A Bluetooth-Based Proposal of Instrument Wireless Interface

Luigi Ferrigno, Vincenzo Paciello, and Antonio Pietrosanto, *Member, IEEE*

***Abstract*—This paper deals with a Bluetooth-based interface for instrumentation, whose aim is that of satisfying today's arising need of wireless connection in developing automatic measurement systems. After some notes about the Bluetooth communication protocol, the interface architecture of both hardware and software parts is described. Finally, some experimental results allow early conclusions to be drawn about the interface performance.**

***Index Terms*—Bluetooth, instrumentation bus, interface, measurements automatic stations, wireless.**

I. Introduction

EVEN though the IEEE-1155 instruments (better known as VXI instruments) were presented in the early nineties as the new instrumentation frontier, they have not yet substituted stand-alone IEEE-488 or RS-232 controlled instruments [1]–[3]. The undoubted improvements in weight and dimension, data transfer rate, triggering capability, and electromagnetic compatiblity (EMC) specifications are not sufficiently considered in a lot of industrial applications to justify the higher cost that they require. In some other cases, the manual use allowed by stand-alone instruments still seems to be indispensable. As a consequence, they are still widespread in the world of automatic test equipment for both industrial and scientific applications [4]–[6]. Today, however, there is more interest, both in commercial and scientific environments, in the numerous opportunities given by wireless connection among electronic devices [7]–[9]. Instruments are usually obliged to proximity by interface cables when a measurement station has to be made up. Cables themselves constitute an obstacle which, in both industrial and laboratory environments, reduces safety and free space. Moreover, a change in the physical configuration of these measurement stations cannot be done without service interruptions and software updating. With the aim of overcoming these and other similar problems concerning cable-based interfaces, a lot of wireless systems have been proposed in technical and scientific sites. Most of them concern the use of "Bluetooth" modules for realizing short-range wireless communication among electronic devices [10], [11].

Bluetooth is a short-range radio link conceived of replacing cable connections among portable and/or fixed electronic devices. The key features are robustness, low complexity, low power, and low cost. Bluetooth operates in the unlicensed ISM band at 2.4 GHz. The symbol rate is 1 Ms/s. A slotted channel is applied with a nominal slot length of 625 μs. For full duplex transmission, a time-division duplex (TDD) scheme is used. On the channel, information is exchanged through packets. A packet nominally covers a single slot, but can be extended up to five slots. The Bluetooth protocol uses a combination of circuit and packet switching. Slots can be reserved for synchronous packets. Bluetooth can support an asynchronous data channel, up to three simultaneous synchronous voice channels, or a channel which simultaneously supports asynchronous data and synchronous voice. The asynchronous channel can support a maximum of 723.2 kb/s asymmetric (and still up to 57.6 kb/s in the return direction), or 433.9 kb/s symmetric.

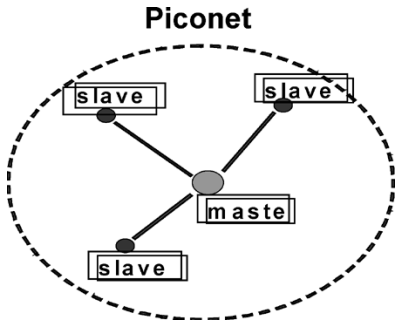

Fig. 1. Example of Bluetooth piconet.

Bluetooth (BT) modules are able to provide a point-to-point connection (only two BT modules involved), or a point-to-multipoint connection. In the point-to-multipoint connection, several BT units share the physical channel. Two or more units sharing the same channel form a ***piconet*** (Fig. 1). In a piconet, there is only one BT master module, while the other modules act as slaves, and up to seven slaves can be active simultaneously.

These characteristics have led the authors to design a BT communication channel-based wireless instrumentation interface, where both master and slave interface modules are conceived of being simply fit respectively in the serial port of a PC and instruments. This means that every stand-alone instrument whose RS232 interface port is fit with the proposed slave BT module could join up to seven other devices in the piconet and can exchange data and device commands with a PC hosting a standard BT master module, without any hardware or firmware modification. This solution seems economical, easy to install, and general because its hardware and software parts do not depend on instruments in any way.

Manuscript received January 29, 2003; revised June 9, 2004.

L. Ferrigno is with the Dipartimento di Automazione Elettromagnetismo Ingegneria dell'Informazione Matematica Industriale (DAEIMI), University of Cassino, 03043 Cassino, Italy (e-mail: ferrigno@unicas.it).

V. Paciello and A. Pietrosanto are with the Dipartimento di Ingegneria dell' Informazione e Ingegneria Elettrica (DIIIE), University of Salerno, 84084 Fisciano, Italy (e-mail: apietrosanto@unisa.it).

Digital Object Identifier 10.1109/TIM.2004.840245

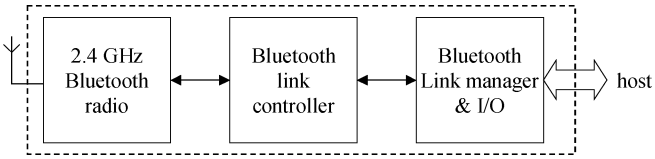

Fig. 2. Architecture of a Bluetooth module.

II. Bluetooth Fundamentals

A BT module (Fig. 2) is composed of: a radio unit; link control and link management units (link controller); and a host terminal interface unit [12], [13]. The link controller can operate in two major states **Standby** and **Connection**. The **Standby** state is the default low-power state of the BT module. Only the native clock is running and there is no interaction with any module whatsoever. In the **Connection** state, master and slaves form a ***piconet*** and can exchange data and audio packets using a frequency-hopping scheme. Between these major states there are seven substates, which are used to add slaves or make new connections in the piconet: *page; page scan; inquiry; inquiry scan; master response; slave response; and inquiry response*.

The **inquiry procedure** enables a module to discover which modules are in range, and determines the addresses and clocks for the modules. Once the inquiry procedure has been completed and the other module addresses are asigned, connections can be established using the **paging procedure**. The unit that performs both these procedures will be the **master** of the connection [14]–[16].

A BT module in the **Connection** state can be found in any of the four following modes: **Active; Hold; Sniff;** and **Park** mode.

In the **Active mode**, the unit actively participates in the physical channel (piconet); the master schedules the transmission on the basis of traffic demands to and from the different slaves. In **Hold**, **Park**, and **Sniff** mode, the modules are synchronized to the piconet in different power-saving modes. Up to 255 slaves can remain locked to the master using the parked state. Multiple piconets may cover the same area. If multiple piconets cover the same area, a BT module can participate in two or more overlaying piconets by applying time multiplexing. A BT module can act as a slave in several piconets, but as a master in a single piconet.

BT is organized as an International Standards Organization-Open Systems Interconnect (ISO-OSI) stack (Fig. 3) To start a communication between two modules, the BT master module and the BT slave module have to execute a particular host controller interface (HCI) protocol, that allows the slave to be visible to the master and the master to execute the paging sequence. In particular, if the B module is the master of the piconet and the A module is the slave, a master-slave connection can be established if the B module is able to go in a paging scan status and the A module is in a paging mode status, then the connection is set up using HCI commands [14], [15] After creating this connection all the other BT connection (audio, RFCOMM, TCP-IP) can be activated by that.

In the last two years, several BT products have been presented on the market. Consulting technical and commercial web sites, BT modules provided by a software driver for the most common operative systems are widespread. They have been proposed

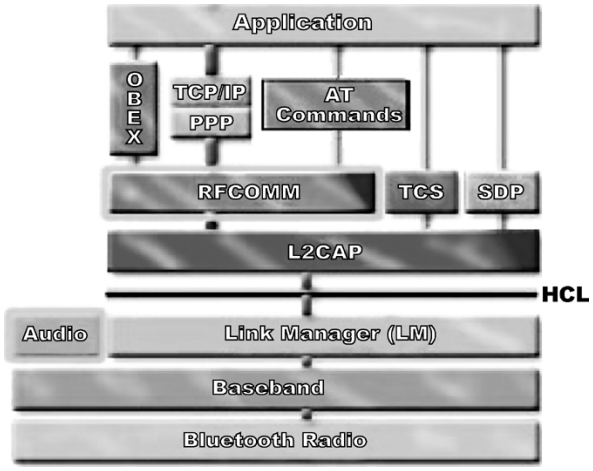

Fig. 3. Example of Bluetooth Stack.

with different host terminal interface units: RS232 or USB interface ports; PCI or PCMCI boards. These BT modules, whatever their host interface, once installed, allow a PC to send BT commands and receive BT events. It is also possible to buy programs that implement the BT software stack. These programs, available for Windows or Linux based PCs, make the creation of a piconet between PCs equipped with BT modules easy, because they automatically manage the BT stack and the PC ports. At the same time, BT modules with no precabled host interface port are also available on the market for customers interested in designing a custom hardware interface link.

III. Bluetooth Instrumentation Bus

On the basis of the aforementioned considerations, in order to allow two PCs to be connected to a BT piconet, the following conditions have to be verified.

1) each PC must be equipped with a BT module (RS232, USB, PCI, or PCMCI interface);
2) the corresponding PC I/O interface port (RS232, USB, PCI or PCMCI) must be programmable;
3) a software that implements the BT stack and manages BT commands and events must be installed in each PC.

These conditions cannot be satisfied if one of the two PCs is substituted with a dummy device (i.e. a measurement instrument). The interface port is not programmable and it is not possible to load any software that manages functions indispensable to creating a BT connection. This means that a dummy device cannot be connected to a BT piconet even if a commercial BT module is fit to its interface (RS232 or USB) port.

If a solution has to be found that does not require to change the dummy device communication hardware, to upgrade its firmware, a PC should be inserted between the BT module and the device, thus, greatly reducing the advantages of the wireless connection.

The authors solved the aforementioned problem for dummy devices equipped by a RS232 serial port by implementing the RFCOMM and the HCI management in a microcontroller-based interface to be integrated with a standard slave BT module. In Fig. 4, a block diagram of the proposed microcontroller interface is shown. It creates a double layer RS232 interface, the

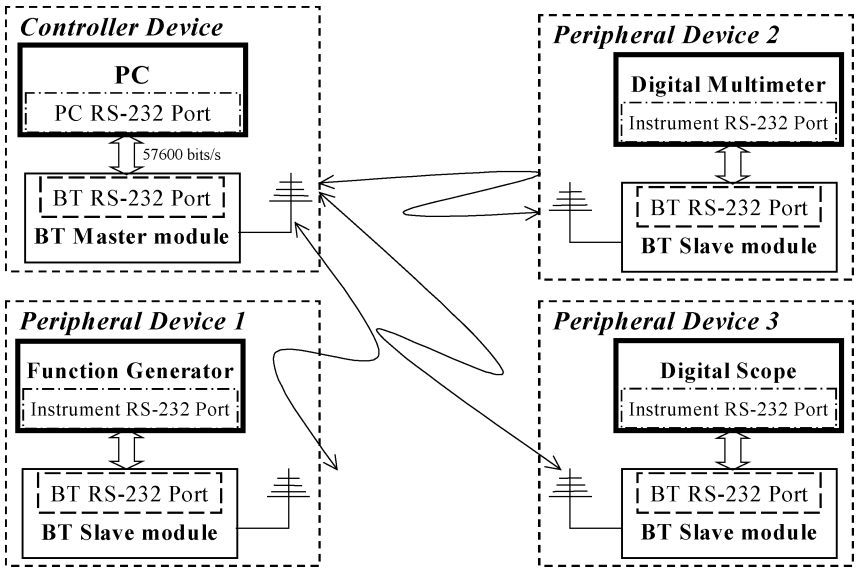

Fig. 4. Proposed microcontroller-based interface.

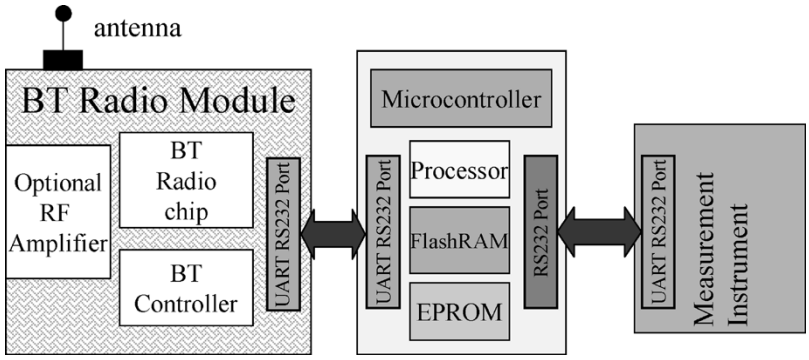

Fig. 5. Block diagram of the proposed Bluetooth instrumentation bus.

former toward the BT module, the latter toward the measurement instrument (or any other device) RS232 port.

This user-friendly and low-cost system allows the BT technologies to be used in creating a wireless connection (piconet) among up to seven dummy devices (i.e. measurement instruments) and a PC controller. It looks like a wireless instrumentation bus, whose block diagram is shown in Fig. 5. Master and slave BT modules are all connected respectively to the RS232 ports of the PC and its peripherals. The PC hosting the master BT module works as a bus controller, while instruments back-fit with the slave BT modules are the peripherals. Device commands and data can be written to and read from each peripheral by the controller. The BT instrumentation bus should be seen as a star bus because it is not possible to directly connect two peripherals. However, the user software can implement an indirect slave to slave communication by making the master buffer the first slave message and dispatch it to the second slave. Finally, it is worth noting that the BT communication structure allows a slave to talk to the master at the same time, and that the number of slaves can be higher than seven if those exceeding this threshold are in park mode. The proposed solutions imply that while the bus controller can be fit in any BT module available on the market, and managed by any commercial BT software stack, the peripheral interface module was designed *ad hoc* and realized for both hardware and software parts.

IV. Master BT Module Software Driver

A suitable software module has been conceived of being installed in the PC and to manage the BT master module. It implements the functions that are necessary to the BT master module in realizing a "plug and play" wireless network where each device can be automatically recognized whenever it goes into the piconet. Further, this software creates and continuously updates a table that reports, for each device recognized in the piconet, the distinctive Bluetooth board address (BD_ADRR), the assigned logical address, and other useful information like instrument type, model, and manufacturer. In this paper, the table will be called the ADDRESSES TABLE. Once the piconet has been assessed, the user can send device commands to connected devices as if hey were RS232 devices onto the PC RS232 port.

The master software runs the following steps.

2) At start up, all the powered-on BT modules standing in the piconet area are detected, recognized and classified in the ADDRESS TABLE. This procedure is then continuously repeated with a repetition period fixed by the user.

2) Once the active slave modules have been found and classified in the ADDRESS TABLE, a BT connection is realized between each slave and the master module.
3) Finally, for each module, the RF_COMM level is assessed, thus, allowing modules in the piconet to be each addressed by the user software just by writing or reading on a corresponding virtual RS232 Com. Thanks to this driver, the piconet configuration is dynamically registered in the ADDRESS TABLE and to each active slave module corresponds a virtual RS232 Com in the host PC.

It is important to note that if a commercial software driver were used, a connection would always be possible, but it would require the user software devoted to the instrument control to be modified to include the BT master module commands.

V. Slave BT Module Serial-to-Serial Interface

A. Hardware

A rough prototype of the realized microcontroller-based interface board is shown in Fig. 6. The heart of the proposed interface board is a Microchip PIC16f877 microcontroller. Its main characteristics are: a 20 MHz operating frequency; 8-kB program ram; 33 I/O pins; two RS232 serial ports [one universal synchronous/asynchronous receiver/transmitter (USART) RS232 port and one synchronous port realized by using the serial peripheral interface (SPI) bus]. The interface USART RS232 port is connected to the slave BT module (not shown in the picture), while the SPI port is connected to the RS232 port instrument. Either a dc power supply or a couple of 9-V rechargeable batteries provides the 1.3 W needed for the whole system. In this prototypal realization, the interface board looks big, due to a lot of unnecessary debugging circuits. The aim of the authors is to contain the final version in a DB9 serial port female package.

B. Software

While the master BT module software is implemented onto the PC, the slave BT module software runs on the realized microcontroller interface module.

The microcontroller software allows any device equipped with an RS232 port to be connected to a Bluetooth piconet as a slave, and to exchange device messages with PC hosting the master BT module. This software realizes two interfaces (Fig. 7): the former forward the BT module, the latter forward the RS232 port of the instrument.

It operates in the following way (Fig. 8). When the microcontroller module is powered on, the RS232 port communicating with the BT module is disabled (Port A), while an "*IDN?" device command is sent on the other port (Port B). In this way, if the instrument respects the IEEE 488.2 standard, useful information such as the instrument type, the manufacturer, and so on are collected in the microcontroller stack memory reserved to the instrument. Successively, the RS232 port communicating with the instrument is closed and no other device command or service request are processed.

After a reset command is sent to the BT module, its response is processed to determine whether the module is connected to the microcontroller or if there is any problem.

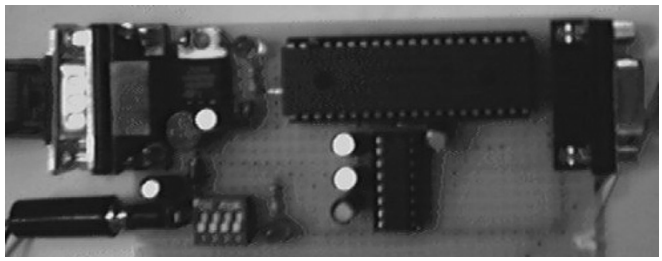

Fig. 6. Photograph of the realized interface board.

If every thing is ok, the software enables the BT module to be discovered, for the time required for it to be located by the master BT module. When the master module sends an inquiry command, the slave module is then able to give its board device address (BD_Addr). If the instrument is an IEEE 488.2 device, further information is passed to the master and collected here in the peripheral information stack.

Here, the software module manages all the BT commands and events to create a connection. This is possible because a small BT ISO-OSI stack was implemented in the microcontroller module. This stack contains the minimum instruction set able to create a BT connection.

Once the connection is created and a physical channel established, the slave software continuously monitors both the instrument and the BT module RS232 ports. If a byte stream is received on the BT module port, it determines if it represents a command or an event. In case of command, it is memorized on an internal stack, and the payload is pushed into another stack called an instrument stack, where it is finally popped to be sent to the instrument. If the received byte stream represents a BT event, the microcontroller module manages it properly. If a data stream is received on the instrument RS32 port, it is pushed onto the instrument stack, then it is translated into a BT stream. Finally the BT packet is sent to the BT module. A BT packet contains appropriate header, payload, and final byte, which are necessary to decode the BT instruction.

It's worth noting that the implementation of the BT stack in the module required a hard test phase, because it was necessary to experimentally find out the minimum instruction set and the right timing sequence necessary to create a BT piconet. In addition, since the BT module writes on the first microcontroller serial port, the whole BT packet (excluding only the baseband information), the structure of each BT command or event has been studied in order to make the microcontroller module able to manage the BT connection and extract the device command sent to the instrument. In this way, the realized software is absolutely general in purpose, and not component based, because the data sent to the first serial port are in BT standard while the data received on the second serial port are pure device commands.

Thanks to the aforementioned association between device logic address and slave BT module physical address made by the PC software driver in the ADDRESS TABLE, the master is always able to address the slave to send device messages to the instrument and receive data from it. The slave BT module can get the attention of the controller, using software service request routines.

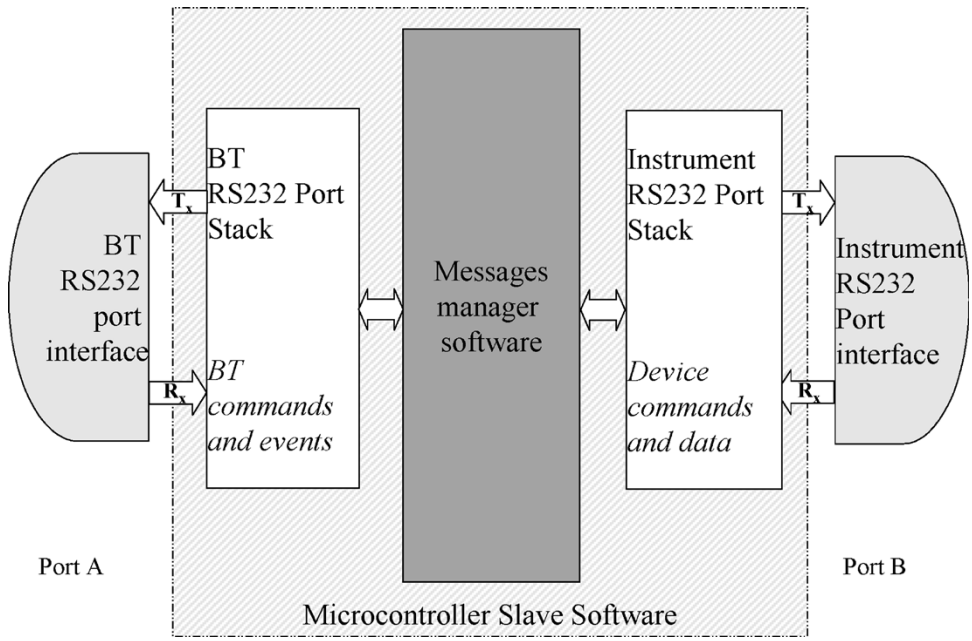

Fig. 7. Slave software organization.

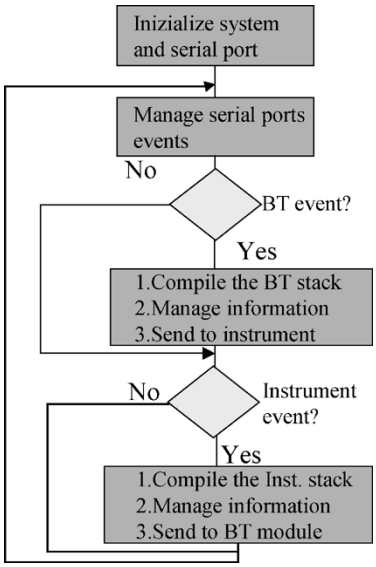

Fig. 8. Block diagram of Messages manager software.

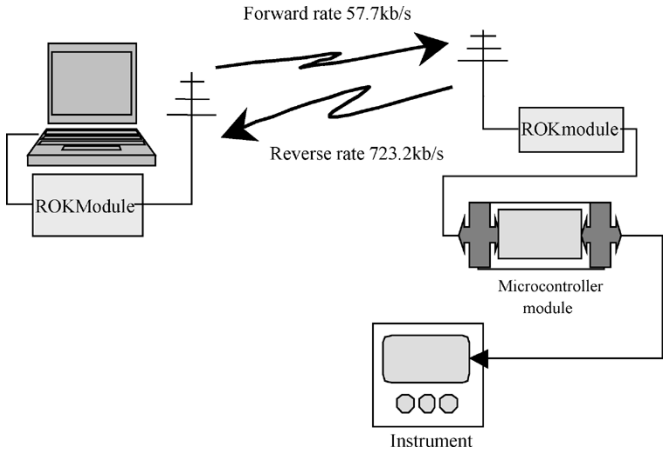

Fig. 9. Realized automatic measurement station.

TABLE I
EXCECUTION TIMES EXPRESSED IN MILLISECONDS

T0	T1	T2	T3	T4	T5
52.9	~0	47.3	~0	52.9	7.8

An example of service request management is also reported. The master always starts the transmission in the even-numbered slots, whereas the slaves start their transmission in the odd-numbered slots. If the slave module, addressed by its AM_ADDR, needs to send a service request, it writes a special request message in its assigned odd-numbered slot. The master module put this request in a message stack and starts the interrupt routine. Finally, all service requests present in the stack are processed using a round-robin algorithm.

VI. PERFORMANCE ANALYSIS

Once proposed, the instrument interface needs to be analyzed in terms of: 1) transfer rate, costs, maximun connected devices, software requirements and so on; and 2) comparison to competitive approaches.

As far as the former point 1) is concerned, many tests on the proposed interface are still being run, but some results are already available, while as for the latter point 2), some considerations about other products available on the market are given.

As described in the BT specification core, the throughput of BT systems is strongly dependent on the link type, the packet type, and the state of each module in the piconet. Therefore, an accurate analysis would require that a high number of combinations of these variables were experimentally explored. For the sake of brevity, only a limited set of configurations and tests can be made to be reported in this paper. As an example, in the following, some results about a wireless connection between a

TABLE II
COMPARISON BETWEEN THE PROPOSED BT INSTRUMENTATION BUS AND THE MAJOR WIRED COMPETITORS

Specification	BT	IEEE 488	EIA RS485
Maximum number of connected devices	255 (between active and parked)	20	32
Maximum Distance	100 m (10mW) 10 m (1mW)	20 m	1200 m
Transfer rate	1 Mbs	1 Mbs to 100 Mbs (HS IEEE 488)	10 Mbs(short distance <1m) to 100 bps (long distance)
Bus Structure	STAR (byte serial; bit serial)	PARALLEL (byte parallel; bit parallel)	PARALLEL (Byte serial; bit serial)
Cost	50 USD	200 USD	40 USD

peripheral device (i.e. an instrument) and a controller using ACL DH5 packets (Fig. 9) are reported. In particular, the test station is composed of a PC, ruling as the bus controller, connected with a Master Ericsson ROK BT module, and a peripheral (Fluke 45 dual display multimeter) back-fit on its RS232 port by the proposed microcontroller-based interface, equipped with another ROK module.

Two possible situations were tested:

1) The controller sends device messages to the instrument;
2) The controller reads data from the instrument.

Total and partial transfer times were measured in each case.

Case 1) deals with a controller that sends 339 B to the instrument (399 B is the maximium DH5 user payload). The measured total transmission time T_{totalW} is about 200 ms. $T_{\mathrm{totalW}} = T0 + T1 + T2 + T3 + T4 + T5$ where: $T0$ is the time that the RS232 PC serial port employs to dispatch data on the RS232 ROK serial port; $T1$ is the time the ROK module needs to create the radio baseband packet; $T2$ is the time the packet spends to reach the other BT ROK module; $T3$ is the time the ROK module needs to clear the HCI packet from the radio baseband packet and make the packet ready on its serial port; $T4$ is the time requested to transfer data between the ROK and the microcontroller RS232 serial port; $T5$ is the time the microcontroller module spends to make data available on the instrument RS232 port. Their values are reported in Table I.

In case 2), the same length data packet is read from the controller. The measured time T_{totalR} was in this case equal to about 160 ms. The sum of terms is the same of case 1), but in the inverse sequence. The difference between T_{totalW} and T_{totalR} is due to the asymmetry of the packet which grants 57.6 kB/s in forward connection, while 723.2 kB/s in reverse (slave/master) connection. The transfer rate is quite low if compared to the theoretical value of IEEE-488, but it is competitive with wired serial busses. The bottleneck is constituted by the instrument RS 232 port which, in the best case, could be able to transmit and receive at 24000 baud. Better performance will be achievable when the USB ports of PC and instruments are used instead of the RS232 ports. Other figures of merit strictly depend on the BT communication protocol and are briefly summarized in the Appendix. Finally, some considerations can be made about emissivity of the interface device. Specific EMC tests are ongoing in the anechoic chamber, but a simplified test was made. In the above mentioned connection with a PC controller, a Fluke 45 digital multimeter was used as peripheral. The multimeter was set up and triggered via the BT connection to execute a set of 100 dc voltage measurements on a dc 1,000 V reference. Results were collected via the BT connection by the PC controller and used to evaluate mean and standard deviation. These two values compared to those obtained from the measurements made in absence of the BT connection did not exhibit significant differences.

Natural competitors of the proposed instrumentation interface are either wired solutions like the parallel IEEE-488 and the serial EIA-RS485 bus, or other wireless connection devices available in the market.

As far as the wired competitors are concerned, Table II shows a comparison among IEEE 488, EIA RS485 and BT main specifications. As it is possible to see, while BT allows to connect a greater number of devices to the piconet (by using both the active and the parking state) the IEEE 488 and EIA RS 485 buses allow a high transfer rate to be reached. Moreover, the cost of the BT interface is comparable to that of the cheapest wired, and, instead of the other instrument interfaces, a new BT peripheral is automatically included into the piconet at power up without requiring any software manipulation. As a conclusion, whereas a wireless connection is needed, the load that the proposed BT interface requires to pay in terms of reduced transfer rate is minimum if compared to the increment of flexibility.

On the other hand, the current market offer of wireless connections among electronic devices is quite wide. In particular, as far as other BT solutions are concerned, recently Impulsesoft has launched onto the market a product called ISPA 2000, that should be able to allow a wireless connection between a PC and a dummy device, in substitution of the serial cable. A great limit of this product is that it is only able to make a one-to-one serial link replacement; this means that it does not allow realizing a piconet. Further, this product is published but not yet sold. A valid competitor of the proposed BT interface is the IEEE 802.11 [wireless local area network (WLAN)] wireless connection. On the market are present IEEE 802.11 products that allow 10 Mbs to be transferred at a distance of some kilometers using 50-mW powered modules. This very good performance, however, implies a cost of about $2000 USD for each device that cannot be accepted in designing automatic measurement systems. Thus, the proposed BT interface can be viewed not as a WLAN competitor but as a WLAN integration for short-range applications. Recently, National Instruments launched onto the market an Ethernet IEEE 488 controller module (GPIB ENET), which allows the remote control of up to 14 GPIB devices connected to a GPIB-TCP-IP interface. It is clear that this product is conceived with the aim of allowing the Ethernet control of a wired IEEE 488 measurement station.

TABLE III
DHx, THROUGHPUT WITH/WITHOUT INTERFERENCE (kB/s)

	Ideal Conditions	Without Interference	With Interference
DH1	172.80	166.66	120.78
DH3	384.00	373.32	329.40
DH5	432.60	417.24	373.32

VII. CONCLUSION

A wireless instrument interface bus is proposed. It is based on the BT protocol and allows up to seven peripherals at a time to be active and connected to a controller. Different interface devices are required for controller and peripherals, but both can be fit into their RS232 ports without any hardware modification. Only a software driver must be loaded onto the host PC, which allows each peripheral to be accessed by user software as a virtual serial port. First experimental tests show transfer rates in writing and reading device messages to and from a peripheral that are comparable to wired serial buses. Future improvements concern the use of USB ports instead of RS232 ports in order to improve the data transfer rate, and the implementation of a BT multicast protocol in allowing the controller to simultaneously access a set of peripherals.

APPENDIX

BT is a wireless communication system that operates indoors using low-power transmission; consequently, it suffers from fading, interference, path loss, and presence of noise on the channel. BT faces these problems using both hardware and software solutions.

The former solution is realized through a frequency hopping scheme, and using a gaussian frequency shift keying (GFSK) modulation that allows only two symbols to be sent. The possibility of transmitting over 76 different frequency channels allows reducing the probability of code loss or collision between packets. The latter solution is realized through both the use of a cyclic redundancy code (CRC) at the end of each packet, and event-based acknowledge messages sent by the slave [17], [18]. Results in the presence of additive white gaussian noise (AWGN) noise are shown in Table III. It can be noted that when data packets are corrupted by noise, the throughput decreases because the frequency jump rises and the data packet must be resent.

ACKNOWLEDGMENT

The authors wish to thank Dr. F. Giacobino for his help given in the development of the software.

REFERENCES

[1] P. Arpaia, A. Baccigalupi, and A. Pietrosanto, "Performance optimization of VXI-based measurement stations," *IEEE Trans. Instrum. Meas.*, vol. 44, no. 3, pp. 828–831, Jun. 1995.

[2] L. Angrisani and A. Pietrosanto, "Performance comparison of IEEE-488 and 1155 based measurement stations," in *Proc. Atti del VII IMEKO TC-4* Prague, Czech Republic, Sept. 1995, pp. 634–638.

[3] M. Gooding, "VXI plug'n pray," in *Proc. AUTOTESTCON —IEEE Systems Readines Tech. Conf.*, Aug. 1998, pp. 448–451.

[4] F. Alegria and H. G. Ramos, "A remote controlled automated measurement system," in *Proc. IEEE IMTC*, Ottawa, ON, Canada, May 1997, pp. 1186–1189.

[5] D. Aviles and O. Gutierrez, "Automatization of the multifunctions laboratory," in *Conf. Precision Electromagn. Meas. Dig.*, 1998, pp. 199–200.

[6] S. V. Wunnava and P. Hoo, "Remote instrumentation access and control (RIAC) through internetworking," in *Proc. IEEE Southeastcon*, Mar. 1999, pp. 116–121.

[7] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, May 2001, pp. 2033–2036.

[8] K.-S. Tang, K.-F. Man, and S. Kwong, "Wireless communication network design in IC factory," *IEEE Trans. Ind. Electron.*, vol. 48, no. 2, pp. 452–459, Apr. 2001.

[9] M. Dunbar, "Plug-and-play sensors in wireless networks," *IEEE Instrum. Meas. Mag.*, vol. 4, no. 1, pp. 19–23, Mar. 2001.

[10] U. Bilstrup and P. A. Wiberg, "Bluetooth in industrial environment," in *Proc. IEEE Intern. Workshop Factory Communication Systems*, Sep. 2000, pp. 239–246.

[11] J. C. Haartsen, "The Bluetooth radio system," *IEEE Pers. Commun.*, vol. 7, no. 1, pp. 28–36, Feb. 2000.

[12] *Specification of the Bluetooth System, vol. 1, Bluetooth Version 1.1*, Feb. 22, 2001.

[13] *Specification of the Bluetooth System, vol. 2 profiles, Bluetooth Version 1.1*, Feb. 22, 2001.

[14] *Bluetooth Security Architecture Version 1.0 Bluetooth WHITE PAPER*, Jul. 15, 1999. 1.C.116/1.0.

[15] *Bluetooth Protocol Architecture Version 1.0 Bluetooth WHITE PAPER*, Aug. 25th, 1999. STATUS 1.C.120/1.0.

[16] D. Famolari and P. Agrawal, "Architecture and performance of an embedded IP Bluetooth personal area network," in *Proc. IEEE Int. Conf. Personal Wireless Communications*, Morristown, NJ, Dec. 2000, pp. 75–79.

[17] A. El-Hoiydi, "Interference between Bluetooth networks—upper bound on the packet error rate," *IEEE Commun. Lett.*, vol. 5, no. 6, pp. 245–247, Jun. 2001.

[18] N. Golmie, R. E. Van Dyck, and A. Soltanian, "Interference of Bluetooth and IEEE 802.11: simulation modeling and performance evaluation," in *Proc. ACM Int. Workshop Modeling, Analysis, Simulation Wireless Mobile Systems*, Rome, Italy, Jul. 2001, pp. 553–558.

Luigi Ferrigno was born in Nocera Inferiore, Italy, in 1972. He received the M.S. degree in electronic engineering from the University of Salerno, Fisciano, Italy, in 1998, and the Ph.D. degree in electrical engineering from the University of Napoli, Naples, Italy, in 2002.

Since 2001, he has been an Assistant Professor of Electrical and Electronic Measurements at the University of Cassino, Cassino, Italy. His current research interests are concerned with measurement of R, L, and C parameters in nonsinusoidal conditions, wireless and wired sensor realization and characterization, and realization of measurement systems for nondestructive testing via eddy currents.

Vincenzo Paciello was born in Salerno, Italy, in 1977. He received the M.S. degree in electronic engineering, University of Salerno, Italy, in 2002. He is currently pursuing the Ph.D. degree in electronic engineering at the University of Salerno.

His current research interests include wireless sensors and instrument interfaces and image processing-based measurements.

Antonio Pietrosanto (M'99) was born in Naples, Italy, in 1961. He received the M.S. and P.h.D. degrees in electrical engineering from the University of Napoli, Naples, Italy, in 1986 and 1990, respectively.

In 1991, he became an Assistant Professor, and in 1999, an Associate Professor of Electrical and Electronic Measurements at the University of Salerno, Fisciano, Italy, where he has been a Full Professor of Electrical and Electronic Measurements since November, 2001. His scientific interests are principally concerned with instrument fault detection isolation and accommodation, digital signal and image processing for real-time diagnostic and process control, and fiber-optic sensors.