A Smart Electric Wheelchair Using UPnP

ETS Ingeniería Informática. Universidad de Sevilla.
Av. Reina Mercedes, s/n. 41012, Sevilla, Spain
`danic@atc.us.es`

Abstract. People with disabilities in general, and wheelchair users in particular, are one of the groups of people that may benefit more from Ambient Intelligent (AmI) Systems, enhancing their autonomy and quality of life. However, current wheelchairs are usually not equipped with devices capable of accessing services in AmI environments. In this paper, we describe how an electric wheelchair is equipped with an UPnP based module that allows the integration in AmI systems.

1 Introduction

Although the Ambient Intelligent (AmI) concept is not oriented towards any particular group of people, it is obvious that the AmI emphasis on greater user-friendliness, more efficient services support, user-empowerment, and support for human interactions [1] would be especially useful for people with disabilities and elderly people. In this paper, we focus on the integration of wheelchair users in AmI systems. Consider, for instance, the following scenario:

A wheelchair user with several mobility restrictions and a mobile computer in his/her wheelchair enters a building (let's say the rehabilitation centre), provided with Ambient Intelligent facilities. As soon as he/she gets into the building, the Ambient Intelligent System (AmIS) discovers his/her presence and the devices announce the services that can be used, according to his/her special needs (cognitive, sensorial, physical and communication abilities), and technological constraints (display resolution, voice, text, pixel-based, bandwidth, computing power, etc.).

The AmIS offers communication with a remote information centre that appears in the user's display, adapted to his/her physical and cognitive characteristics (text menu, voice, icons…). The AmIS offers information about where he/she is and where to go from the current position and, after knowing where to go, the possibility of a route-guiding tool appears on its display, which the user accepts. It uses a location service and gives information via text messages. The user is located and receives his/her position, together with a message of where to go now. The AmIS has calculated the best path to follow, taking into account the user constraints (for instance, avoiding stairs, changing the timing of automatic doors, etc.) and the information of occupation in the building at that time of day.

Finally he/she arrives at the destination and the AmIS sends information about what domotic devices are installed in the room (and can be used). From its joystick with buttons (or any other adapted input device), the user sends orders to control devices offered by the domotic system (i.e., switch the lights off or on, roll up or down blinds, change the television channel, set the temperature of the air conditioning system, etc.). Afterwards, he/she decides to leave the building, but this time the route-guiding tool is not employed, and the joystick is used to guide the wheelchair to the exit.

Note that some characteristics of Ambient Intelligent systems are particularly well suited for people with mobility restrictions:

- Ubiquitous access: allows access to services in a way that is not restricted by the location of resources and/or the user's mobility (remote control for TV, air conditioning, etc.; answering the phone from the wheelchair, etc.).
- Context awareness: apart from the obvious use of location awareness, other dimensions are useful in our case, particularly personal awareness (dynamic adaptation to user needs, abilities or preferences) [2].
- "Invisible" computing and networking: allows un-noticed user monitoring in terms of safety: falls, care for people who may get lost, etc. (e.g. elderly residences).

However, since many of these people usually require the use of mobility aids (like wheelchairs) and adapted user's interfaces, these assistive devices should also be integrated into the AmI system. For instance, it may be useful for the handicapped to access assistive services through their personalized interfaces (especially in unfamiliar environments). Furthermore, wheelchairs should be able to use and provide services (location, semiautomatic navigation, etc.). However, current wheelchairs are usually not equipped with devices capable of accessing services in AmI environments. At most, a wheelchair user may carry a portable computer or a PDA to access services, but in this case, the wheelchair itself is not integrated in the AmI system.

In this paper, we describe an electric wheelchair that is equipped with a module that allows the integration in AmI systems. We first describe a hardware module that serves as an interface between the wheelchair and the external devices. In the next sections, we explain the software architecture, which is based on UPnP[1] (Universal Plug and Play), and we focus on the User Interface software. Finally, we present the conclusions.

2 Hardware Architecture

The wheelchair used in this work is based on a former prototype called *Tetranauta* [3,4,5], a low cost, fully open steering system that allows people with severe motor impairments to move in known environments: hospitals, schools, home, etc. Navigation is assisted by allowing the wheelchair to follow predefined paths (with tracks

---

[1] http://www.upnp.org

marked on the floor) and also by using an infrared-based obstacle detection system. As a result, the user effort and safety in driving the wheelchair, especially in long paths, is improved.

So far, the majority of the efforts in wheelchairs have been oriented to empowering the autonomous capabilities, like obstacle avoidance, trajectory tracking, efficient suspension of wheels or stair-climbing capabilities. An example of this is OMNI [31] that included obstacle avoidance, human-machine interface, high maneuverability and navigational intelligence. NavChair [25] is a smart wheelchair that is able to avoid obstacles, and to follow a direction indicated by a user with tremor or another type of severe mobility impairment. IBOT-3000 [29] was capable of climbing stairs with its two pairs of balanceable wheels. The Smart wheelchair of the CALL Centre [26] allows several types of interaction modes with the user, according to his/her disability and skills. The wheelchair Tetranauta was in this line of developments, being an isolated wheelchair without communications with another devices. In [18], a revision of this type of wheelchairs can be found.

The next generation of wheelchairs contemplates smart capabilities for autonomous operation and communication capabilities with other systems (like wheelchairs, domotic devices or the like). In this sense, electric wheelchairs have been used like test beds for communication systems, like E-wheelchair [27] that was used to prove the viability of IPv6 communications. The design shown in [28] used its wireless communications to improve its autonomous capabilities (to communicate with a GPS to obtain the position of the wheelchair).

However, these last ones were an attempt of expanding the capabilities of wheelchairs through communications, but not to integrate the wheelchair in a bigger system. In this line, AmIChair [30] can use communications to control the devices of the environment, but environment devices can also monitor and control the wheelchair. The wheelchair is integrated in the whole system and it is only a part of it. Our development pursues a similar idea: being a device integrated in a bigger system, but now the user can interact with the system using its proper wheelchair's adapted interface.

The control unit of the wheelchair is composed of an embedded computer, plus several functional modules controlling different parts of the wheelchair: power module, steering device module, etc. All these modules are inter-connected by a DX serial bus [6], a *de facto* standard in electric wheelchairs. As a result, the wheelchair becomes a distributed embedded system where new functional modules may be connected with relatively few software changes. Particularly, in this paper we describe how a new gateway/bridge module is incorporated into the wheelchair so that the DX sub-systems can communicate with other devices in an AmI environment.

The system architecture is shown in Fig. 2. Essential DX modules are the User Control Module (UCM) and the Power Module (PM). The UCM is normally part of a control module (which includes a joystick or any other type of speed and direction control), and has the function of processing signals from the wheelchair user and sending instructions to the other modules. These instructions and other data are sent using messages that are named Network Variables (NV). The PM provides the controlled voltage to drive the wheelchair's motor(s) and operates the park brakes. These are modules already included in any DX-based wheelchair.
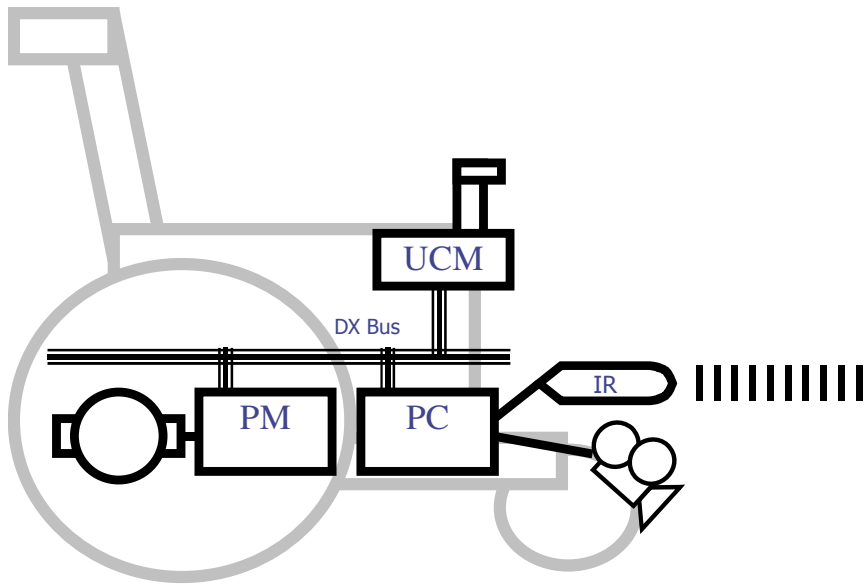
Fig. 1. Basic prototype's scheme

From our point of view, the central element is the user interface (UI), not included in standard DX wheelchairs. The UI should have the following characteristics:

- It should be a mobile system, easily handled by the user (who may not always be seated in the wheelchair) or any other person (relative, carer, nurse, etc.).
- It should have enough resources (computing power, memory, screen size, bandwidth) to run useful (visual) applications adapted to the user's needs and preferences. Furthermore, development kits should be available in order to write application-specific software.
- It should incorporate multiple communication links, preferably wireless, to allow ubiquitous access to other devices and services.

Advancements in handheld devices (such as PDAs, mobile phones and portable PCs) are enormous and there are now commercially available devices of acceptable performance at relatively low cost. In our prototype, we decided to use a StrongARM® based PDA as our user interface (UI), running under WindowsCE® 3.0. The UI is equipped with a color screen, as well as with serial, Bluetooth and 802.11 interfaces. Although usually a wireless connection is more adequate, sometimes a simple serial (wired) link is a simpler and more robust solution (for instance, when the PDA is attached to the wheelchair).

Another added element is the *DX-Bridge*, a new DX module that captures the Network Variables (NVs) flowing through the DX bus. This module allows NVs (for instance, wheelchair control data) to be exported as input for the user interface (UI), as well as to receive new values for the NVs from the UI. So far, the only allowed way to capture DX data from the bus is using a device named DX-KEY, provided by
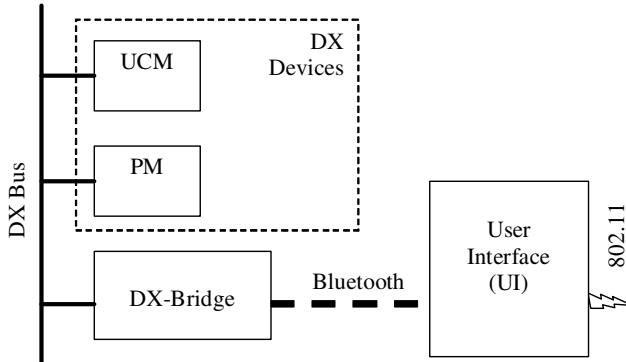
Fig. 2. Scheme of wheelchair's hardware

Dynamic Controls [7]. This is a DX module that provides an interface between the DX bus and an external system, allowing access through a parallel port to DX variables. Therefore, the DX-Bridge is a hardware module based on a Cygnal C8051F330 microcontroller [8] with serial and parallel interfaces: the DX-KEY is accessed through the parallel port, and on the other side a serial link (RS-232, 115200bps) is used to connect with the UI. As we said before, some times this wired link is enough. However, in order to provide a wireless link, we also use a commercial serial-Bluetooth module that works under the RFCOMM profile (Serial Cable Emulation).
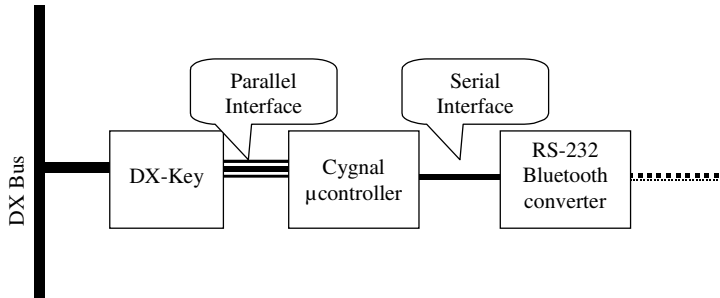

Fig. 3. Hardware in the DX-Bridge

The DX-Bridge has two operation modes: *driving,* and *domotic* modes. In the driving mode, the DX-Bridge simply captures the commands (NVs) sent by the User Control Module (UCM) and then it delivers these commands to the Power Module (PM). These commands are delivered without any changes, so the wheelchair operation is not different from a standard manually operated wheelchair. In the domotic mode, the DX-Bridge captures the commands sent by the UCM and then it delivers these commands to the user interface-UI. In this way, these commands are no longer used to control the wheelchair, but they are interpreted by the UI as commands to control a domotic system or the like. Obviously, the domotic mode cannot be used unless the wheelchair has reached a secure state, avoiding dangerous situations for the user.
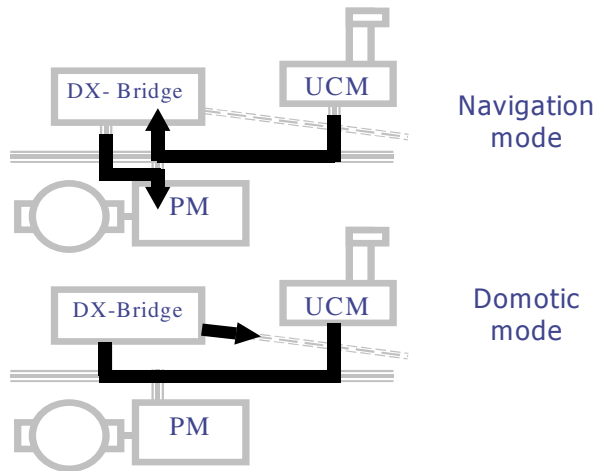
Fig. 4. Operation modes of DX-Bridge

The use of the wheelchair UCM to control a domotic system has two main advantages: first, the user is probably used to handling the control device (e.g. joystick) to drive his/her wheelchair, and therefore he/she would probably learn to use the external devices more easily. And second, since the UCM would probably be adapted to the user's needs and/or difficulties, we have an adapted control device for domotic systems "for free". This is a key question because with this solution, the cost of domotic systems does not depend on the user's physical and/or cognitive abilities.

Finally, although it cannot be considered as an operation mode, the DX-Bridge may be used by the UI to set new values for some DX variables. For instance, if an external service provides positioning and location information, these data can be sent to an optional assisted navigation DX module.
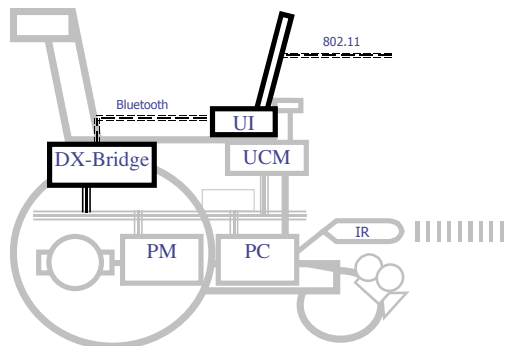

Fig. 5. Final architecture of the wheelchair

3 UPnP Architecture

As we said before, the wheelchair must be capable of obtaining information from the environment, for instance positioning information or the list of available devices and/or services, and the domotic system (or the like) must be capable of obtaining information from the wheelchair (like orders to the different devices of the room). Connection between these devices can be implemented in several ways: infrared cards, radio communication systems, or even wired connections. In our system, communications are centered on the user interface (UI), which should be able to communicate across heterogeneous and dynamically changing links and networks. For instance, the UI would be able to communicate with data networks, Internet Access Points, domotic buses like EHS (a Powerline-like bus used for the control of home devices) [9], etc. Wireless personal (e.g. Bluetooth) and local (e.g. Wi-Fi) area networks now permit low-cost commercial solutions for this type of communication, but there are still open problems like efficient roaming, reachability, intermittent failures, fault tolerance, security, etc.

Note that it is not only a problem of *interconnectivity* or interactions at the internetworking level, but also of interaction among devices at higher levels: control, configuration and information sharing in different formats, import/export services, etc. Among the different communication architectures available (Juni, UPnP, HAVi), as discussed in [10], we consider that UPnP is a good choice for implementing our communication system. UPnP is a lightweight set of protocols to extend the Plug & Play concepts to network devices, and it supports all mentioned functions including the dynamic connection of a device to a network, services offering and discovery, everything based on a unified description of functions and attributes of services through XML (*eXtended Mark-up Language*) documents [11]. UPnP is capable of working with scarce resources and unreliable connections (devices can suddenly appear and disappear), and a further reason to choose UPnP is that there are a large number of available SDKs for several platforms and operating systems [12,13]. Furthermore, there are two factors that make UPnP especially attractive from our point of view: one is the use of open and standard protocols; second is the use of the IP protocol at the lowest level.

Indeed, IP protocol has demonstrated its success in the interconnection of heterogeneous devices (a good example is the Internet). Most devices can be connected through a backbone IP network while secondary, maybe simpler, devices (e.g. sensors) may be connected using non-IP communications. In this case, a gateway is used to interconnect IP and non-IP sub-networks. For instance, in our prototype, we need gateways to interconnect the IP backbone network to the DX bus (a non-IP control network). Furthermore, UPnP operates with a set of existing and well-tested protocols and only needs an auto-IP network for running. UPnP works in a distributed philosophy, and classifies devices into two roles: *control points* or clients, and *host devices* or servers of services. However, almost all UPnP devices implement both client and sever functionalities, so peer to peer communications are possible.

For the wheelchair to be integrated into an AmI environment, the wheelchair must export information, acting as a host device: we need to know the position of the joystick, if any button of the console was pressed, the battery status, and so on. On the other hand, the wheelchair also needs to act as a control point (client): it has to know

its location in order to know how to get to another room/place, selecting the route from a map, etc. All these information/services are supplied from other devices (services) in the network. Since not only the wheelchair, but also the domotic system and most other elements of the AmI system get information from other devices, control points must be implemented on them.

The role of the user interface (UI) is to serve as an interface between the wheelchair and these UPnP services. All devices implementing UPnP services (host devices) should be connected by means of a backbone IP-based network. In our system, since the wheelchair is a mobile system, and services should be accessed "on the move", we use an IP wireless network (802.11b/g [14]).

Among the many different host devices that may be present in an AmI system, we identify the following for our scenario (see Fig. 6):

- Monitoring: exports wheelchair's status variables (i.e.: joystick position) and it is implemented in the wheelchair's gateway through serial (or Bluetooth) interface and the DX Bridge.
- Map storing: a small processor with an associated memory for storing and reading maps. This device can be in a fixed place of the building.
- Location: offers a positioning service, maybe out-doors positioning (like a GPS device attached to the wheelchair), or in-doors positioning [15].
- Domotic: offers as a service the kind of operations that may be performed with the associated domotic devices, tells which are the available devices at a specific room. An UPnP control point attached at the same domotic service's host device can read the status of the wheelchair to generate an alarm in the domo-tic system if the battery is low or can read the position of the joystick for using it as an input device to handle a graphical interface of any domotic device.

4   The User Interface (UI)

In this section, we describe the implementation of the UPnP host device in the user interface (UI). This description serves as an example, since all host devices can be implemented in a similar way. The UI has been implemented over a StrongARM®-based PDA equipped with serial, Bluetooth and 802.11b interfaces. The latter is used for supporting UPnP activity, as described in the previous section. We use an under-request protocol implemented *ad hoc*, that works as follows. The UI periodically sends a set of inquiry frames to the DX sub-system to know what the values of the DX variables are. When the micro-controller in the DX-Bridge (see section 2) receives an inquiry frame, it returns the value of the inquired variable. We preferred this simple solution because the Cygnal micro-controller used in the DX-Bridge is not powerful enough to support the UPnP stack. However, we could have implemented an UPnP DX variable service at an increased cost and complexity.

The UPnP software in the UI runs as an application over WindowsCE® 3.0. Basically, the application contains two protocol stacks: Serial communication stack and UPnP stack (see Fig. 7). The former implements the communications protocol between the DX-Bridge and the UI. This protocol is very simple; it only needs PHY and MAC layers, including some error correction capabilities like frame retransmission.

Over these two layers, a layer named *DX variable store* is implemented in order to guarantee the consistent storing of values (note that these DX variables are "resources" accessed both by the Serial Communication Stack and by the higher layer, so this layer has to cope with the consistent use of these shared resources). On top of both stacks is the UPnP service layer, which implements two threads: a DX inquiry loop (that gathers DX variables from the DX variable store layer) and the UPnP processing thread (responsible of gathering all the UPnP requests and generating the appropriate responses). DX inquiry loop notifies when a DX variable changes in order to notify this change to the UPnP clients connected to the service.
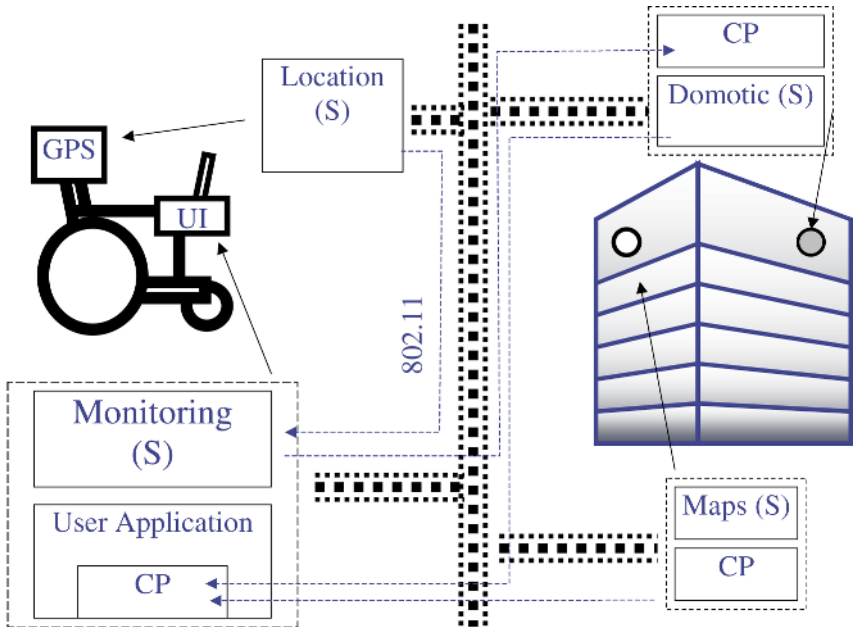

Fig. 6. UPnP services (S), Control Points (CP) for UPnP architecture. Dots indicate data flows.

The main task of UPnP stack is the implementation of the UPnP service. This is composed of a set of actions (methods) and status variables that the UPnP client (control point) can invoke at any time.

Service's status variables are all the DX variables that are desired to be monitored: joystick position; button, battery, DX control unit and serial communication status, and so on. The UPnP Monitoring Service maintains an inquiry loop responsible for holding the latest values of these variables. On the other hand, these services' status variables can be read by other UPnP devices in the IP network under demand (this is the default mode) or by a change notification event. Under this latter mode, the client receives an event every time the DX variable changes. When this event occurs, the client receives the variable name and value.
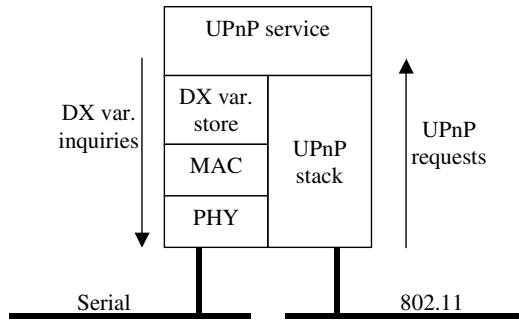
Fig. 7. UPnP application architecture in wheelchair's gateway device

Additionally, actions included in the service allow to read/write all the DX variables (not only the service's status variables) and even some internal DX variables that can be identified by its DX identification number (note that all DX variables have an identification number). However, these hidden internal DX variables cannot be read by events. Additionally, there's an action that resets the DX stack and the microcontroller (like a warm-reset).

Finally, the UPnP layer also implements a web interface, which allows all the status variables to be read and all the actions described above to be executed by means of a web page (see **Fig. 11**). As a result, the system is not only accessible as an UPnP device, but it could also be accessed as a simple web-controlled device. Since the wireless backbone IP network may be connected itself to the Internet with a *Residential Gateway*, remote control or maintenance via a web page is allowed.

5   Final Remarks

Some aspects have to be taken into account when using the wheelchair as described above. First, remember that commands sent by the UCM (i.e., the wheelchair's joystick) can be captured by the DX-Bridge and delivered to the user interface-UI in what we called the *domotic* mode. These commands are no longer used to control the wheelchair, but they are interpreted by the UI as commands to control an external system. Since in these cases the wheelchair acts as an UPnP host device (*monitoring service*), all these commands are transmitted through the UPnP protocol stack, including IP. Due to the high response times of these protocols, only very simple interfaces can employ the monitoring service as an input device. For instance, if we want to use the joystick movements to control an external domotic system, the corresponding changes of DX variables should be sent through UPnP. Tracking the joystick movements to control something similar to a mouse cursor would not be possible since it requires to send an event every time the DX variable changes.

However, DX-variables containing joystick positions can be used directly for controlling the user interface, since communications between the DX Bridge and the UI are performed through a serial/Bluetooth link and not through UPnP. For instance, in

our prototype, the joystick can be used as a mouse to control the cursor in the PDA. This allows us to centralize in the PDA the control of domotic devices using the UPnP domotic service. For instance, browsing a few buttons using the joystick and then sending the selected button is a less demanding solution for UPnP communications. Note that this implementation is not incompatible with the possibility of using the monitoring service in the system. The domotic devices could activate an alarm if the wheelchair's user presses an emergency button in the wheelchair's console, for example.In figure 8, the architecture of this modification is shown.
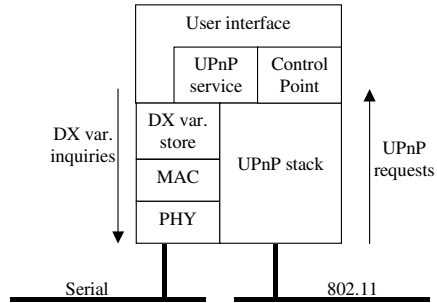

Fig. 8. Modified structure of the software in the PDA

Another aspect that has to be taken into account is that the use of Bluetooth to communicate the DX Bridge to the PDA may be problematic due to possible interferences. First, due to the coexistence between Bluetooh and 802.11b links in the same device, Bluetooth 1.2 links are recommended because they implement *adaptive frequency hopping* (AFH) techniques that allow coexistence with 802.11 links [16]. AFH works by using fewer than 79 channels in the frequency hopping mechanism if the Bluetooth device detects that there is interference on some of these frequencies. In this way, frequencies occupied by 802.11 are avoided, allowing co-existence.
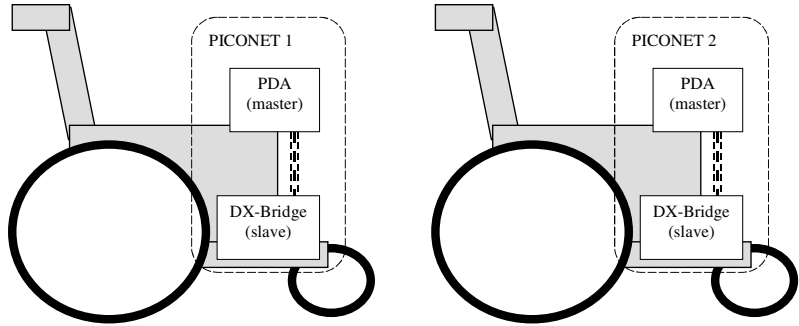

Fig. 9. Independent Bluetooth links formed in wheelchairs

In addition, the Bluetooth link may also suffer from other independent Bluetooth devices in the same area. The wheelchair may be placed in an area where other users have active Bluetooth links, for instance mobile phones, PDAs, MP3 players with headphones, or even other wheelchairs equipped with the same devices. In this case, every independent Bluetooth device uses its own frequency hopping sequence, so collisions may occur if two or more of these devices happen to choose the same frequency. If the DX variables sent through the Bluetooth link have time constraints (for instance, if they are used to activate an alarm or to control an external device) then some QoS guarantees are needed. For instance, in [24] the worst-case deadline failure probability of Bluetooth messages is obtained as a function of the number of independent interfering devices. Depending on the application, a number as low as 5-10 (which may not be strange in conference halls, airports, etc.) may be unacceptable.
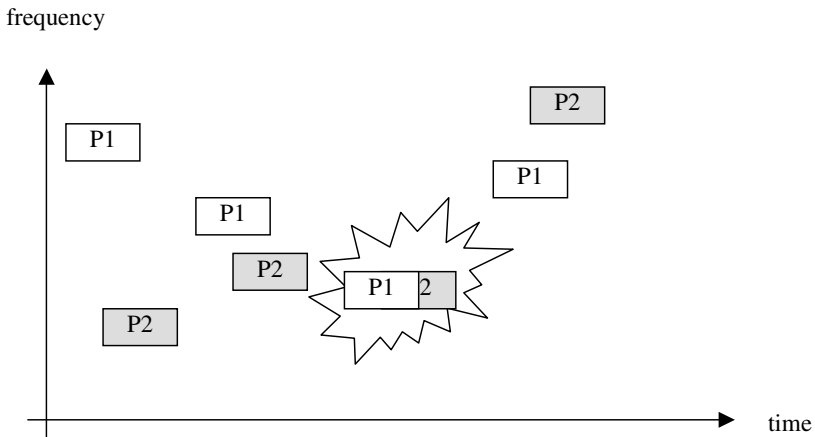

Fig. 10. Example of collision between piconets

6 Conclusions

In this paper, we describe how an electric wheelchair is adapted to allow its integration in AmI Systems. The wheelchair used is a prototype from a previous project, which is based on a DX Bus, a *de facto* standard in electric wheelchairs. We describe a new DX hardware module that allows the DX sub-systems to communicate with other devices, particularly with the user interface (in our prototype, a StrongARM®-based PDA equipped with serial and 802.11b interfaces). We also describe the software developed for this PDA, which is based on the UPnP architecture. The current prototype is able to interact with external devices in two ways: on the one hand, commands sent by the wheelchair UCM (joystick) are interpreted as commands to control a domotic system or the like. On the other hand, DX variables can be accessed from external devices, both as an UPnP device, as well as via a simple web page.

Fig. 11. Web page of the UPnP monitoring service

Acknowledgments

References

[1] ISTAG; Scenarios for Ambient Intelligence in 2010; Final Report, Feb 2001, EC 2001: http://www.cordis.lu/ist/istag.htm

[2] J.L. Sevillano et al.: On the Design of Ambient Intelligent Systems in the Context of Assistive Technologies. 9th International Conference on Computers Helping People with Special Needs, Paris 2004. LNCS 3118, pp. 914-921. Springer 2004.

[3] A. Civit, J. Abascal: "Tetranauta: A Wheelchair Controller for Users with Very Severe Mobility Restrictions". Improving the Quality of Life for the European Citizen. I. Plasencia, E. Ballabio (eds.). pp. 336-341. IOS Press, 1998.

[4] S. Vicente, et al.: "TetraNauta: a intelligent wheelchair for users with very severe mobility restrictions". Proc. IEEE Int. Conf. on Control Applications, Pp: 778 - 783. Sept. 2002.

[5] S. Vicente Díaz. "Una aportación al guiado de sillas de ruedas eléctricas en entornos estructurados" (in spanish). PhD Thesis. Universidad de Sevilla, July 2001.

[6] http://www.dynamic-controls.co.nz

[7] Mike Meade, "DX Key Technical Description. For DX Key Application Designers." Dynamic Controls Ltd., 1997.

[8] http://www.silabs.com

[9] http://www.ehsa.com/

[10] J. Abascal, J.L. Sevillano, A. Civit, G. Jiménez, J. Falcó: Integration of heterogeneous networks to support the application of Ambient Intelligence in assistive environments. IFIP Conf. on Home Oriented Informatics & Telematics HOIT 2005 (York, U.K.. April 2005).

[11] Jeronimo M, West J: UPnP Design by Example: A software developer's guide to Universal Plug and Play. Intel Press (2003).

[12] http://www.intel.com/labs/connectivity/upnp/index.htm

[13] http://www.plug-n-play-technologies.com/

[14] 802.11 Working Group's Site: http://grouper.ieee.org/groups/802.11/

[15] R. Casas, "Sistema interoperable de localización en interiores aplicado a tecnología asistencial" (in spanish). PhD Thesis. Universidad de Zaragoza, Spain. Sept. 2004.

[16] The official Bluetooth website. http://www.bluetooth.com

[17] Axel Lankenau, Thomas Röfer: "A versatile and safe mobility assistant", IEEE Robotics and automation magazine, p29-37, March 2001.

[18] Dan Ding, R. A. Cooper: "Electric-powered wheelchairs: a review of current technology and insight into future directions". IEEE Control Systems Magazine, p22-34, April 2005.

[19] http://www.atc.us.es/?op=investigacion

[20] J.C. Haartsen, The Bluetooth Radio System,. IEEE Personal Communications 7(2000) 28-36.

[21] The Bluetooth Special Interest Group: Specification Of Bluetooth System - Core Vol.1 V1.1. Feb 2001. www.bluetooth.com.

[22] D. Cascado, J.L. Sevillano, S. Vicente, F. Díaz del Río, G. Jiménez, A. Linares, A. Civit-Balcells. Modeling Effects of Co-channel Interference over Performance in Single-Slave Bluetooth Piconets. The 15th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2004).

[23] D. Cascado. Study and evaluation of a wireless communication system for personal area networks (in spanish). Ph. D. Tesis. University of Seville (2003).

[24] J.L. Sevillano, D. Cascado, F. Díaz del Río, S. Vicente, G. Jiménez, A. Civit-Balcells. Statistical QoS guarantees in Bluetooth under co-channel interference. 10th IFIP International Conference on Personal Wireless Communications (PWC 2005).

[25] R.C. Simpson and S.P. Levine. Automatic adaptation in the NavChair assistive wheel-chair navigation system. IEEE Trans. Rehab. Eng., vol. 7, no. 4, pp. 452–463, 1999.

[26] Nisbet, P.D. (2002) Assessment and Training of Children for Powered Mobility in the UK. Technology & Disability 14 (2002). p173–182. IOS Press. ISSN 1055-4181/02.

[27] Thierry Ernst. E-Wheelchair: A Communication System Based on IPv6 and NEMO. 2nd International Conference On Smart homes and health Telematic (ICOST2004).

[28] Chuan-Heng Hsiao et al. A design of small-area automatic wheelchair. IEEE International Conference on Networking, Sensing & Control, p1341-1345. (Taiwan 2004)

[29] R.A. Cooper, M.L. Boninger, R. Cooper, and A.R. Dobson.Technical perspectives: Use of the Independence 3000 iBOT Transporter at home and in the community. J. Spinal Cord Med., vol. 26, no. 1, pp. 79–85, 2003.

[30] Salvador, Z., Bonail, B., Lafuente, A., Larrea, M., Abascal, J. and Gardeazabal, L., AmIChair: Ambient Intelligence and Intelligent Wheelchairs. Proceedings of HOIT 2005 (Home Oriented Informatics and Telematics 2005).

[31] H. Hoyer. The OMNI wheelchair. Service Robot: An International Journal, Vol.1 No.1, MCB University Press Limited, Bradford, England, pp. 26-29, 1995.