Low-cost workbench client / server cores for remote experiments in electronics

José M. M. Ferreira[1,2], Américo F. S. Dias[1,3], Paulo J. S. Sousa[1]

[1] Universidade do Porto - Faculdade de Engenharia, Porto, PORTUGAL
[2] Buskerud University College, Kongsberg, NORWAY
[3] Instituto de Engenharia de Sistemas e Computadores, Porto, PORTUGAL

Zorica Nedic[4], Jan Machotka[4], Ozdemir Gol[4], Andrew Nafalski[4]

[4] University of South Australia, Adelaide, AUSTRALIA

Abstract— **This paper offers an open-source solution to implement low-cost workbenches serving a wide range of remote experiments in electronics. The proposed solution comprises 1) a small (9,65 x 6,1 cm) Linux server board; 2) a server core supporting two TCP/IP communication channels, and general purpose I/O pin drivers to interface the remote experiment hardware; and 3) a client core based on a multi-tab user interface supporting text file management to exchange experiment scripts / status information, and a mini-browser for webcasting in distance learning scenarios, live image feedback from IP cameras located in the remote workbench, etc. Additional drivers and application-specific tabs can be added to the server / client cores, to suit the requirements of each application. Two remote workbenches for microcontroller and digital systems testing courses were developed, and are also presented in this document.**

Index Terms—**Remote workbenches, virtual laboratories, microcontrollers, boundary-scan.**

I. INTRODUCTION

The development of remote laboratories occurred rather randomly during the 1990s and throughout this last decade, but it is possible to perceive a standardisation trend in recent years, particularly along the MIT iLabs architecture [1] (initiatives to implement various iLabs-compatible solutions are currently under way in Europe and Australia). Acceptance of a common standard in this area will benefit institutions and users, and enable content and resource sharing among partner universities. However, the resources required to set up an iLabs-compatible architecture may not be within reach of small institutions, or perhaps will not be justifiable in cases where only a small number of remote experiments is envisaged. A low-cost solution would be preferable for these scenarios, particularly if based on open-source resources that might be reused and easily adapted to the needs of each institution.

This paper describes a low-cost workbench server built upon the ICnova AP7000 Base board, and the corresponding client interfaces. At a cost of 95 EUR, this AVR32 32-bit MPU based Linux board offers an ideal solution to implement a low-cost remote workbench server, easily adapted to various application areas. To achieve this objective, two main development tasks had to be undertaken: 1) the board drivers and server code to interface the experiment hardware; and 2) a client interface adaptable to each application area.

The following section introduces the ICnova AP7000 Base board and describes all the development work that was done at this level. Section 3 presents the client interface and explains how it can be customised to different experimentation domains. Sections 4 and 5 illustrate two remote workbenches that were built using the proposed AP7000 hardware and client interface (to support microcontroller applications and digital systems testing). A final section dealing with conclusions and further research directions closes the paper.

II. THE ICNOVA AP7000 BASE BOARD

This section introduces the Linux board that was used to implement the remote workbench server, and explains the development work needed to customise the workbench to the selected experimentation domains.

A. Board resources

The ICnova AP7000 Base board, also known as "grasshopper", is shown in figure 1.
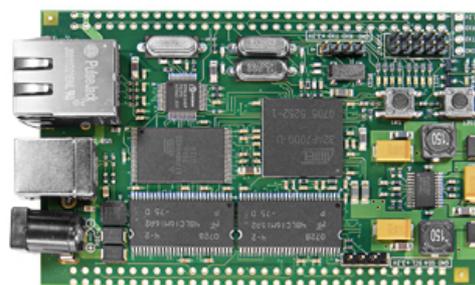
Figure 1. The ICnova AP7000 Base board (9,65 x 6,1 cm)

Its resources make it rather popular for developing embedded projects within the Linux community [2,3]:

- Software development support includes a GNU C compiler (http://gcc.gnu.org/), a C library optimized for embedded systems (http://www.uclibc.org/), small executable modules containing many common UNIX utilities (http://busybox.net/), a telnet daemon for remote command line access, Dynamic Host Configuration (DHCP), HTTP server, and full control over the general purpose input / output (GPIO) pins and other devices

1

- All Linux sources are open source and delivered on the accompanying CD
- Main hardware features: The board has a 32-bit data bus, runs at 140 MHz (max. 200 MHz), offers 64 MB SDRAM and 8 MB Flash RAM, a USB-UART, 10/100 Mbps Ethernet, 64 GPIOs, an I2C bus, and a built-in step-down voltage regulator.

B. Server core

All remote experiments rely on a core server that was written in ANSI C and compiled using GCC. Two communication channels using different TCP/IP ports were created – one bidirectional channel used to exchange commands / responses between server and client, and a unidirectional channel that enables the server to send data to the client application. This core is based on a multithreading TCP/IP socket server that launches two independent threads to monitor all connection requests to any of the two TCP/IP ports referred above. For each successful attempt, two threads are created to process the information generated by the connection associated to the requesting client. The first client obtains control of the server, and holds it until the end of its session (either by closing the connection, or by timeout in the case of connection problems). All clients attempting to connect during an on-going session are rejected with a BUSY response.

A client that obtained control of the server is obliged to send PING commands at no longer than 30s intervals, to ensure that the connection is working properly, and is then able to send commands and retrieve responses.

III. THE CLIENT SOFTWARE

In this section we will present the client interface that enables access to our remote workbench server, and indicate how it can be adapted to suit the specific needs of each experiment.

A. Interface design

The client software was written in Visual Basic .Net 2008 and offers an easy-to-use interface to all server applications. The functional core of the client interface comprises various windows selectable by the corresponding tabs. According to each specific remote experiment, further tabs can be added to the basic core, which includes a text editor window (to edit and exchange information with the server), and a mini-browser window (to enable integrated web casting / conferencing, live image feedback from the remote workbench, etc.).

B. Core modules

The core client code integrates the two TCP/IP communication channels that are also present in the server core (see section 2.2): 1) a bidirectional synchronous channel to exchange commands and 2) an asynchronous channel for data transfer. A text editor module is also available, supporting parser and syntax check routines that can be adapted to all applications where the remote experiment is controlled by a (text) script.

The client core code also comprises a graphic engine to draw the digital waveforms at a selected subset of the ICnova AP7000 Base board GPIO pins. The waveforms window supports zooming in and out, and offers a "detach window" option to enable observation together with other tab windows.

The two following sections will present remote workbenches built upon the proposed low-cost server and client interface software, supporting experiments in microcontroller programming, and digital systems testing.

IV. CASE STUDY #1: 80C51 MICROCONTROLLER CLASSES

A. Experiment description

Microcontroller programming is an excellent case study to illustrate the application of the proposed low-cost workbench server and client interface software. Students attending microcontroller courses are typically given lab assignments comprising a general purpose microcontroller board, and additional hardware to achieve the experiment objectives. An electronic dice is a good example of such assignments. In this case, four microcontroller parallel output pins control seven LEDs representing the dice dots, which should cycle through the six possible results shown in figure 2.

PP.1	0	1	0	1	0	1
PP.2	1	1	1	0	0	0
PP.3	1	1	1	1	1	0
PP.4	1	0	0	0	0	0
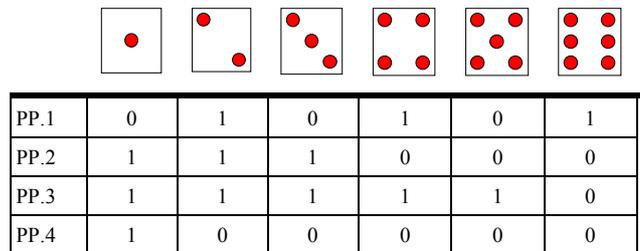
Figure 2. Parallel port patterns corresponding to the six e-dice results

The patterns sent to the microcontroller parallel port will repeat the six results shown in figure 2, until a STOP button is pressed. Since the display rate is very high, the seven LEDs seem to be ON at the same time, and freeze into a specific result when STOP is pressed.

The microcontroller program will be written, compiled and simulated off-line, e.g. using the KEIL's mVision environment illustrated in figure 3 [4].
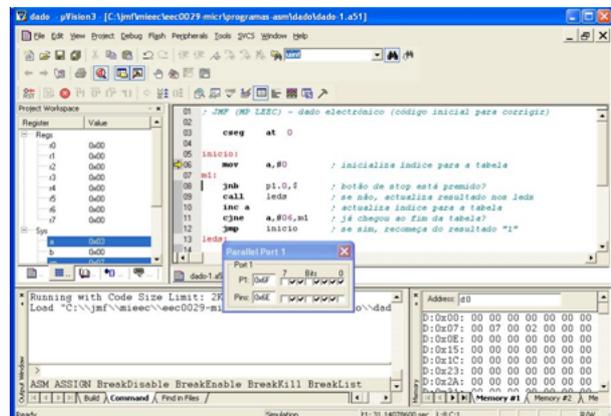
Figure 3. KEIL's μVision environment for off-line tasks

A remote workbench to enable microcontroller programming experiments shall therefore comply with the following requirements:

1. Upload data or code files
2. Run the uploaded code
3. Set up external conditions (e.g. the STOP button in the e-dice example)
4. Provide a live video stream from the remote workbench

To validate the proposed remote microcontroller programming workbench, a general purpose 8051 (an 8-bit microcontroller originally developed by Intel) card offering 4 digital inputs and 4 digital outputs was used to implement the e-dice experiment. The card receives the object code via a serial RS232C connection, which is also used to exchange status and control information. This serial link is managed by our ICnova AP7000 Base workbench server, which exchanges data and information with the client interface installed at the user's computer.

B. Server customisation

Further to the server core that was described in section 2.2, the ICnova embedded device drivers were used to read and write to the GPIO pins. Since the GPIO outputs are only used to emulate push-buttons connected to 4 parallel inputs of the 8051 microcontroller, the relative low-speed of these drivers does not degrade the workbench performance. Embedded drivers were also used to establish an RS232C serial channel between the ICnova board and the 8051 microcontroller hardware (used to send object code and to exchange commands and status information).

C. Client interface

The client core described in section 3 was adapted to the requirements of the microcontroller workbench, producing an interface that offers three tabs, and application-specific buttons on the right side. The hex file containing the object code to be executed is loaded using the "Select File" button, and is displayed in the middle tab window shown in figure 4.
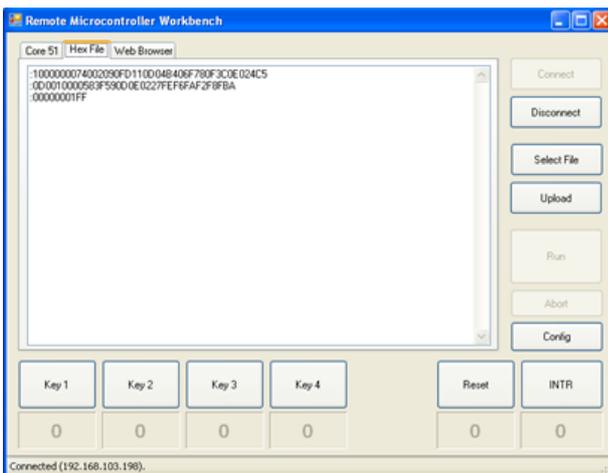
Figure 4. Remote microcontroller application –
File transfer window

The object code can then be sent to the remote workbench using the "Upload" button shown in figure 5. The information transfer window (left tab in the client interface) shows the response of the microcontroller board, indicating that it waits for an order to start execution – pressing "Space" in a local keyboard, or "Run" in the client application.

When the start order is received, the microcontroller board starts to execute the uploaded program, and the corresponding status information is sent to the client and displayed as shown in figure 5.
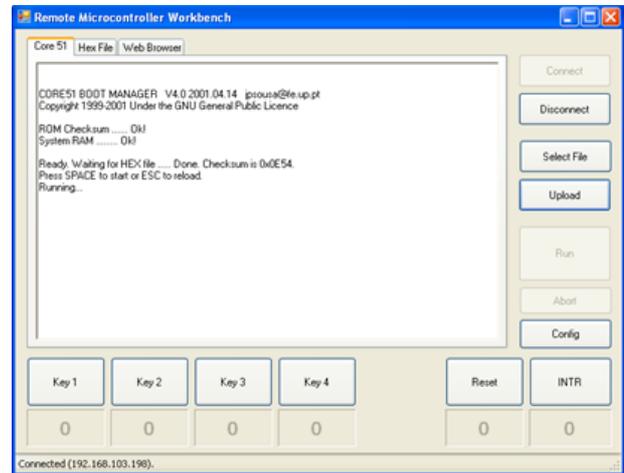
Figure 5. Remote microcontroller application –
Information transfer window

In the case of the e-dice experiment, the uploaded code cycles continuously through the six possible results shown in figure 2, until the STOP button – "Key 1" in the client interface – is pressed. When that happens, the combination currently driven to the LEDs freezes and e-dice result becomes available.

The rightmost tab selects a mini-browser that can be used to watch the video stream provided by an IP camera located in the remote workbench. In the case illustrated in figure 6, the STOP button had been pressed, and the e-dice result was "5".
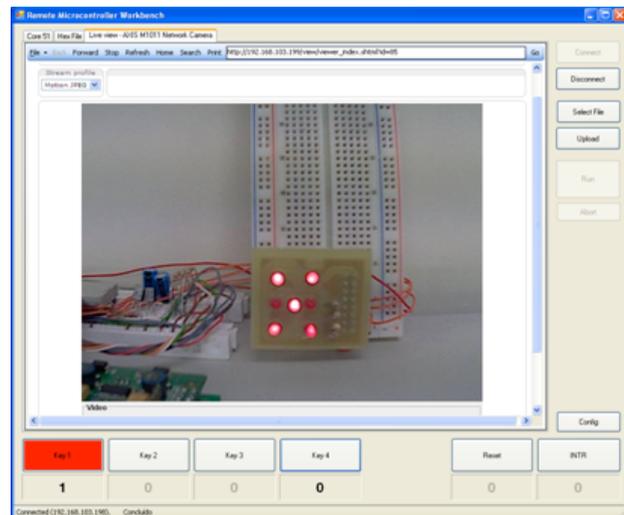
Figure 6. Remote microcontroller application –
Live video from the remote workbench

The mini-browser window may also be used for other purposes, besides showing the video stream produced by the remote workbench IP camera. Figure 7 shows it being used to run a Dimdim session [5], webcasting a presentation where the lecturer explains the remote e-dice experiment.

3

Figure 7. Remote microcontroller application –
Mini-browser window showing Dimdim

V. CASE STUDY #2:
BOUNDARY-SCAN TEST (BST) CLASSES

A. Experiment description

Boundary-scan is a test technology developed in the mid-1980s and approved as IEEE standard 1149.1 in 1990 [6]. Every 1149.1-compatible chip comprises a set of test cells placed in the device boundary, enabling observation and control of every functional pin. Access to the test infrastructure is done through a 4-pin test access port (TAP), ensuring a common protocol to all test data operations (shifting, capturing test responses, application of test vectors), irrespective of the device or its manufacturer. These 4 pins enable data shifting (TDI and TDO to shift in and out of each device), control (TMS to select the required test mode), and timing (TCK for test clock). Each device possesses an instruction register (IR), present in the same scan chain, which specifies the required operating mode for the test logic.

Test generation is done automatically from the printed circuit board netlist, the description of the BS infrastructure present in each device, and eventual test vector sets generated for clusters of non-BS devices. The complete set of test vectors is then serialised, and the binary test vector streams represented using SVF (Serial Vector Format) [7].

The example illustrated in figure 8 comprises two circuits, each of them with 8 functional pins (and the associated 8 BST cells). To find out if a short-circuit exists between two interconnects, opposing logic values shall be shifted into the two driving BS cells, and the responses captured at the corresponding inputs shall be shifted out and checked against their expected values.
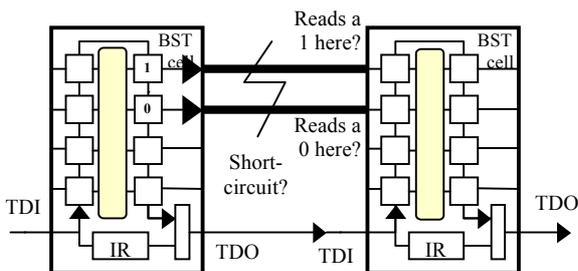
Figure 8. A simple circuit comprising two BS devices.

The corresponding SVF code for this case might be represented as follows:

```
STATE RESET
    ; initialise the test logic
SIR 8 TDI(00)
    ; set the two devices in external test mode
    (assumes 4-bit IRs)
SDR 16 TDI(0800)
    ; shift in the test data for the two driving
    cells
SDR 16 TDI(0800) TDO(0010) MASK(0030)
    ; shift out the test response and check the
    two receiving cells
```

A remote workbench to enable practical BST experiments shall therefore comply with the following requirements:

1. Provide text editing features to load / edit the SVF test code
2. Execute the SVF test code in "step" / "run" modes
3. Observe the waveforms in all TAP pins (logic analyser)

In the case of mixed-signal circuits, where analogue signals are controlled by the digital BST devices, video streaming from the remote workbench will enable the users to visualise an oscilloscope or other measurement equipment.

B. Server customisation

Since all TAP signals come from ICnova GPIO pins, the corresponding server code dictates workbench performance. It may be necessary to shift long bit streams through the remote BST hardware, so the frequency of the test clock (TCK) signal should be as high as possible to minimise experiment latencies.

Since the embedded ICnova GPIO drivers do not go beyond a few hundred Hz, new device drivers were written to execute all IO operations in kernel space. Figure 9 summarises how these device drivers work, and the interaction between user and kernel space.
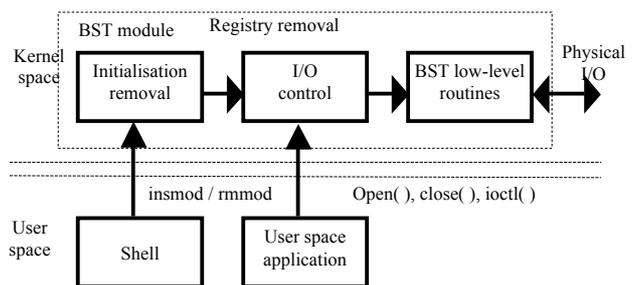
Figure 9. Device drivers and user / kernel spaces interaction

The device driver / kernel module must be loaded dynamically via the command shell using the instruction insmod (this task can be done automatically when booting the system). Once this module is activated, it can interact with the user space application using the available system calls, e.g. open( ), close( ), ioctl( ), etc. This solution enables all low-level control functions to be executed by the device driver in kernel space, and the high-level operations to be executed in user space.

C. Client interface

The requirements set up for the BST workbench led to the design of a client interface offering four tabs and an application-specific set of buttons on the right side. The SVF test code corresponds to the experiment script, and is displayed in the text editor window illustrated in figure 10. File management functions (Open, Save, Save as) are provided on the bottom-left corner ("Tools"). When ready, the SVF code can be executed step-by-step, or completely in one run, using the corresponding buttons shown in figure 10.
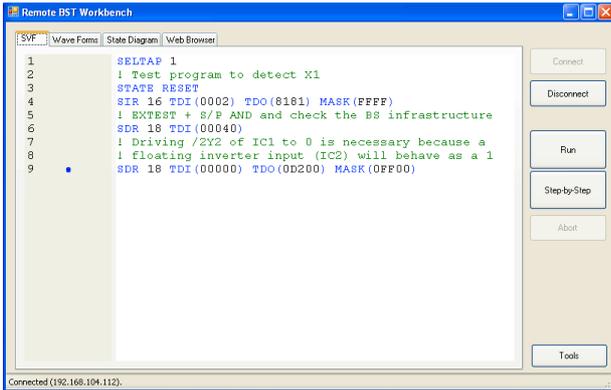
Figure 10. BST controller application – the SVF test program

The BST workbench client interface uses the graphic engine provided by the core client to produce a waveform display window showing the digital signals present in the two sets of TAP pins. Following the execution of SIR (Scan Instruction Register), SDR (Scan Data Register), or of any other commands generating TAP activity, this window enables the users to see the effect of every SVF command on each pin. SVF line numbers are indicated below the sets of waveforms associated to each TAP, as illustrated in figure 11. The zoom function, available on the bottom left part of this window, allows the user to see longer segments, or to analyse finer details.
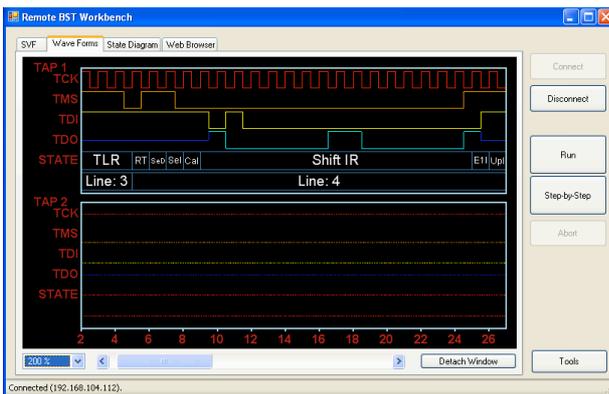
Figure 11. BST controller application – waveform display

The operating mode of the BST test logic inside each chip is specified by an instruction shifted into the instruction register, in combination with the state of a small finite state machine called the TAP controller (shifting test data or instructions only takes place when the TAP controller is in the corresponding "Shift-XR" state). An additional tab was therefore added to the client BST

controller application, to show the current state of the TAP controller, as illustrated in figure 12.
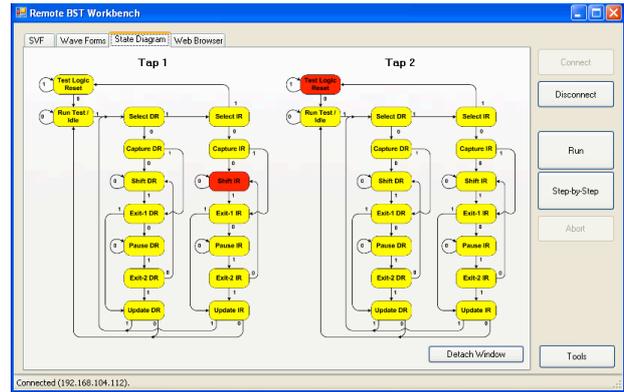
Figure 12. BST controller application – TAP state diagram

Live video feedback from the remote workbench may or may not be necessary for BST experiments. In the case of strictly digital test experiments, the waveforms and the data shifted out of the board under test will contain all the necessary information. When live video is required, the stream produced by an IP camera can be visualised in the mini-browser, as illustrated in figure 13.
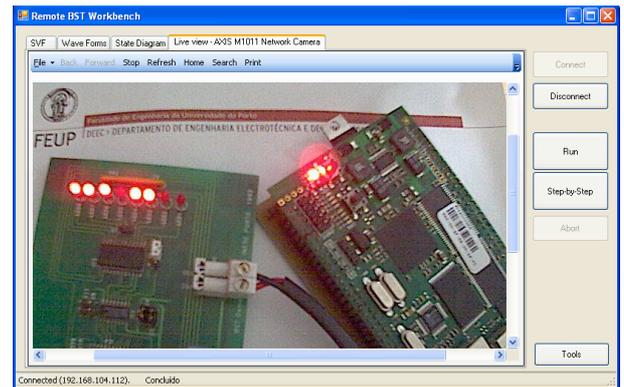
Figure 13. BST controller application –
Live video from the remote workbench

Likewise, this mini-browser window can also be used by the lecturer to webcast a demonstration / presentation of structural test detection in distance learning scenarios.

VI. CONCLUSION AND FURTHER RESEARCH

This paper presented a low-cost solution that enables a quick implementation of remote workbenches for a wide variety of practical experiments in science and technology courses. The workbench server is based on a Linux ICnova AP7000 Base board, selling at unit prices of 95 EUR. The client interface was written in Visual Basic .Net 2008 and can be easily customised to suit any required experiments. The proposed solution offers two main advantages: 1) low-cost; 2) easy reusability. The server and client open source code sets are available for the two workbenches from the following web addresses (Code license: GNU General Public License v3; Content license: Creative Commons 3.0 BY-SA):

- Server applications: http://code.google.com/p/rmws/ and http://code.google.com/p/rbstws/
- Client applications: http://code.google.com/p/rmw-client/ and http://code.google.com/p/rbstw-client/

For some types of remote experiments, it is also possible to envisage a low-cost workbench server based on a netbook computer, at a price similar to our proposed solution (the netbook webcam might be combined with a video streaming application to provide live images from the remote experiment). Although the prices of these two solutions would be similar, the netbook is far more limited in terms of general I/Os available to interface the experiment. If an external board is needed for this or other purposes (e.g. to provide a serial port connection to the experiment hardware, as happens in one of our examples), then the netbook approach clearly looses in favour of our proposed solution. Additionally, the ICnova has lower power consumption, and occupies less space.

It is important to stress the fact our ICnova-based solution is able to run multiple remote experiments at the same time, provided that the requested server resources (e.g. IO pins, serial port, etc.) do not overlap. This is what happens with the two workbenches presented in this paper, which can be used at the same time. Simultaneous use of multiple workbenches is an added-value in relation to in-presence experiments, where a lab bench cannot be used by more than one group of students at the same time.

Our remote workbench does not compete with higher-end systems, such as those based on the iLabs architecture or National Instruments' ELVIS platforms [8]. On the other hand, it represents a satisfactory solution when the economic resources available are scarce, or the number of experiments does not justify the overhead of a full-fledged solution. The mini-browser included in the client interface enables the integration of web video-conferencing to support collaborative work (e.g. using Adobe Connect [9]), or webcasting to demonstrate practical experiments in distance / e-learning courses (e.g. using Dimdim). With a complementary IP camera, our proposed solution offers a complete lab environment supporting live video streaming, at a cost that is far below most other systems.

ACKNOWLEDGMENT

REFERENCES

[1] *iLabs: Internet access to real labs - anywhere, anytime*, available on-line at http://icampus.mit.edu/ilabs/ (visited on March 5th 2010).

[2] *ICnova AP7000 Base*, available on-line at http://www.ic-board.de/product_info.php?language=en&info=p75_ICnova-AP7000-Base.html (visited on March 5th 2010).

[3] *icnova ap7000 base*, available on-line at http://www.avrfreaks.net/index.php?module=Freaks%20Tools&func=viewItem&item_id=874 (visited on March 5th 2010).

[4] *µVision® IDE & Debugger*, available on-line at http://www.keil.com/uvision/ (visited on March 5th 2010).

[5] *Dimdim for live meetings, demos and webinars*, available on-line at http://www.dimdim.com/ (visited on March 12th 2010).

[6] *IEEE 1149.1-2001 (Revision of IEEE Std 1149.1-1990) Standard Test Access Port and Boundary-Scan Architecture*, IEEE Computer Society (Test Technology Standards Committee), 25 October 2001.

[7] *Serial Vector Format (SVF)*, available on-line at http://www.asset-intertech.com/support/svf.html (visited on March 5th 2010).

[8] *NI ELVIS: Educational Design and Prototyping Platform*, available on-line at http://www.ni.com/nielvis/ (visited on March 5th 2010).

[9] *Adobe Acrobat Connect Pro: Web Conferencing, Web Conference, Online Meetings*, available on-line at http://www.adobe.com/products/acrobatconnectpro/ (visited on March 12th 2010).

AUTHORS

José M. M. Ferreira is with FEUP, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal, and HIBU, P.B. 235, 3603 Kongsberg, Norway (jmf@fe.up.pt).

Américo F. S. Dias, is with FEUP and INESC, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal (americo.dias@fe.up.pt).

Paulo J. S. Sousa is with FEUP, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal (sousa@fe.up.pt).

Zorica Nedic, is with the University of South Australia, Mawson Lakes Boulevard, Mawson Lakes, South Australia 5095 (Zorica.Nedic@unisa.edu.au).

Jan Machotka is with the University of South Australia, Mawson Lakes Boulevard, Mawson Lakes, South Australia 5095 (Jan.Machotka@unisa.edu.au).

Ozdemir Gol, is with the University of South Australia, Mawson Lakes Boulevard, Mawson Lakes, South Australia 5095 Ozdemir.Gol@unisa.edu.au).

Andrew Nafalski, is with the University of South Australia, Mawson Lakes Boulevard, Mawson Lakes, South Australia 5095 (Andrzej.Nafalski@unisa.edu.au).