XVII IMEKO World Congress
Metrology in the 3rd Millennium
June 22–27, 2003, Dubrovnik, Croatia

ENHANCEMENT OF A BLUETOOTH-BASED INSTRUMENT WIRELESS INTERFACE

L. Ferrigno[1], *A. Pietrosanto*[2], *A. Celano*[2]

[1]DAEIMI, University of Cassino, Cassino Italy
[2]DIIIE, University of Salerno, Fisciano Italy

Abstract – The paper deals with the enhancement of a Bluetooth-based interface for instrumentation, which has the aim of satisfying the need of wireless connection today arising in developing automatic measurement systems. The interface improvements are widely described as for both hardware and software components. Finally a complete experimental test set allows conclusions to be drawn about the interface performance.

Keywords: instrumentation bus, wireless, Bluetooth.

1. INTRODUCTION

The traditional and consolidated idea (scheme) of automatic measurement system takes its origin from the applications which introduced the need of automation in measurements. In the seventies the first automatic test equipments were developed in electronic industry to reduce costs and to improve reliability of circuit testing. Hereinafter automatic measurement systems have been designed as a set of electronic instruments connected to a computer via an interface bus, and often constrained in rack and stack structures allowing cable connection to be easily made. Even though the field of application of automatic measurement systems has been greatly widespread during the successive decades, no significant innovations to the first idea were proposed. Some proposals were aimed to reduce dimensions of instruments and to increase data transfer rate (IEEE 1155), others to satisfy industrial (field-bus) or automotive (CAN bus) needs, but architectures are always centred on the adopted standard interface bus. Moreover, in every case the wirings among instruments impose a closeness which depends on the single interface but cannot be avoided. Recently LAN or WAN solutions have been proposed to extend the area that can be covered by a measurement station even over geographic distances. These solutions, however, uses communication ways which are characterized, either by not deterministic data transfer times (Internet), or by high cost of wireless device (WAN).
This means that any of these solutions has been designed to satisfy the need of short range (10 - 100 m), low cost, wireless connections among general purpose measurement instruments. These consideration led the authors to design a BT communication channel based wireless instrumentation interface, where both master and slave interface boards are thought to be simply installed on the serial port of a PC and of instruments respectively (Fig.1). This means that every stand alone instrument equipped with a RS232 interface could join other devices in the piconet and be controlled by the PC, without any firmware modification. This solution looks like economic, easy to install and general because its hardware and software parts do not depend in any way on instruments.

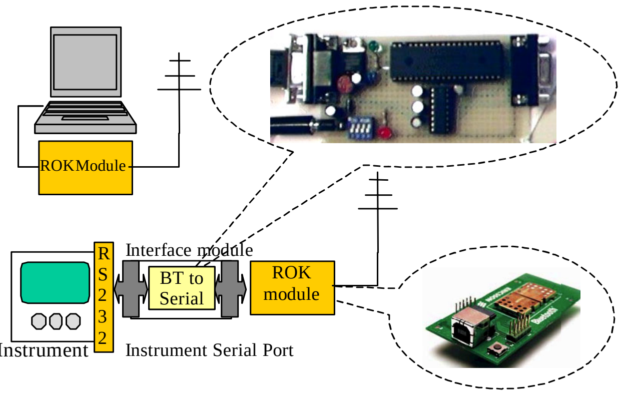

Fig.1. The previous serial to serial BT wireless instrumentation interface

This first interface module was realized for measurement instruments equipped with a RS232 serial port. This allows the wireless connection in the BT piconet among RS232 instruments and a controller. The controller module is made of a commercial BT dongle for serial or PCI PC port, while the instrument module has been ad hoc designed and realized. It is based on a microcontroller which implements the RFCOMM and the HCI management, and creates a double layer RS232 interface, the former toward a slave BT module, the latter toward the measurement instrument (or any other device) RS232 port.
In this way it is possible to use, easy and with low cost, the BT technologies to create a wireless interface bus between measurement instrument. Both master and slave BT modules are connected to the RS232 ports of PC and peripherals, respectively. The PC hosting the master BT module works as Controller, while instruments back-fitted with the slave BT modules are the peripherals. Device§ commands and data can be written to and read from each peripheral by the controller.

Even if perfectly working with the same data transfer rate of the wired serial connection, the wireless interface needs both improvements and additional features or options: (i) software trigger commands cannot be implemented yet; (ii) not more than 300 bytes can be exchanged in a unique solution with an instrument without causing memory and stack problems to the slave interface module; iii) only seven slaves can be active simultaneously; iv) a IEEE 488 version of the slave interface module misses.

Some of the aforementioned limits of both master and slave modules have been overcome in a new slave BT interface module, which has been designed to be connected to the IEEE-488 GPIB. Broadcast messages have been implemented in both master and slave software to add a software trigger option; the scatternet is allowed to be realized, thus increasing the maximum number of devices simultaneously active; the micro-controlled interface has been featured with a 8Mbyte RAM to enlarge memory stacks. In the paper the new IEEE-488 slave module is presented and both hardware and software are described in detail. Then, a test phase is carried out with the aim of characterizing the wireless interface in a configuration including both serial-RS232 and IEEE-488 instrumentation.

2. THE SLAVE BT-IEEE488 INTERFACE MODULE

The hardware

The proposed BT- IEEE 488 interface module has the same hardware organization of the BT-RS232 interface module: a microcontroller based board has been designed to be connected from one side to the IEEE 488 GPIB, and from the other side with the RS232 serial port of a BT slave module.

The Microchip PIC16f877 is the heart of the microcontroller-based board. It is a low cost microcontroller whose main characteristics are: a 20 MHz operating frequency, 8 kbyte program ram, 33 I/O pins, two RS232 serial ports (one USART RS232 port, and one synchronous port realized by using the Serial Peripheral Interface (SPI) bus). An external 20 Mhz oscillator allows the microcontroller to be used at its highest processing rate. In order to allow the microcontroller based board to be connected with the RS232 port of the slave BT module, an interface driver (Maxim MAX232) converts the TTL voltage level provided by the PIC microcontroller USART to those required form the EIA RS232 standard. On the other side, a compatible IEEE 488 interface port has been realized. The transceivers for the 16 command and data line of the IEEE 488 bus, have been realized using 16 digital I/O pins of the microcontroller. In particular 8 pins have been used to implement the IEEE 488 DIO0-7, 3 pins for the handshake (lines NRFD, NDAC e DAV), 4 pins have been used for the ATN, EOI, REN and IFC lines, and finally the last pin has been reserved for the SRQ line. This last pin allows IEEE488 devices to get the controller attention. In this prototypal realization (Fig.2) the interface board looks big, but the aim of the Authors, is to contain the final version in a IEEE 488 connector package.

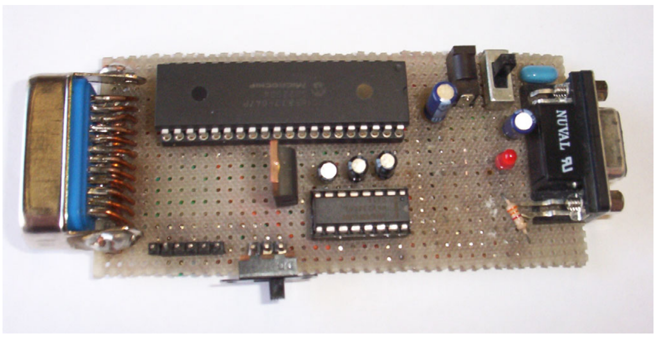
Fig.2 The microcontroller –based board of the BT-IEEE 488 module.

The software

The software kernel of the slave BT IEEE-488 interface module has been designed to allow the module itself to communicate with both a single IEEE 488 instrument, and a group of IEEE 488 cable connected instruments. Its architecture has been developed hypotizing that the BT master be able to send broadcast commands to all the slave BT modules present in the piconet. Consequently each slave module must be able to recognize its address when a command is sent by the master. Whether addressed it has to operate consequently, else, no actions have to be taken on receipt. In this framework, the slave module software was organized as it follows:

i. At the startup, the microcontroller module fills an internal table with information got by the IEEE 488 instruments connected with the module. At first the microcontroller try to make a logical address list of the connected IEEE 488 instruments. To do this, the interface module sends sequentially a Selected Device Clear to all the IEEE 488 possible logical addresses. Only one device at time answer this command thus allowing the interface module to collect all the logical addresses of the GPIB active instruments. Then, the microcontroller module sends an IEEE-488.2 identification query, in order to memorize instrument type and manufacturer of 488.2 instruments.
ii. If a BT packet, transmitted by the master, is received by the slave BT module, the microcontroller software polls the two bytes which correspond to the PSM field of the received L2CAP packet, as it will be described in the next section. If the first PSM field is a \x1001 (packet directed only to IEEE 488 instruments) or if it is a \x10EF (general purpose packet directed both to RS232 and IEEE 488 instruments) the microcontroller elaborates the received command, otherwise the packet is neglected by the module.
iii. If the recievd L2CAP packet is a IEEE488 command, the microcontroller software looks for the received device address in its address list: if the received address misses no actions are taken, otherwise the following section starts.
iv. In order to interpret the received command the microcontroller software investigates the first byte of the payload of the received L2CAP packet. In this byte

seven IEEE 488 command have been codified. These command are described in Tab.1. Once the right IEEE 488 command has been identified the interface module translate it in terms of handshake and/or data line driving. Particular care has been had for the transfer of large amount of data on the wireless channel. Suppose that a set of waveform points has to be acquired from a scope, say 1 million points. If the scope uses two bytes for each waveform point, 2 millions bytes have to be transmitted on the wireless channel. The payload that BT assigns to the user is composed by not more then 128 bytes; for this reason several BT packets are necessary to complete the data transmission. The problem is that both the slave interface module and the master software must understand that all these packets compose the same data transmission. To this aim, a suitable software procedure has been adopted. When a module starts a long data transmission, it sets true the most significant bit of the second byte composing the PSM register. This bit assumes the significance of "wait for another packet", and it is reset to false on the last BT packet. By this way, any module is able to reconstruct the whole data set.

Tab.1. The IEEE 488 functions implemented on the slave interface modules

Command Name	Corresponding Action
BT_SIC	Interface Clear
BT_SRE0	Disable the remote status (Set the REN line to false)
BT_SRE1	Enable the remote status (Set the REN line to true)
BT_CMD	Used from the controller to send an IEEE 488 command to a device (i.e. to address a device as talker or listener)
BT_WRT	Allows controller writing data to a device
BT_RD	Allows controller reading data from a device
BT_GTS	Set ATN to false and allows communication between a talker and listeners

3. THE MASTER SOFTWARE

Before describing the master software architecture, some choices made to allow both RS232 and IEEE 488 instruments to be included in the same BT piconet have to be underlined. The master only manages broadcast packets, while the slaves send unicast packets toward the master. This choice in the communication procedure was imposed by the IEEE 488 structure, which allows the possibility that the controller sends universal commands to all the devices on the bus, while talkers can send data only toward addressed devices. For this reason each BT packet is allowed to reach always all the slave modules of the piconet; as a consequence each slave interface module has to understand whether the received packet was really directed to it or not. As above mentioned the decision is made considering the PSM field in the L2CAP sub-packet composing baseband BT packets. The PSM field consists of two bytes register that represents a BT multiplexer key toward the higher protocol levels of the BT stack. The values of the PSM register that allows reaching the higher BT stack levels have been decided by the Special Interest Group (SIG), when BT has been defined. For example to activate a RFCOMM connection the value \x0003 has to be chosen. The SIG allows all the odd PSM values greater than \x1000 to be used for user defined applications. The choice made in this project is to use the PSM hexadecimal code 1001 to select an IEEE 488 packet, the 1003 code to define an RS232 packet and the hexadecimal 1005 code to select an USB packet. Finally, the \x10EF code is used to define a general packet that has to be directed to every instrument in the piconet. Using this bus organization will be possible, in the future, to implement also new communication levels, for an example to add IEEE 1394 instruments to the wireless bus. The realized master software module is composed by the following parts:(a) the RS232 and IEEE 488 drivers, (b) the virtual communication port management, (c) the physical communication port management, (d) the BlueDaemon daemon thread.

(a) The base principles of the realized RS232 driver have been described in a previous work. As far as the IEEE 488 driver is concerned, the authors choose to define the same function prototypes the National Instrument proposes for its software. By this way a user that have already realized an IEEE 488 software using the National Instrument functions has only to rebuild its project excluding the National Instrument libraries and including the realized BT software libraries. After that, the user doesn't see the BT bus, and his realized measurement software will work as an IEEE 488 application. It is worth noting that linking both BT-RS232 and BT-IEEE 488 libraries the user can create a unique software that controls both serial and parallel instruments.

(b) Virtual communication ports substitute the physical IEEE 488 or RS232 ports on the PC controller. In truth, these ports exist on the operative system only as a set of user defined registers where commands or messages sent to the physicals IEEE 488 or RS232 ports are written.

(c) The physical communication port is a real RS232 or USB port present on the PC controller, where is connected the master BT module.

(d) BlueDaemon is the software kernel resident on the PC controller. The kernel of the realized master software module is a daemon thread able: (i) to capture the messages that RS232 and IEEE 488 based software exchange with the serial and IEEE 488 busses, (ii) to create the BT packet and (iii) to redirect these messages to the BT channel. These operations are made without requiring any action to the user that, after a configuration phase, sees on its PC the RS232 serial and IEEE 488 interface ports, as they were really available on the PC. The realized daemon is able to manage all the BT stack levels, and for these reason it can be used also for general purpose software. In particular, the realized kernel is able to send any BT command and manage the events coming from the queried module.

4. CASE STUDIES ABOUT THE PROPOSED WIRELESS INSTRUMENTATION BUS.

In order to better describe the functionality of the proposed interface, some possible wireless configurations have been described and analyzed. In the first one, where the measurement station is composed of a controller and some IEEE 488 instruments, each instrument is connected with its own BT slave module interface.

In this case, some situations were explored: if the bus is in command mode (ATN = true) the master sends a broadcast packet, which reaches all the BT slave modules present in the piconet area. Now each slave analyzes the PSM register in the L2CAP sub packet and, if it finds the hexadecimal value 1001, recognizes the presence of a IEEE 488 command. Starting from this moment each BT slave interface module decodes the received command and, if it represents an addressed command, it searches the presence of the logical IEEE 488 address of the connected instrument. It is worth reminding that each BT slave interface module knows the IEEE 488 logical address of the connected instruments thanks to its start up routine. Successively, if the controller changes to the data mode (ATN = false), only the addressed interface modules analyze the received data packet. If an instrument has to send data to the controller the BT interface module, which has been addressed as talker before, write its data in the first reserved odd time slot assigned by the master module.

A last consideration has to be made about the IBGTS function that triggers the communications among a talker and listeners. Since different IEEE 488 instruments are connected to different interface modules, only the slave BT module connected with the talker instrument sends its data to the master, when the IBGTS function is received. The master module, in its turn, knows that the previous command it has sent was the IBGTS command, and transmits the received packet using a broadcast packet. Consequently, the interface modules that recognize an own listen addressed instruments will retain and process the packet. The previous action can be made because each interface module reserves on its RAM the three flag bits shows on Tab. II. In particular, by using both the OWN_TALK and the OTH_LIST flags, the interface module knows that it has to send data to the master that addresses the active listeners.

Tab. II Description of the flow control flag bits used

	Flag Name	Description
1	OWN_LIST	There is almost a listen addressed instrument on the IEEE 488 bus subsection connected to the interface module
2	OWN_TALK	There is almost a listen addressed instrument on the IEEE 488 bus subsection connected to the interface module
3	OTH_LIST	There is almost a listen addressed instrument on another IEEE 488 bus subsection controlled by a different slave module

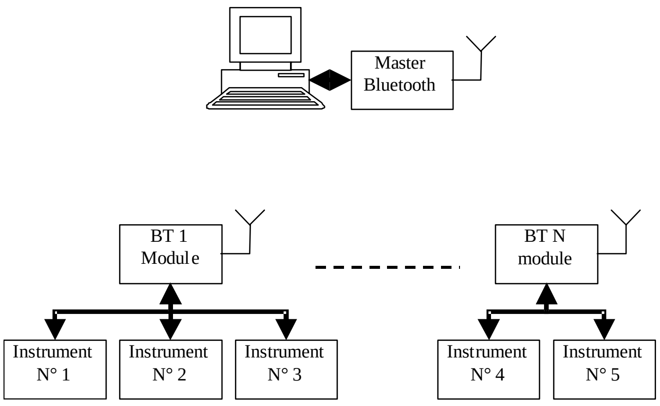

Fig. 3. The connection realized in the second case study

The second case study concerns a measurement station formed by a controller and several IEEE 488 instruments wire connected to the same BT slave interface module

As far as IEEE 488 commands sent by the controller are concerned, this second case is equal to the first one. On the contrary, substantial differences can be found when a data has to be transmitted from a talker to some listeners. In this case, it could be possible either that the talker and listeners be wire connected to the same BT slave module or not.

Considering the schematics reported in Fig.4, supposing that the instrument N°1 has been addressed as talker and the instruments N°2 and 4 as listeners. The BT1 slave module knows that a talker and a listener are connected to it and that there is almost another listener on a different slave module thanks to the previous mentioned flags. In particular, by using the address table it built in the start up phase, it knows also the address of the IEEE 488 instrument connected to it. The data transmission goes on in the following way:

(i) When the ATN line goes to zero, the talker instrument puts its data on the GPIB bus;

(ii) The IEEE instrument N°2 takes these data because it is wire connected to the talker an therefore the handshake procedure can happen;

(iii) The slave module reset the OWN_LIST and OWN_TALK flags and sends the data to the master.

(iv) The master resends the data packet using a broadcast communication.

(vi) The arrived packet is processed by the BT2 module, which has the OWN_LIST at the true value, and is not considered from the BT1 interface module that has the same flag at the false value; (vii) When ATN becomes true again, every BT slave interface resets the three considered flags.

In the third case study, the measurement station is formed by a controller, several RS232 instruments, and several IEEE 488 instruments connected to separate BT slave interface modules. This scenario is the great novelty of the realized wireless instrumentation bus: the possibility to create a communication link between IEEE488 and RS232 instruments. Differences respect to the others cases can be reassumed as follows:

(i) The master software module creates a virtual com port for each serial instrument (BT to serial interface board) present on the piconet area. The master software module uses also the \x1003 and \x10EFcode in the PSM register to indicate the presence of serial or general purpose packets

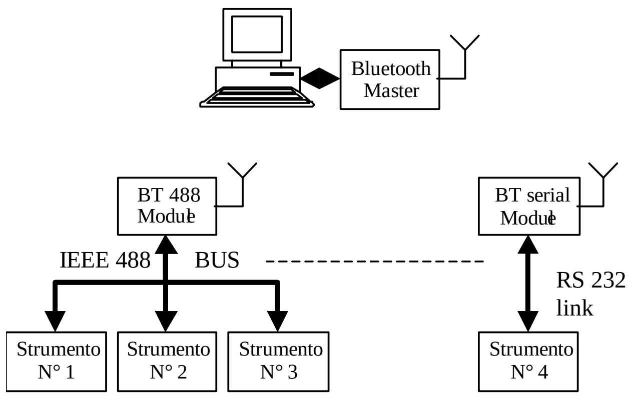

Fig. 4 The connection realized in the third case study

The master puts at the beginning of the L2CAP packet the address of the serial BT module to connect.
(ii) If the PSM code is \x1003, the slave interface module reads in BT address value at the beginning of the L2CAP. If its own BT address is recognized the data packet is passed to the instrument.
(v) If a serial instrument has to communicate with the master, the respective BT slave interface sends a unicast packet to the master, which locates the BT address and redirects the packet to the correct virtual COM port.

PERFORMANCE ANALYSIS AND CONCLUSIONS

In this section, the performance of the realized wireless instrumentation bus is explored.
In particular, for both the BT RS232 serial and BT IEEE 488 interface some features have been analyzed. The results are reported in Tab. III where the performance has been compared with that obtainable by using the respective wired busses.
As it is possible to see, the transfer rate is quite low if compared with theoretical value of IEEE-488, but it is competitive with wired serial busses. The bottle neck is constituted by the instrument RS 232 port which in the best case could be able to transmit and receive at 57600 baud. Better performance will be achievable when the USB ports of PC and instruments will be used instead of the RS232 ports.

REFERENCES

[1] L. Ferrigno, A. Pietrosanto "A bluetooth-based proposal of instrument wireless interface" IEEE International Symposium Virtual and Intelligent Measurement Systems, 2002. pp 72-77
[2] P. Arpaia, A. Baccigalupi, A. Pietrosanto: "Performance Optimization of VXI-Based Measurement Stations", IEEE Trans. On Instrumentation and Measurement, Vol. 44, num. 3, pp 828-831, June 1995.
[3] L. Angrisani, A. Pietrosanto: "Performance Comparison of IEEE-488 and 1155 Based Measurement Stations", Atti del VII IMEKO TC-4, pagg.634-638, Praga (Czeh Rep.), Sept. 1995.
[4] S.V. Wunnava, P. Hoo: "Remote instrumentation access and control (RIAC) through internetworking", Proc. of IEEE Southeastcon '99, pagg.116-121, March 1999.
[5] M. Dunbar, "Plug-and-play sensors in wireless networks ", IEEE Instrumentation & Measurement Magazine pp 19 – 23, Volume: 4 Issue 1, March 2001.
[6] U. Bilstrup, P. A. Wiberg: "Bluetooth in industrial environment", Proc. of 2000 IEEE Intern. Workshop On Factory Communication Systems, pagg.239-246, Sept 2000.
[7] J.C. Haartsen: "The Bluetooth radio system", IEEE Personal Communications, Vol.7, Issue 1, pagg. 28-36, Feb 2000.
[8] D. Famolari, P. Agrawal, "Architecture and performance of an embedded IP Bluetooth personal area network", 2000 IEEE International Conf. on Personal Wireless Communications, pp75 – 79, NJ, USA, Dec. 2000.

Authors:

Luigi Ferrigno, DAEIMI, University of Cassino, Via G. di Biasio, 43, 03043 Cassino (FR), Italy, Phone: (39) 0776-299673, Fax: (39) 0776-299707, Email: ferrigno@unicas.it

Antonio Pietrosanto DIIIE, University of Salerno, Via Ponte don Melillo, 84084 Fisciano (SA), Italy, Phone: (39) 089-964248, Fax: (39) 089-964218 Email: apietrosanto@unisa.it

Tab. III Comparison between the proposed BT instrumentation bus and the major wired competitors

Considered Parameters	BT to serial interface	RS232 interface	BT to IEEE 488 interface	IEEE 488 interface
Type of connection	Wireless	Wired	Wireless	Wired
Maximum number of connected instruments	1 for each interface module Up to 255 interfaces modules contemporaneously between active and parked	1	20 for each interface module Up to 255 interfaces modules contemporaneously between active and parker	20
Maximum distance from the Controller/Master	100 m (10mW BT modules) 10 m (1mW BT modules)	20m	100 m (10mW BT modules) 10 m (1mW BT modules)	2
Maximum transfer rate toward the instrument	57600kbs	115200 kbs	57600kbs	1 Mbs to 100 Mbs
Maximum transfer rate toward the master	1 Mbs	N/A	1 Mbs	N/A
Bus Structure	STAR	None bus is allowed	PARALLEL (broadcast messages)	PARALLEL
Cost	50 USD	5USD	50USD	200 USD