THE VIRTUAL LABORATORY PROJECT

Faculty of Electrical Engineering and Computing, University of Zagreb
Unska 3, 10000 Zagreb, Croatia
Goran Mužak, Igor Čavrak, Mario Žagar
goran.muzak@fer.hr, igor.cavrak@fer.hr, mario.zagar@fer.hr

Abstract: This paper describes the idea and implementation of the virtual laboratory. The lab supports faculty courses involved in embedded systems, development of control algorithms, distributed systems and open computing systems. Short description of VL (virtual laboratory) environment is given as well as its hardware and software components description. Comparison with classic laboratory environments is given. Ways of the system deployment in training process are described. Further development aims and enhancement possibilities are given.

Keywords: virtual, laboratory, network, CAN, CORBA, embedded, remote access

Motivation

Basic motivation for development and implementation of virtual laboratory environment is raising quality of laboratory exercises for different courses related to area of computer systems and processes. Laboratory exercises which support course "Digital Computers and Processes", provide students with basic knowledge on embedded systems programming including direct usage of on-board I/O devices for monitoring and controlling of physical processes. Siemens C16x single-chip microcomputer family is taken as a basic platform. For training purposes, limited simulation of C167[1] microcomputer is developed on the ATLAS simulation system. ATLAS is powerful software tool, used for target digital system simulation, using the system's logical level description. Problems arise during various I/O (input/output) devices simulation, because ATLAS is not suitable for that purpose. Two years of laboratory exercises performing practice show that students often do not have feeling about "real-world" problems. ATLAS uses simulated processor to control simulated process in environment that is not completely suitable for this purpose. Better solution is laboratory environment in which student has direct access to real hardware (e.g. C167 development board) attached to the workstation with development system. Real physical process (e.g. DC motor drive model) is under control of development board. Even more natural, such environment is too expensive and too hard to maintain. Fact that more than 120 students have to perform 8 different exercises within three months has to be taken into consideration. This means that a large number of expensive development systems should be deployed. The problem of performing these exercises involving distributed control and measurement systems remains unsolved.

Virtual laboratory is built by taking good characteristics of approaches described above, employing modern networking and object-oriented technologies. It is a set of hardware and software components, placed around the Ethernet and CAN[2] [1] networks. Special software modules allow cooperation of components on these, conceptually different, networks.

Basic Concept

To access the VL, user[3] uses graphical front-end software written in Java. Multi-function client applet is downloaded from WWW server. It communicates with a classical monitor program that runs on target the development kit, which is placed somewhere on the CAN network. Monitor program has capability to communicate with client application over the combination of CAN and Ethernet networks.

---

[1] C16x family member used in this case.
[2] Controller Area Network is high-speed serial bus, designed for industrial usage in real-time systems. It is developed in late 80's, primarily for automotive industry. Today is widely used in industrial systems.
[3] Talking about VL, term «user» primarily addresses students who use the system.

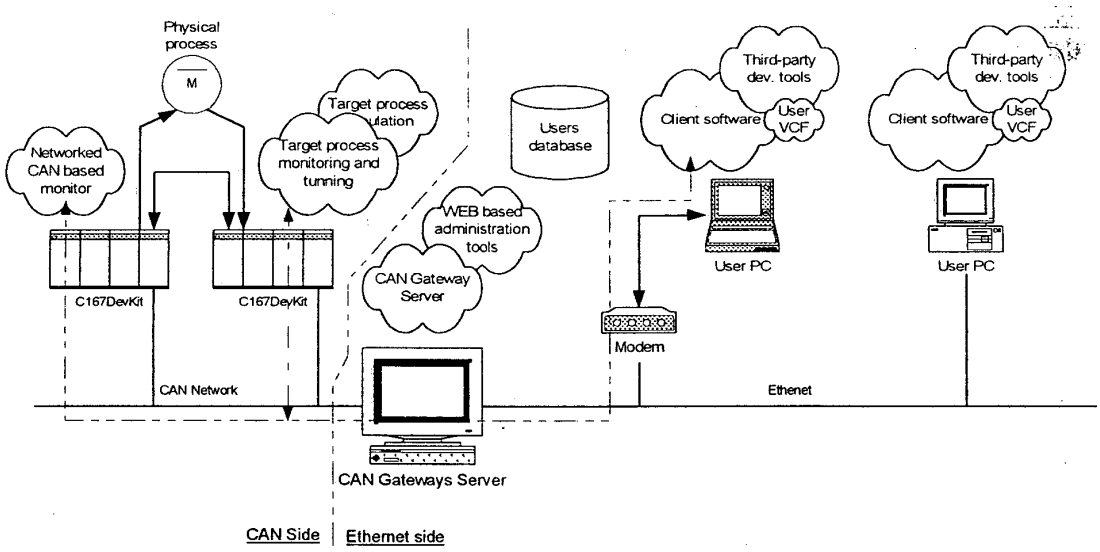

Fig. 1. The virtual laboratory structure overview.

The CAN Gateway Server comprehends significant differences between the CAN and the Ethernet networks. Communication mechanism is absolutely transparent for clients on Ethernet and servers on the CAN side. Gateway Server exports a set of CORBA [2] interfaces used by the client software to communicate with the CAN network functions. Interfaces are hierarchically organized allowing different ways of client-type dependant server access.

Another term introduced is VCF (Virtual CAN Function[4]). It is a process on a machine connected to the Ethernet network that simulates CAN device. Basically, all clients attached to the CAN Gateway Server are build as VCFs. Through CAN Gateway Server VCF can mutually interact with all other physical CAN devices and other VCFs. VCFs are suitable for building simulations of different CAN devices. Students can build their own VCFs. Because of some crucial differences between Ethernet network on that VCFs reside, and CAN network, which is VCF target, following issues should be specially considered.

Real-time Operations

CAN is real-time based network. Packages are very short and arbitration method allows, for most important messages, to be delivered to the target node in guaranteed time. In general, Ethernet has no such capability. If relatively large VCF response time is acceptable, VCFs could be very usable.

CAN Messages Dispatching

Connections to Ethernet network are established between two points while CAN nodes broadcast messages that can be received by all interested nodes (including VCFs). It means that CAN Gateway Server should broadcast received CAN messages to all VCFs. That would cause significant overhead. In most cases, VCF is not interested for message at all. To adopt those two philosophies, special mechanism is used. Each VCF makes a subscription on a CAN Gateway Server for particular CAN message types. In that way, message filtering centralization and network traffic reduction is achieved. Subscriptions can be changed at runtime.

[4] Term «function» is used in this context rather than «device», because it is assumed several functions can be implemented within single device.

Code Development Process

User develops assembler code using external development application, which produces binary file ready for testing (Fig. 2). It is possible to download developed code to the target development kit (using FTP/CAN[5] protocol) and to interact with monitor program at the target system (using Term/CAN[6] protocol) by means of interchanging human-legible messages. The monitor supports more or less standard set of debugging commands for code execution control, breakpoints management, variables and registers viewing (such as "go", "print", "break", "cont"...). Special graphic environment is developed at the client side as a wrapper around symbolic monitor control commands. In combination with source code available at the client side, very efficient yet simple to build debugging environment can be implemented.

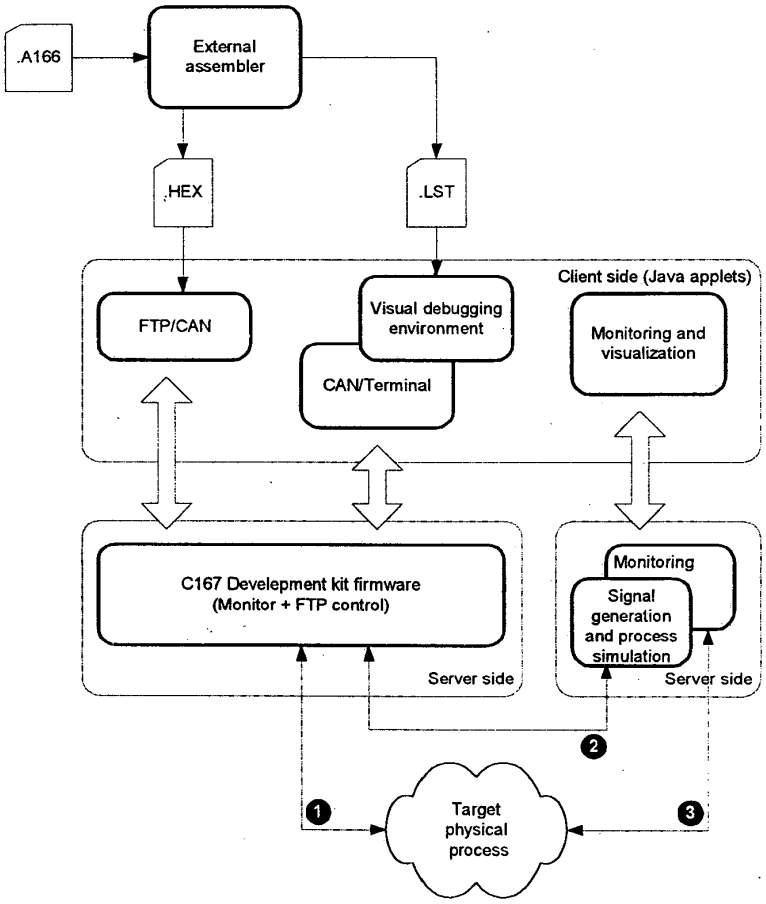

Fig. 2. Target code developing process overview.

Interaction with Controlled Processes

User developed program code interacts with external processes through I/O components build on the development kit (Fig. 3). It interacts with physical processes (e.g. model of regulation DC motor drive loop) or with processes simulated on another development kit. In both cases interaction is performed through real I/O devices that is of major importance. Another interesting possibility is that simulation

[5] Custom protocol used for file transfer over CAN.
[6] Custom protocol for terminal communication over CAN.

runs on a PC machine and uses I/O capabilities of one of development kits to generate process I/O signals. In all cases monitoring and visualization tool attached to the simulated process is available. In that way student has direct visual contact with the process, controlled by the code.

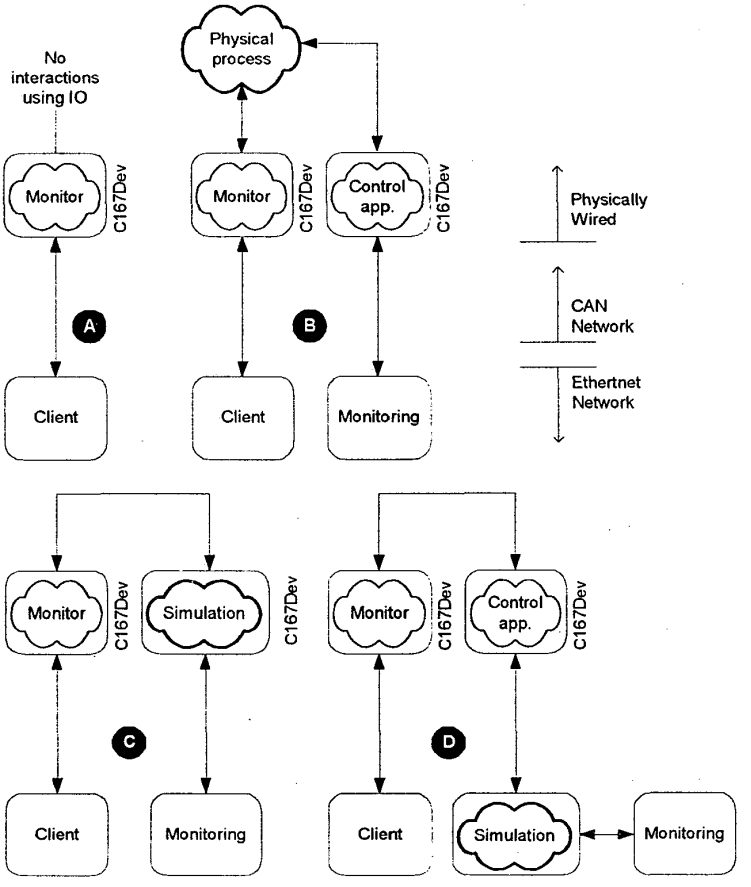

Fig. 3. Interaction of developed code with target physical process or its simulation.

Administration issues

Student reserves time in the VL using administration utilities. These are accessible as CGIs (common gateway interface) through the WWW server [4,5]. Student has to allocate resources necessary for target exercise, due to its status and set of related parameters. Resource allocation is complicated procedure because of mutual interdependencies between different VL components. Of course, the end-user has a simple Web based interface used for reservation procedure. When allocating VL resources, problem that has to be solved is presented to the student. It is possible to customize exercises for each student separately[7]. When task is finished student submits results in appropriate way to the system in order to perform automatic results validation, when it is possible. Exercise performing, from student's viewpoint, looks as follows:

1. In the beginning of semester set of exercises, which must be finished during the specified period, are given to a student. These exercises cover different areas of course material. Certain freedom can be given to the student to choose order and time when he/she will work on the task.

[7] That means a selection of random exercise from the database, for the specific area, as well as customization of target system parameters.

2. Using a simple Web based interface, student reserves time to access VL. In this time all needed VL resources are reserved. A detailed exercise specification is generated after registration as well as passwords needed to access the VL.
3. During the reserved time period student can access the VL through the downloaded graphical interface.
4. Student has reserved time to finish the exercise
5. When finished, student has to submit results via email. Results are formatted in the report file automatically generated by its client software. The report file contains:
   - Program code;
   - Exercise parameters setting, when customized;
   - Results read directly from the system simulation.

Software Implementation Technologies

There are several VL subsystems, each employing technique which satisfies specific needs:

- ***Server software*** – written in C++, running on Windows NT/2000 or Linux platforms. Related services as well as set of virtual CAN functions are, also, written in C++. CORBA [2] technology is used for communication with clients on the network.
- ***Client software*** – written in Java [2,3], as applets, which can be downloaded to target user machine from WWW server. Different clients can be built, depending on specific type of exercise. In first versions of the system, external assemblers will be used.
- ***Development kit firmware*** – these software parts represent real servers in VL environment. Client software uses CAN Gateway Server and related services as a bridge to access the networked monitor programs on development boards. The development board software is written in C/C++ language.
- ***Administration utilities set*** – CGI programs accessible through the Web pages.

Hardware Base

The virtual laboratory employs a set of custom-build development kits based on Siemens C167CR-L25 single chip microcomputer at clock frequency of 25[MHz]. That is inexpensive component, with wide range of on-board I/O devices. It is based on RISC like 16-bit processing unit as its core. Each development board has up to 512[kB] of FLASH memory and up to 256[kB] of SRAM. One of the key features is CAN interfacing capability. CAN is a fast, real-time, serial bus oriented toward industrial and process control applications. Because of its characteristics, CAN perfectly fits into this system. The CAN bus is used for development board interconnection. Host machine for CAN Gateway Server, must be equipped with CAN controller card[8], as well as Ethernet card, to provide access to both networks. To run the client software, one should have Java enabled machine with access to the network, hard-wired or modem linked. In other words, the VL system does not require any special hardware, with exception of development kits and appropriate PC CAN controller.

[8] Custom designed CAN controller ISA card, based on Intel 82527 CAN controller, is designed and build to meet requirements for VL usage.

Comparison of Classic and VL Environment

Following table gives comprehensive view of VL environment properties in comparison with classic laboratory environment, in which students work on target system simulation or directly accessing the development kit.

Table 1. Comparison of virtual laboratory environment and classic laboratory.

Improvements	Difficulties
• Exercises can be customized for each student • Quality of results validation can be significantly improved and automatically performed in a certain cases. • More efficient usage of laboratory resources. Student can access laboratory from anywhere and at any time when needed resources are free. • Open system architecture gives a freedom to develop and integrate new parts (software and hardware) of the system easily. • Higher level of reality that in simulation. Even the access to the laboratory environment is virtual, target development system and attached devices exist physically. • High system scalability. Number of development kits can be easily increased depending on expected usage. • Easy access to the VL using simple and intuitive graphical interfaces written as Java applets. • Efficient access control. System parts can be exposed for a public usage.	• There are no direct help and suggestions that could be given by assistant that is in direct contact with students. Partial solution is online help system, but it is obvious that significant part of the training must be done in a classic way.

Conclusion

First practical experience with VL system is expected during test period on a system with up to 4 networked development kits, and appropriate software and hardware infrastructure support. In the first phase limited number of exercises will be performed in VL because higher resources are needed for more than 500 hours of active usage per semester. Networking and distribution technologies used for the implementation offer relatively simple and efficient solutions for most parts of the code. One of the most problematic parts is development kit firmware. It must be able to perform fast recover from fatal errors that may occur in developed code, without significant influence on the rest of the related components. Important direction of improvements is usage of the VL system in other courses and updates of VL components depending on specific requirements of those courses.

References

[1] Schofield, M.J. (2000), *Controller Area Network (CAN) on a Device Level*, John Wiley and Sons
[2] Henning M., Vinoski S. (1999), *Advanced CORBA Programming with C++*, Addison-Wesley
[3] Harkey D., Orfali R. (1998), *Client/server Programming with Java and CORBA, Second Edition*, John Wiley and Sons
[4] Felton M. (1997), *CGI: Internet Programming in C++ and C*, Prentice Hall
[5] Hunter J., Crawford W., Ferguson P. (1998), *Java Servlet Programming*, O'Reilly & Associates