortunately, as in Window XP protocol stack, BlueZ does not support the Headset Profile and any type of audio SCO synchronous links.

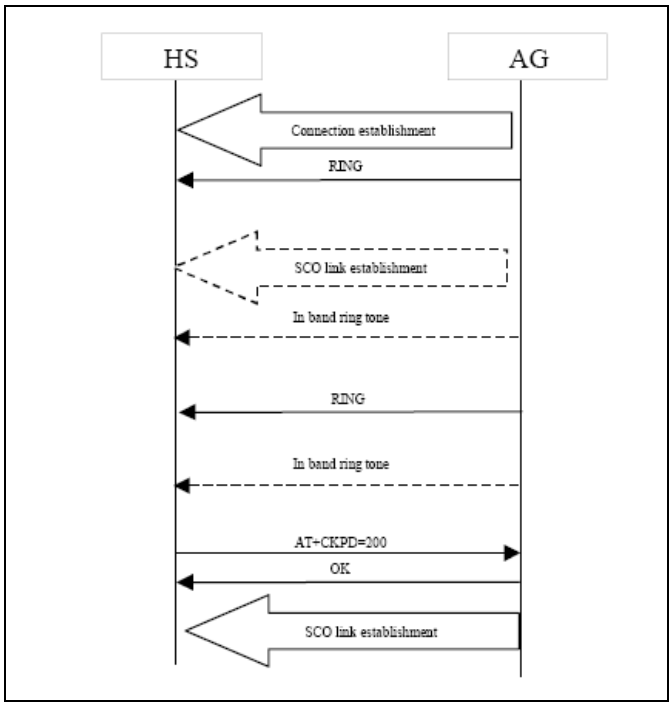

Figure 4. Bluetooth Headset Profile: SCO link establishment, initiated by the Audio Gateway (AG) equipment towards the Headset (HS) device.

To overcome this limitation, in the BlueZ protocol stack, several software programmers have developed, in open-source code, the so-called "Bluetooth-ALSA Project" [6]. This project deals with the development of Linux software, in order to enable Bluetooth SCO connections. The principal aspects of this software are referred just above.

2) SCO connections (Bluetooth-ALSA Project)

Concerning the software which enables SCO connections in Linux, this was developed by Bluetooth-ALSA Project and is based on the BlueZ protocol stack and on the "Advanced Linux Sound Architecture" (ALSA) sound system [7]. This software is entitled "*btsco*" (Bluetooth Synchronous Connection Oriented) and is available to download at [8].

"Btsco", besides installing the software which manages all audio and SCO links, creates a Bluetooth sound module likely of being added to Linux kernel, by using the "modprobe" command (modprobe "snd-bt-sco"). That sound module is, in fact, a kernel's ALSA driver which uses the BlueZ stack to communicate with headsets. Thus, when using "btsco", it becomes possible the connection to Bluetooth headsets and the exchange of quite satisfactory voice quality audio, through the use of the ALSA sound system.

Regarding the selected Linux distribution, the choice was the Kubuntu 6.06, after some installation and operational problems with SuSE 10 and Fedora Core 5. Besides all, Kubuntu 6.06 distribution includes the "Kbluetoothd" program, which assists the creation and the administration of Bluetooth networks and manages the respective connections.

C. Bluetooth hardware devices selection

1) USB dongles

All USB dongles available in the market are designed to operate with Windows, being rare the products specifically designed to work with Linux. Nevertheless, it is possible to find several USB dongles, not intended to be used with Linux, still functioning in perfect with this operating system. Nevertheless, the selection of an USB dongle, to operate with a certain Linux distribution, requires always some previous practical assessment.

In the scope of this work, two USB dongles were used, connected to the same Linux computer. The first one was an Anycom USB-200, which was used to make the Bluetooth neighborhood scanning, being the other a SMC BT-10. This last was used to establish the SCO (audio) connections with the Bluetooth headsets attending in the surrounding area.

2) Bluetooth Headsets

The choice of the Bluetooth headsets, like the USB dongles, is easier in Windows than in Linux. In this operating system, it is essential to know the compatibility of the headset device with the Linux distribution (BlueZ stack) in use, as well with the "btsco" Linux software. Information relative to headsets compatibility with Linux distributions can be found in Bluetooth-ALSA Project homepage.

A distinctive characteristic, in the headset selection process, is the number of available buttons. Between the two used models, the Nokia HS-11w (having four buttons, including a *power-on* and a *power-off*) and the Logitech HS04 (having just two buttons), it is noticeable the increased versatility of the Nokia headset. Concerning the maximum radio range, all Bluetooth headsets are of class 3, that is, they have a maximum range of 10 meters.

IV. Linux Software Modules Development

The primary approach to the development of the new Linux software modules, necessary to the intended IVR system, was to perform the adaptation and the evolution of some related software code. As previously referred, one part of that code was the "btsco" software, created by the Bluetooth-ALSA Project.

That code is written in "C" language, making practicable the inclusion of the functionalities and the automatic procedures, intended for the new software. Some of these improved functions are concerned with the automatic scanning of the IVR system neighborhood, the detection of commands by analyzing pressed headset buttons and the launch of voice dialogs simultaneously with the establishment of the SCO links.

A. Adaptation and expansion of "btsco" functionalities

As previously referred, the open-source "btsco" code was the development basis for the new software modules. A first change, to the original software, was to enable the new software to work in cycle, allowing, in this way, successive connections of distinct headsets. Another modification, which expands the functionalities of the original program, is the redefinition of the routine which detects the headset buttons, in order to enable a broaden user–system interaction. (referred ahead in subsection D.)

The other main functions, added to the "btsco" code, were the automatic scan of Bluetooth devices, the establishment of the RFCOMM (Radio Frequency Communication) connection and the automatic trigger process of the audio files.

B. Main program implementation

The developed main program is composed by two distinct software modules, entitled "*ivrBTSCO*" and "*ivrASound*". From "ivrBTSCO" module are launched two main *threads*, one to perform the permanent scanning of the Bluetooth devices, in the neighborhood of the IVR system, and another to carry out the establishment of the SCO connections.

After a scanning, returning a valid Bluetooth MAC address, the "ivrBTSCO" module automatically creates a RFCOMM link. In the following, the "ivrASound" module is launched by the "ivrBTSCO", starting the execution of an IVR dialog menu. As soon the "ivrASound" module has an audio file ready to play, the "ivrBTSCO" opens a SCO connection, in order to transmit that audio file.

The "ivrASound" module, besides enabling the navigation of the user in the IVR's dialog menu, it manages the repetition of the messages, when the user doesn't interact in the defined time window. In fact, at the end of each played message, it is launched a thread to detect the amount of elapsed time, without a command reply from the user. In case of no user answer, the "ivrASound" launches the repetition of the last played message, at most, two more times. If the absence remains, it finishes by carrying out an ending message. The ending process can also be initiated by the user, when properly selected in the dialog menu. In both situations, after the execution of the ending message, the respective "ivrASound" module stops and it is extinguished the RFCOMM link, previously created by the "ivrBTSCO" module.

In summary, the main program execution implies a continuous running of the "ivrBTSCO" module, being the "ivrASound" module launched only when there is a connected Bluetooth headset device. Concerning the structure of the IVR dialog menu, further details of the implemented IVR can be found in section V.

C. Scanning routine

As mentioned above, the search of Bluetooth devices, in the neighborhood of the IVR system, is carried out by a thread launched by the "ivrBTSCO" module. This developed C code, besides a standard basic scan, it is responsible by the continuous operation of the scan process, by the handling of the results and by the selection of detected Bluetooth devices. Therefore, that thread is constantly running having, always, an updated track of the devices in the surrounding area.

In the tested configuration, the USB dongle, exclusively assigned to this task, was the Anycom USB-200. In fact, by assigning distinct USB dongles to the scan and to the communication functions, it is possible to achieve a better management and a superior performance to the system.

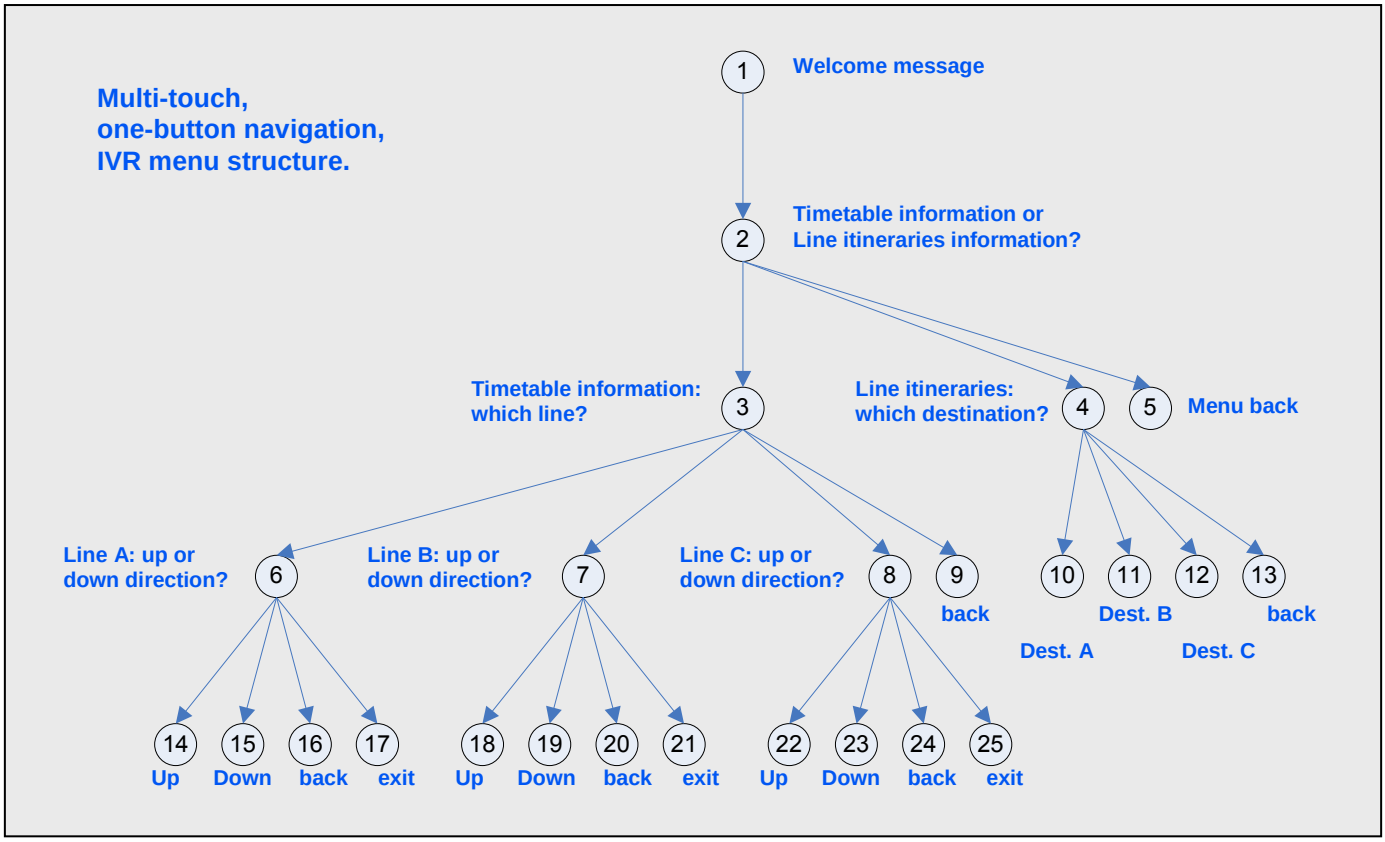

Figure 5. Demonstration of the implemented multi-touch, one-button navigation, main IVR menu structure.

D. *User-system interaction*

The user-system interaction is done by applying simple touches in the headset device button(s). Each of these touches corresponds to the Attention Sequence command "AT+CKPD=200", being the selected action dependent on the number of received commands, within a certain time-window.

By limiting the maximum number of admissible touches in a sequence, and by specifying the maximum time interval, between successive touches, it becomes practical the navigation within an IVR menu. Therefore, using the number of "AT+CKPD=200" commands which are sent, the respective menu selection is performed. That is, if only one AT command is received, it corresponds to option 1, two AT commands in a row, option 2, and so on.

In the present case, the detection function to a pressed headset button is performed by the main "ivrBTSCO" module. As a result, this module includes all the detection process of the "AT+CKPD=200" commands but also their handling.

Upon the detection of the end of an "AT+CKPD=200" sequence, the "ivrBTSCO" module evaluates the number of respective AT commands, that is, it gets aware of the option indicated by the user. On the following, it passes that information to the "ivrASound" module, which continues the IVR menu execution.

Due to practical reasons, for each menu selection step, the maximum number of options should be no more than four. Idem, the maximum time interval, between successive touches, should be around one second.

With the above tuning, this kind of multi-touch, one-button user–system interaction represents a low cost and a fair reliable solution when compared with voice recognition systems. In fact, these ones are much more expensive and present some weaknesses when dealing with a heterogeneous universe of users.

V. System Demonstration Scenario

To prove the validity of the developed work, it was prepared a demonstration scenario, simulating an information access point in a metro station, providing timetables and line itineraries information.

For each station, the line itineraries consist in relatively static information, but timetable information includes the dynamic indication, for each line/direction, of the remaining time of each vehicle arrival.

In this scenario, it were considered three lines/destinations (A, B and C), due to the established limit of four successive touches in the headset button. The implemented main IVR menu structure is depicted in the Fig. 5, being the IVR navigation possible using a single button.

The demonstration relies, thus, in a nearby server, loaded with the developed IVR system, and some mobile Bluetooth headsets. After server initialization, the headsets are detected when they are within a range of, approximately, 10 meters and

in pairing mode. After the connection of a headset, it is possible the interaction of the user with the IVR system, being allowed the respective menu exploration.

After the end of a connection, if a new headset is in the neighborhood of the IVR system, it will be linked, and the previously described process repeated. If there aren't any more headsets in the neighborhood, the IVR main program continues to scan the surrounding area, in search of new Bluetooth headsets.

VI. CONCLUSIONS

When considering communication processes between user mobile devices and fixed computing/electronic equipment, there is still a long way to improve the daily quality of life of general users. This is even more important, and of unquestionable social fairness, when thinking in the visually impaired citizens, which can not rely on the sight sense.

In accordance with these facts, and by using IVR system concepts, a substantial help can arise from the creation of human-machine interfaces based on voice dialogs. Furthermore, and consisting in one of the novel approaches presented here, by using the Bluetooth radio technology, the access to the voice dialogs can be done in a private and in a discreet manner.

Hence, the development of IVR applications embedded in general computing/electronic equipment, and communicating with the aid of the Bluetooth technology, surely represents a considerable advance in the way how people might interact with that equipment.

The developed work serves, as well, to reveal a communication concept that, besides the visually impaired citizens, can also improve the daily quality of life of the ones with mobility handicap. In fact, further to the exchanging of audio signals, the Bluetooth technology can be used to send remote commands, from respective personal Bluetooth devices, to machines or equipments with due compatible interfaces.

REFERENCES

[1] R. Dettmer, "It's good to talk", IEE Review, June 2003.

[2] L. Motiwalla, "Speech-Enabled Mobile Learning Application", Wireless Telecommunications Symposium, 2005.

[3] A. Talevski, E. Chang, "Reconfigurable Software Architecture for Voice Access to Data Services", IEEE International Conference on Digital Ecosystems and Technologies, 2007.

[4] C. Bisdikian, "An Overview of the Bluetooth Wireless Technology", IEEE Communications Magazine, December 2001.

[5] "BlueZ Home", On-line at: http://www.bluez.org/ (2006).

[6] "Bluetooth-ALSA Project Home", On-line at: http://bluetooth-alsa.sourceforge.net/index.html (2006).

[7] "Advanced Linux Sound Architecture – ALSA Home", On-line at: http://www.alsa-project.org/ (2006).

[8] "Btsco 0.42-r0 package", On-line at: http://www.angstrom-distribution.org/repo/?action=details&pnm=btsco (2006).