TOOL-ORIENTED LEARNING MODULS AND LABORATORY FOR TEACHING VIA INTERNET

Heinz-Dietrich Wuttke[1], *Karsten Henke*[2]

Abstract — *Referring to the topic "Interactive web-based learning material" in our paper we would like to present a learning material for teaching digital design that supports several phases of the learning process including self- study as well as laboratory and is available via the Internet.*
This contribution introduces a concept of a Web based laboratory, the ”FIPS”-project, offering several experiments in controlling, observing and analysing of real objects via the Internet. All single steps of the laboratory process - starting with the design of control algorithms up to the web-based validation - will be presented on the basis of an existing laboratory experiment. Derived from that, the required system architecture of the project “FIPS”' will be shown and different kinds of realizations will be compared regarding the execution of the designed control algorithm.

Index Terms — *Web based laboratory, remote control via Internet, distance learning, Internet based education and training.*

INTRODUCTION

Teaching technical basics in computer science typically requires explaining complex interdependence and dynamic processes. Design methods as well as methods to verify the design have to be taught in a way that students can understand the algorithms behind and use them during a design process.

Computer animation can help to get a deeper understanding of a dynamic process or a complex interdependence during a lecture. A dynamic process has to be explained at first step by step, but later its dynamic behaviour can be demonstrated using interactive simulations. Parts of a complex interdependence have to be explained separately from different views, but later their whole complexity can be demonstrated. Traditional teaching methods using desk, overhead or "PowerPoint™"-presentations can explain such a process or interdependence only partially.

The primary problem in describing such processes is that after the lecture, the dynamic changes get lost and the students only retain the static part of the whole scenario in their memories. A correction of this situation can be reached if the whole material, containing all scenes of the dynamic process, can be accessed via the Internet. The problem, left here, is that students have no possibility for an active contention with the materials. So the motivation to work with the "lifeless" material is not very high.

To solve this problem, we create and use interactive teaching modules that can be accessed via the Internet and used in a lecture as well as for individual self-studies, independent of time and place. We call them "Living Pictures". We developed an Internet based teaching method, which supports various phases of the learning process. Finally, after listening to the lessons and using these learning modules, the students are able to design small control algorithms. To proof the solutions of the design in a real environment, we have realized a Web based laboratory in the project FIPS. This contribution also introduces the concept of the “*Web based laboratory*” with several experiments offering controlling, observing and analysing features via the Internet.

Goal of the project “FIPS” is to show new ways and chances of remote controlling and remote observation of real processes. The abbreviation “FIPS” stands for **F**erngesteuertes **I**nternet **P**raktikum **S**chaltsysteme - in English: remote controlled laboratory for the course “Digital Logic Design”. In this laboratory the students get a first impression of this technology e.g. in the fields of control engineering, robotics and tale-control engineering, dealing with integrated and interactive usage of modern Internet and intranet technologies.

All single steps of the laboratory process - starting with the design of control algorithms up to the Web based validation - will be presented on the basis of the existing laboratory experiment.

SUPPORT BY “LIVING PICTURES”

At first we want to explain the concept of “Living Pictures”. Main idea to realize “Living Pictures” is to find a summarizing overview of a learning subject, which can be abstracted in a graphical manner, in some books called “The Big Picture”. Mostly such a picture is placed at the end of a section. This picture is what we directly use as graphical user interface of an applet. The applet offers the whole functionality of what the graphic tries to explain. We define the concept as follows:

Given a tricky, quite complicated situation of the learning subject in a graphical form on the computer, the graphic has to be self-explanatory and must provide interaction possibilities. By using these possibilities the

[1] Heinz-Dietrich Wuttke, Technical University of Ilmenau, Faculty of Computer Science and Automation, D-98684 Ilmenau, wuttke@theoinf.tu-ilmenau.de
[2] Karsten Henke, Technical University of Ilmenau, Faculty of Computer Science and Automation, D-98684 Ilmenau, wuttke@theoinf.tu-ilmenau.de

students can generate examples that are interesting enough to encourage their own experiments but that are not too complicated for learning. They can produce input stimuli and watch the reactions. Analysing the inputs, a *simulation component* of the applet starts, which executes the method that has to be taught, and presents its results using a *visualization component*. Thus the students immediately get a correct reaction of their inputs. In that learning phase it is important that there is a simple interface and that no assessments of the students' inputs occurs. They can use “Living Pictures“ like a game: they act and the system reacts by using a yet unknown method (that one, that should be learned). Furthermore, if the students have some more knowledge, they can use them as *tools* because the simulation component solves a part of the real design process. In addition to the simulation component there is also an *explanation component*, describing the unknown method step by step, using the actual chosen or generated example.

Those applets open a wide field of possible use cases during the learning process. They can be used

- During a *lecture* to demonstrate some interesting examples including extreme situations.
- During the *self-study* to repeat the examples given in the lecture and to solve the homework.
- Before and during the *laboratory* as tools in the design process.

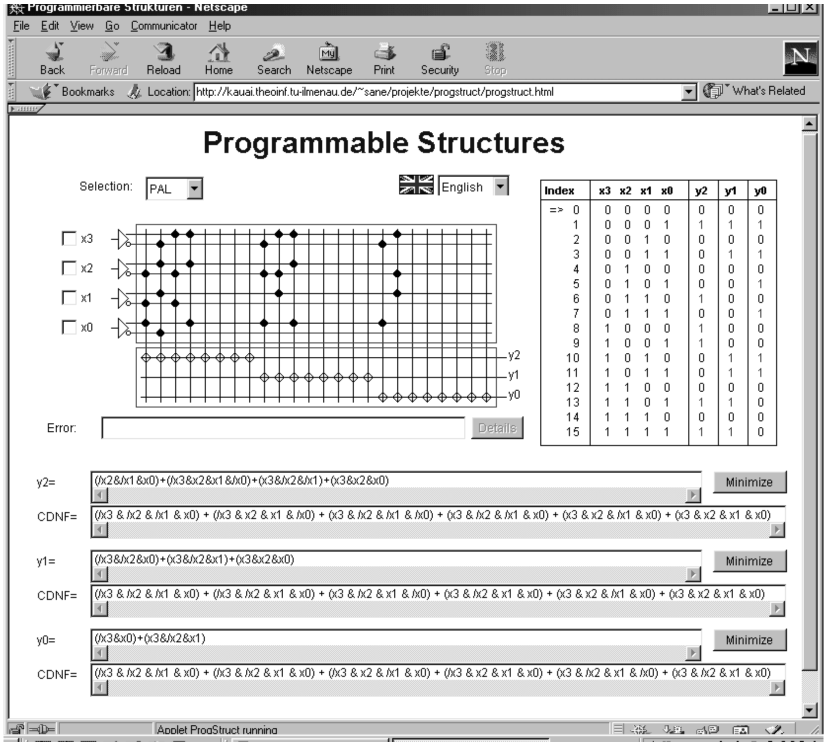
Actually we have developed a collection of about 10 applets and transferred the concept also to other teaching subjects. [Ubar, Fengler].

EXAMPLE OF A "LIVING PICTURE"

To explain the concept in detail we would like to present an example of a "Living Picture" dealing with programmable logic devices (PLD).
Figure 1 shows a screenshot of the applet. As described above, we use a summarizing graphical overview of a learning subject as user interface. Every part of the picture is interactive:

At the upper left part learners can select the programmable logic device. They can choice between a ROM, a PLA or a PAL realisation.

Figure 1 shows the programming of a PAL (Programmable Array Logic). Below the choice box programmable field(s) can be programmed interactively. Clicking a cross point in the first matrix installs a connection with a product term, clicking a cross point in the second matrix installs a connection with a sum- term. The programmed functions are simultaneously shown in the truth table at the right hand side and in the form of Boolean expressions in the edit lines below. To understand the functionality of the chosen PLD, it is possible to assign 1 or 0 values to the input variables x_0, x_1, x_2, and x_3 and watch the coloured, active elements in the circuit. Figure 2 shows the situation, where x_0=1, x_1=0, x_2=0, and x_3=1.


Programmierbare Strukturen - Netscape
File Edit View Go Communicator Help
Back
Forward
Reload
Home
Search
Netscape
Print
Security
Stop
Bookmarks
Location: http://kauai.theoinf.tu-ilmenau.de/~sane/projekte/progstruct/progstruct.html
What's Related
□ x3
□ x2
□ x1
□ x0
y2
y1
y0
Details
Minimize
Applet ProgStruct running

Programmable Structures

Selection: PAL

English

Index	x3	x2	x1	x0	y2	y1	y0
=> 0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1
2	0	0	1	0	0	0	0
3	0	0	1	1	0	1	1
4	0	1	0	0	0	0	0
5	0	1	0	1	0	0	1
6	0	1	1	0	1	0	0
7	0	1	1	1	0	0	1
8	1	0	0	0	1	0	0
9	1	0	0	1	1	0	0
10	1	0	1	0	0	1	1
11	1	0	1	1	0	1	1
12	1	1	0	0	0	0	0
13	1	1	0	1	1	1	0
14	1	1	1	0	0	0	0
15	1	1	1	1	1	1	0

Error:

y2= (/x2&/x1&x0)+(/x3&x2&x1&/x0)+(x3&/x2&/x1)+(x3&x2&x0)

CDNF= (/x3 & /x2 & /x1 & x0) + (/x3 & x2 & x1 & /x0) + (x3 & /x2 & /x1 & /x0) + (x3 & /x2 & /x1 & x0) + (x3 & x2 & /x1 & x0) + (x3 & x2 & x1 & x0)

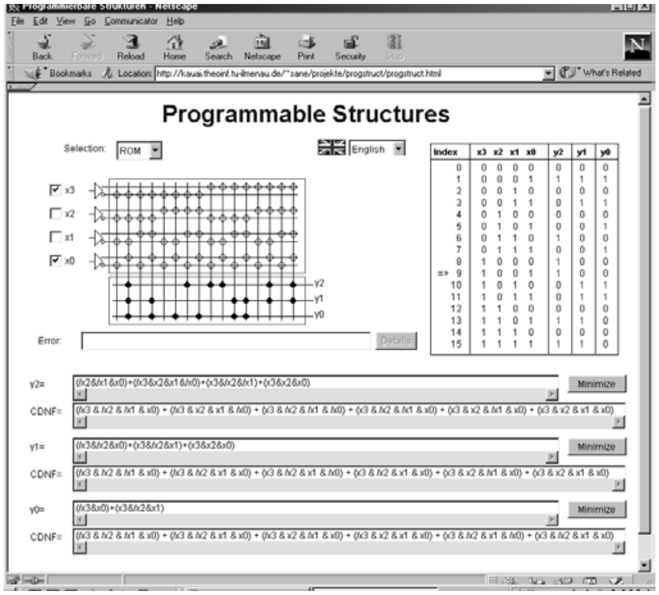
y1= (/x3&/x2&x0)+(x3&/x2&x1)+(x3&x2&x0)

CDNF= (/x3 & /x2 & /x1 & x0) + (/x3 & /x2 & x1 & x0) + (x3 & /x2 & x1 & /x0) + (x3 & /x2 & x1 & x0) + (x3 & x2 & /x1 & x0) + (x3 & x2 & x1 & x0)

y0= (/x3&x0)+(x3&/x2&x1)

CDNF= (/x3 & /x2 & /x1 & x0) + (/x3 & /x2 & x1 & x0) + (/x3 & x2 & /x1 & x0) + (/x3 & x2 & x1 & x0) + (x3 & /x2 & x1 & /x0) + (x3 & /x2 & x1 & x0)

FIGURE 1
EXAMPLE OF A LIVING PICTURE


Programmierbare Strukturen - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://kauai.theoinf.tu-ilmenau.de/~sane/projekte/progstruct/progstruct.html
What's Related
Minimize
Details
y2
y1
y0

Programmable Structures

Selection: ROM

English

☑ x3
□ x2
□ x1
☑ x0

Index	x3	x2	x1	x0	y2	y1	y0
0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1
2	0	0	1	0	0	0	0
3	0	0	1	1	0	1	1
4	0	1	0	0	0	0	0
5	0	1	0	1	0	0	1
6	0	1	1	0	1	0	0
7	0	1	1	1	0	0	1
8	1	0	0	0	1	0	0
=> 9	1	0	0	1	1	0	0
10	1	0	1	0	0	1	1
11	1	0	1	1	0	1	1
12	1	1	0	0	0	0	0
13	1	1	0	1	1	1	0
14	1	1	1	0	0	0	0
15	1	1	1	1	1	1	0

Error:

y2= (/x2&/x1&x0)+(/x3&x2&x1&/x0)+(x3&/x2&/x1)+(x3&x2&x0)

CDNF= (/x3 & /x2 & /x1 & x0) + (/x3 & x2 & x1 & /x0) + (x3 & /x2 & /x1 & /x0) + (x3 & /x2 & /x1 & x0) + (x3 & x2 & /x1 & x0) + (x3 & x2 & x1 & x0)

y1= (/x3&/x2&x0)+(x3&/x2&x1)+(x3&x2&x0)

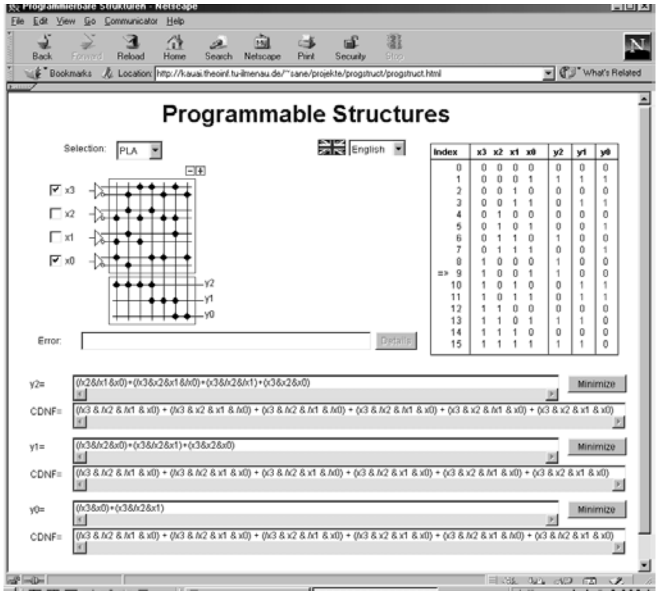
CDNF= (/x3 & /x2 & /x1 & x0) + (/x3 & /x2 & x1 & x0) + (x3 & /x2 & x1 & /x0) + (x3 & /x2 & x1 & x0) + (x3 & x2 & /x1 & x0) + (x3 & x2 & x1 & x0)

y0= (/x3&x0)+(x3&/x2&x1)

CDNF= (/x3 & /x2 & /x1 & x0) + (/x3 & /x2 & x1 & x0) + (/x3 & x2 & /x1 & x0) + (/x3 & x2 & x1 & x0) + (x3 & /x2 & x1 & /x0) + (x3 & /x2 & x1 & x0)

FIGURE 2
ROM REALISATION

Up to three functions can be programmed and edited in the input lines, the programming field or the truth table. It is also possible to watch the same functions and assigned values in the other PLDs. Thus, students can see different realisation principles and compare the effort of resources. Another possibility to compare different kinds of realisations is to watch the differences between a minimized and a canonical form of the function. This feature is shown in Figure 2 and Figure 3.


Programmierbare Strukturen - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://kauai.theoinf.tu-ilmenau.de/~sane/projekte/progstruct/progstruct.html
What's Related
Minimize
Details
y2
y1
y0

Programmable Structures

Selection: PLA

English

☑ x3
□ x2
□ x1
☑ x0

Index	x3	x2	x1	x0	y2	y1	y0
0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1
2	0	0	1	0	0	0	0
3	0	0	1	1	0	1	1
4	0	1	0	0	0	0	0
5	0	1	0	1	0	0	1
6	0	1	1	0	1	0	0
7	0	1	1	1	0	0	1
8	1	0	0	0	1	0	0
=> 9	1	0	0	1	1	0	0
10	1	0	1	0	0	1	1
11	1	0	1	1	0	1	1
12	1	1	0	0	0	0	0
13	1	1	0	1	1	1	0
14	1	1	1	0	0	0	0
15	1	1	1	1	1	1	0

Error:

y2= (/x2&/x1&x0)+(/x3&x2&x1&/x0)+(x3&/x2&/x1)+(x3&x2&x0)

CDNF= (/x3 & /x2 & /x1 & x0) + (/x3 & x2 & x1 & /x0) + (x3 & /x2 & /x1 & /x0) + (x3 & /x2 & /x1 & x0) + (x3 & x2 & /x1 & x0) + (x3 & x2 & x1 & x0)

y1= (/x3&/x2&x0)+(x3&/x2&x1)+(x3&x2&x0)

CDNF= (/x3 & /x2 & /x1 & x0) + (/x3 & /x2 & x1 & x0) + (x3 & /x2 & x1 & /x0) + (x3 & /x2 & x1 & x0) + (x3 & x2 & /x1 & x0) + (x3 & x2 & x1 & x0)

y0= (/x3&x0)+(x3&/x2&x1)

CDNF= (/x3 & /x2 & /x1 & x0) + (/x3 & /x2 & x1 & x0) + (/x3 & x2 & /x1 & x0) + (/x3 & x2 & x1 & x0) + (x3 & /x2 & x1 & /x0) + (x3 & /x2 & x1 & x0)

FIGURE 3
MINIMIZED FUNCTIONS IN A PLA

The described applet supports one step in the design process of a digital circuit, as it is required in a laboratory exercise. For preparing the solutions of laboratory tasks other applets support further design steps and are published in [1]- [3]. To proof the design on a real object, normally the students have to take part on a real laboratory, where they can build or program circuits to solve a control problem on a real moving object. The next paragraphs describe the concept of a Web-based laboratory, where students can run their solution on real objects via the Internet.

CONCEPT OF THE "WEB BASED LABORATORY"

The students' task during the laboratory is to develop and run an algorithm that controls a model of a simple plotter. They have access to four Boolean variables, indicating the final positions of the plotter in each direction (up, down, left, right) and four Boolean variables to control four engines, one for each direction. Additional there is a signal for starting and stopping the movement of the plotter. Thus the algorithm that has to be designed shall control the plotter to move on a defined way from a starting point to an end position. In the real laboratory they develop the algorithm, simulate it and program it into a PLD that controls the plotter. If the algorithm is wrong and the students did not find the error by simulation, they find it out on the real model. At that point they have to correct their algorithm as long as it works correctly.

The concept of the Web based laboratory is to give the students a possibility to test their algorithm remotely before they program a circuit in the real laboratory. First of all, the system architecture, as developed during the project "FIPS", will be introduced in the next paragraph. This architecture is required to realize *Web based laboratory* courses. Afterwards different kinds of implementation will be compared regarding the execution of the designed control algorithm.

System Architecture

The main concept is founded on client/server architecture because of the TCP/IP mechanisms used by the Internet. That is why it has to be examined which system components are required on client side (e.g. the students PC at home) and on server side (e.g. in the laboratory room) as well.

Client based Components

The Client of the system is usually an Internet browser. It has to offer a user interface providing possibilities for input the users' design, starting or stopping the designed control algorithm and interactive manipulation of environmental variables. FIGURE 4 shows a screenshot of the user interface on the students Browser at home.

In the upper left part of the screen the students can edit the Boolean function of their control algorithm. Below these input lines the I/O-monitor is located.

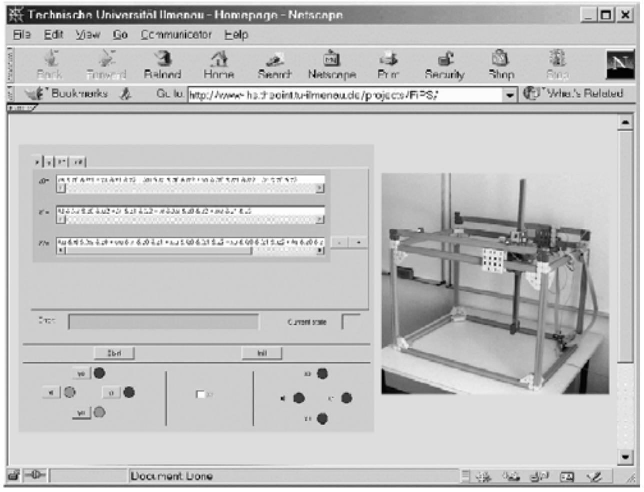

Technische Universität Ilmenau - Homepage - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Go to: http://www-hs.fhpoint.tu-ilmenau.de/projects/FIPS/
What's Related
Start Init
Document Done

FIGURE 4
STUDENTS' BROWSER AT HOME

The I/O-monitor allows an access to the input/ output variables of the process. At the left hand site the students can interact with the four engines by pressing the associated buttons to move the system in a starting position. Highlighted circles near the buttons display active engines. If the plotter reaches one of the four end positions (or two in case of a corner position), the corresponding input variables are also highlighted. The actual position additionally can be watched in the live picture of the laboratory at the right part of the screen.

Thus the input and handling functionality of the client applet offers:

- Input feature for the designed control algorithm
- Handling the laboratory procedure (e.g. starting or stopping the execution of the control algorithms),
- Features for interactive influencing of environ-mental variables.

The supervising and observing functionality, also realised in the client applet, supports:

- Supervising of actual model and environmental variables via I/O monitor,
- Observation of the laboratory procedure and environment via a Web-Cam.

In that way the students get all information they need for preparing and testing their designed control algorithm. To run the students' design and to control the plotter model this information will be transfered via the Internets' TCP/IP protokoll to the server in the laboratory, which have access to the real process.

Server based Components

In the laboratory room the components, showing in FIGURE 5, are necessary to guarantee the server functionality.

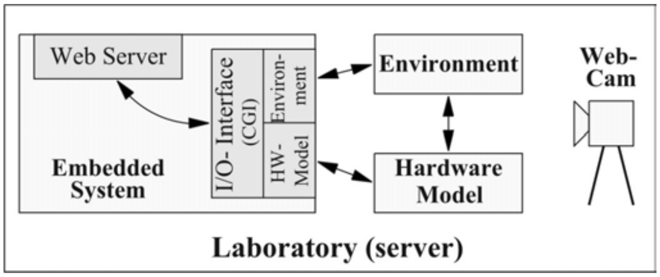


FIGURE 5
COMPONENTS IN THE LABORATORY ROOM

An *embedded system* solves the following tasks:

- realizing the communication with the client through the integrated Web server,
- Exchanging all available model and environmental variables through the I/O interface (using CGI scripts).

The *hardware model* (in the described example the plotter model) to be controlled by the students algorithm is also located in the laboratory room as well as a *Web Cam* offering the observation of the whole hardware set-up (model and surrounding).

Usage of the Web based embedded system simultaneously demonstrates the advantages of single chip PC for such ranges of application. It gives a cost efficient and space saving alternative to a complete PC.

For supervising the laboratory procedure the students can use two information paths:

- *Supervision*
  By supervising, the signals of the designed control (input/output variables) and its environment can be transferred to the client. These variables are interactively changeable.
- *Observation*
  Additional information (not available through the input/output interfaces) will be retrieved and transferred by a Web Cam to provide assertions in case of faulty behaviour.

Web supervising is not obligatorily required for fault free designed control algorithms and ideal environmental behaviour. Because in the project "FIPS" a selected or accidental manipulation of environmental variables is planned from the outset, an observation of the whole hardware set-up is required urgently. This is important e.g. to create different initial conditions or to preset a real malfunctioned condition.

By that almost real laboratory conditions are available at the students home via Internet and they can prepare their laboratory work completely at home.

In Figure 6 the main communication paths are shown:

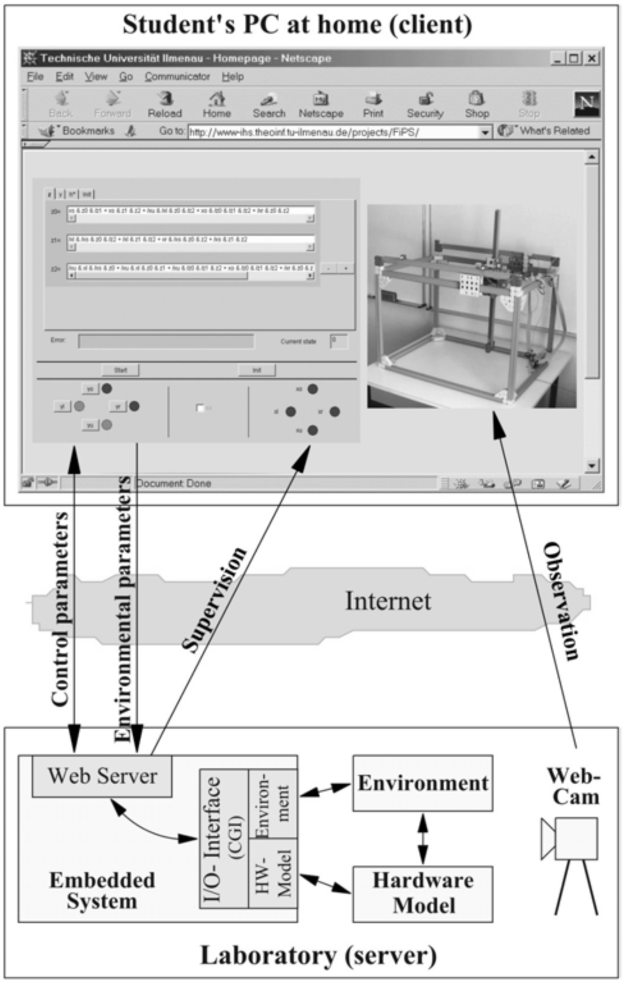

Student's PC at home (client)
Technische Universität Ilmenau - Homepage - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Go to: http://www-ihs.theoinf.tu-ilmenau.de/projects/FIPS/
What's Related
Error
Current state
Start
Init
Document: Done
Control parameters
Environmental parameters
Supervision
Internet
Observation
Web Server
I/O- Interface (CGI)
Environ-ment
HW-Model
Environment
Hardware Model
Web-Cam
Embedded System
Laboratory (server)

FIGURE 6
MAIN COMPONENTS OF THE “FIPS” PROJECT

Not all information can be provided in sufficient manner by the Web Cam (e.g. values of the limit switches or internal signal levels). Therefore it's necessary to supervise this information through an I/O monitor additionally. That way redundant information contributes to the improvement of the understanding.

Execution of the Control Algorithm

In the following it should be examined how the two requirements

- The client based “remote” control as well as
- The server based “on the spot” control

can be realized in the project “FIPS”.

In both cases the equations for model control, created by the students and entered in a browser entry window, are executed through an *interpreter*. In doing so further environmental variables can be influenced and supervised via client/server connectivity (using the environmental I/O interface of the server). It allows an observation of the hardware set-up through a Web Cam. It is not necessary to describe this part in detail, instead we place the variants of the “remote” or “on the spot” control in front.

Client based “Remote” Control

When using the client based remote control the hardware model will be controlled “from the far” through the interpreter running inside the client (implemented e.g. as Java applet). In this case only the input and output signals of the model will be transferred via Internet.

Permanent communication between the interpreter and the model I/O interface of the server occurs to exchange the required model variables. On one hand it is necessary for the I/O interface to query the model variables via a CGI script and to pass on the received information from the Web server to the interpreter using a plain text file or already as HTML code. On the other hand, when transferring data from the hardware model to the client, a CGI script will be used to pass on the results achieved by the interpreter via Web server to the I/O interface of the model..

In doing so it is inevitable to convert the existing serial or parallel information into TCP/IP standard packages. Because TCP/IP is not a bi-directional connection but based on requests, the client has to continually reinitiate relating requests in appropriate time intervals. Only this enables the client to register changes of the model or environmental variables. The chosen interval should be in a sensible area. To keep the network traffic low the time interval has to be selected not too small on one hand. On the other hand the interval must be chosen not too long to make sure that all critical messages reach the interpreter in time. Detailed examination results will be introduced in the presentation.

Executing the control algorithm client based leads to a smaller server load because any client executes its own (extensive) calculations. The only disadvantage is that the results of the client's calculations can get lost if the connection to the server is crashed in the meantime.

Server based “On The Spot” Control

When executing the control algorithm server based the required *z* and *y* equations have to be transferred from the browser via the Web server to the interpreter running inside (on the spot) the embedded system only one time. In this case the interpreter exchanges data with the model I/O interface directly.

One of the main tasks of the Web server is the administration of queries (from several clients usually). That is why it is necessary to estimate carefully which and how many extensive calculating tasks can be executed simultaneously without breakdowns in the different communication paths to the clients. According to this, detailed examinations are carried out presently.

In case of a crashed client/server communication the actually executed calculation can be finished. Using appropriate programming mechanisms ensures a definite reaction, for example a stop of further calculations afterwards.

Observation of the Hardware Set-up

It is required to refresh the offered image of the hardware set-up (model and environment) continuously to have actual information about the state of the model. To transfer this information a resolution of 320 x 240 pixels in a quality of about 30% was chosen in project "FIPS". This provides a good relation between image quality and transfer rate. Presently different image transfer variants are in the process of examination, e.g.

- CGI-Scripts,
- Real or QuickTime streaming,
- JPEG streaming.

Some results will be introduced in the presentation.

User Administration

The projects present state of realization allows access to the hardware model for only one student at a time (single user mode). Future states will offer cooperative work features for several students, for instance to manipulate the hardware model via different parallel automata of a distributed control simultaneously.

To provide a trouble free one-user access a kind of "locking" has to be implemented. It is also required to restrict the access time on the server. This has to be implemented to give as many as possible students the chance to work with the "*Web based laboratory*".

Regarding these problems the following mechanisms will be examined:

- Locking using a file,
- Locking using a background process,
- Interactive locking,
- Authentication.

First results and a comparison of these methods will also be offered in the presentation.

Conclusion

First experiences using the "*Web based laboratory*" in combination with the *"living pictures concept "* show that

- The students will learn about the possibilities and limits of remote control and observation via Internet on practical examples. (Operation of some industrial and research facilities would be impossible without remote control and service because of certain risks in their usage.)
- The "*Web based laboratory*" and the *"living pictures concept"* offer an instrument for preparation of laboratory work in a new way leading to more efficient practical training.
- The student executes his experiments (Web based and in the lab room) always in the same surrounding. This reduces lab times (about 30 %) for more complex on the spot experiments with tutorial support and increases the efficiency of laboratory resources.
- The student - beside knowledge consolidation by practical experiments using these new technologies - is forced to estimate the technologies critically.

References

[1] Henke, K.; Peukert, R.; Wuttke, H.-D. (1999): Internet based education – an experimental environment for various educational purposes. In: *IASTED International Conference Computers and Advanced Technology in Education CATE1999.* Philadelphia, PA USA, 6.-8. May 1999: Proceedings, pp. 128-132.

[2] H.-D. Wuttke, S. Schneider, Supporting Interactive Web Based Learning Modules by Combined Speech-Text Elements, *Proceedings of the IASTED International Conference on Computers and Advanced Technology in Education*, Banff, Canada, 2001,pp. 65-70

[3] Ubar R., Wuttke H.- D.: The DILDIS-Project – Using Applets for More Demonstrative Lectures in Digital Systems Design and Test. *Proceedings of the 31st ASEE/IEEE Frontiers in Education Conference, FIE'2001*, Oct. 10-13, 2001, Reno, NV, USA, pp.SIE-2-7.

[4] Beck (2002): *Beck-IPC Nutzerdokumentation.* Beck-IPC GmbH: http://www.beck-ipc.com/chip.

[5] De Souza, C.P.; Filho, J.T.C (2001): Labnet – Laboratory of remote Access. In: *IASTED International Conference Computers and Advanced Technology in Education CATE2001.* Banff, Canada, *27*-29. June 2001: proceedings, pp. 38-40.

[6] DynaMit (2001): *Erlernen von Eigenschaften dynamischer Systeme mit Multimedia Mitteln.* Ruhr-Universität Bochum: http://astwww.chemietechnik.uni-dortmund.de/~mume.

[7] FIPS (2002): *Ferngesteuertes Internetpraktikum Schaltsysteme.* Technische Universität Ilmenau: http://www-ihs.theoinf.tu-ilmenau.de/projekte/fips.

[8] Goshi, K.; Matsunaga, K.; Kuroki, D. (2001): Educational Intelligent Transport System Assist. In: *IASTED International Conference Computers* and *Advanced Technology in Education CATE2001:* Banff, Canada, *27*-29. June 2001: proceedings, pp.150-154.