A low cost, scalable, virtual control laboratory

Inderpreet Arora, Kannan M. Moudgalya, Kaushik Venkata, Victor Chakraborty, Rupak Rokade and Rakhi R.

***Abstract*— This work presents a virtual control laboratory, based on a single board heater system and free and open source software, such as Scilab, Xcos, Java, PHP/MySQL and GNU/Linux. This solution allows of the order of 20 devices to be handled by a single Server PC. It allows many students to carry out control experiments remotely. It has been extensively tested in two control courses at IIT Bombay. Results of a few experiments, conducted remotely, using this setup are presented.**

I. Introduction

The objective of this work is to provide control laboratory education to students who may not have access to good infrastructure. This can be further detailed into the following specific objectives:

1) Provide a low cost laboratory. One should be able to create, operate and maintain the laboratory at a low cost. This requirement can be further divided into two sub-requirements:
   a) Low cost hardware
   b) Free software
2) Provide remote access so that only Internet access is required at the student end.
3) Provide good learning experience: It is not enough if the students enter parameters of a control algorithm, for example, P, I, D parameters; they should be able to develop and validate their own control algorithm.
4) Scale the solution so that a large number of students can access the laboratory facility simultaneously. This is important in the Indian context that has about a million students enrolling in engineering studies every year.
5) Access to a large number of students should be made with minimal infrastructure. For example, the computer server and the Internet Protocol (IP) address should be able to accommodate many apparatus simultaneously. This will allow minimally endowed Centres also to replicate such a facility and thus help evolve a distributed solution.

The issues raised in points 1(a) and 2 have been addressed in [1], [2] through a single board heater system (SBHS). In the current work, we address all the remaining issues. We begin with a brief literature survey.

Inderpreet Arora is a former student of Systems and Control Engineering, IIT Bombay, Mumbai, India. She is currently with the Eaton India Engineering Center, Eaton Corporation, Pune, India. inderpreet.arora@iitb.ac.in

Kannan M. Moudgalya is a Core Faculty member with Department of Chemical Engineering and an Associate Faculty member with Systems and Control Engineering, IIT Bombay, Mumbai, India. kannan@iitb.ac.in

Kaushik Venkata is a student of Systems and Control Engineering, IIT Bombay, Mumbai, India. kaushik8989@iitb.ac.in

Victor Chakrabory is a student of Department of Computer Science Engineering, IIT Bombay, Mumbai, India. victor@cse.iitb.ac.in

Rupak Rokade and Rakhi R. are Research Associates with Department of Chemical Engineering, IIT Bombay, Mumbai, India. rupakrokade@iitb.ac.in, rakhi@iitb.ac.in

II. Literature Survey

We now present a brief, non-exhaustive, survey of virtual laboratory work reported in the last couple of years.

The work reported in [3] allows multiple users the access to experimental and computational resources through only a Web browser, a low-bandwidth Internet connection, and low user-side technical specifications. The server side is scalable and reproducible since it is based on Linux operating system. This solution makes use of a Matlab license.

The work reported in [4] can be used as a portable telelaboratory, completely contained inside a Live Linux-RTAI DVD, which can be connected to a large number of real plant to perform a process supervision task. This lab architecture makes use of RTAI-Linux and COMEDI and alogorithms could be implemented in Matlab or Scilab.

The work reported in [5] allows the student to connect components and equipment such as capacitors, resistors, transistors, function generators with a switch system of a lab server and to map it to a configuration data structure. At run time, the GUI gets constructed automatically, using Adobe Flash CS3.

The work reported in [6] is based on a PLC and ATmega8 microcontroller for data acquisition and control. The PLC is programmed using OMRON CX programmer 5.0 and the microcontroller is programmed through AVR Studio 4.0. The software component is based on Java and MySQL. Live video is a part of this solution.

The work reported in [7] provide remote access to a thermal control plant and a velocity control plant. The former is built with off-the-shelf components (some transistors and temperature sensors), whereas the latter uses standard LEGO elements and minimal electronics. This work makes use of LabVIEW.

III. Single Board Heater System (SBHS)

A single-board heater system is a low cost, open source, lab-in-a-box setup [1]. It consists of a heater assembly, fan,

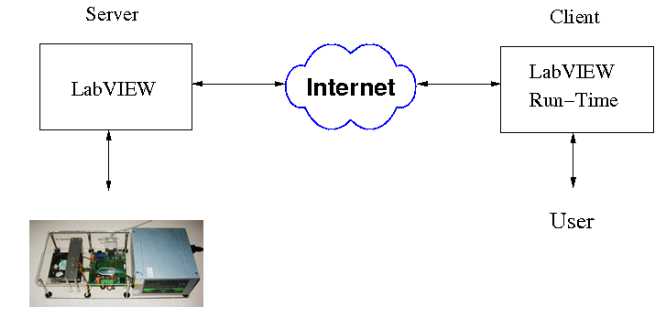

Fig. 1. SBHS virtual laboratory with remote access using LabVIEW

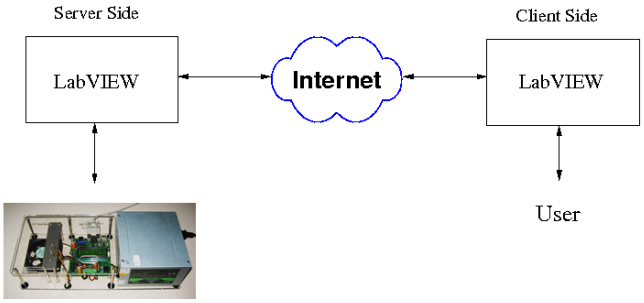

Fig. 2. SBHS virtual laboratory with remote access and live data sharing using LabVIEW

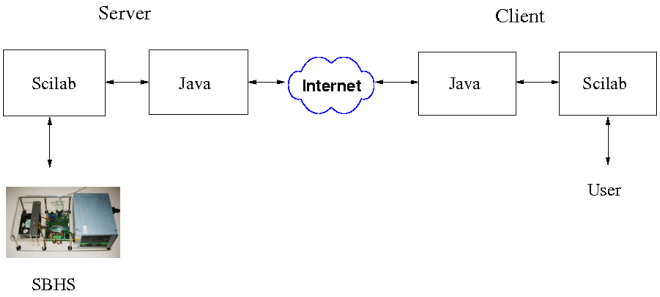

Fig. 3. SBHS virtual laboratory using open source software

temperature sensor, ATmega16 microcontroller and associated circuitry. A stainless steel blade whose temperature has to be controlled serves as the plant. Nichrome helical coil with 20 turns kept at a small distance from the steel blade, acts as the heater element. AD590, a monolithic integrated circuit temperature transducer, is soldered beneath the steel plate. A computer fan, a low cost and commercially off the shelf component, is used to cool the plate from below.

The plant has a small time constant, less than a minute, that allows completion of an experiment in a short time. This in turn facilitates performance of a large number of experiments in a single laboratory session. The speed of response not being too fast allows the measurements to be seen with naked eye, as it happens in industrial systems. It also demonstrates other measurement issues, such as, noise.

The codes for hardware interface and control experiments, and manuals are available at [8], [9]. Unlike [1], for the purpose of remote access, 252 until 255 are reserved as command words where 252 is meant for communication of machine I.D. This allows fan input to vary from 0 to 251 PWM (Pulse Width Modulation) units.

IV. Evolution of SBHS virtual labs

In [2], the control algorithm is implemented at the server end and the remote student just keys in the parameters, as shown in Figure 1. LabVIEW was used for the implementation of the same. The server end consisted of a computer connected with an SBHS with a full blown copy of LabVIEW installed on it. The client has a LabVIEW run time engine available for free download from the National Instruments website. A few LabVIEW algorithms/experiments were hosted on the server. The client accesses these algorithm/experiment over the Internet using a web browser by entering appropriate parameters.

It was realized that the learning experience is not complete for this structure. This is because the server hosts some prebuilt LabVIEW algorithms and a user can only access these few algorithms. The user can in no way change the program and can only input experimental parameters. Hence, we came up with a new architecture as shown in the Figure 2 that used full blown copies of LabVIEW at both server and client ends.

This idea uses the DataSocket technology of LabVIEW. Since now the client is having a complete LabVIEW installation on his/her computer she can now implement her own algorithms. Thus this architecture did provide a complete learning experience to the students. There are some shortcomings as well:

- LabVIEW is expensive and students may not be able to afford to buy it. It is also prohibitively expensive for the Government to distribute it.
- We used the LabVIEW version 8.04, which had restricted scripting language. It was tedious to create new control algorithms in it.

This made us shift to free and open source (FOSS) software. We replaced LabVIEW with Java and Scilab as shown in Figure 3. Scilab at the server end is used for communicating with SBHS. Scilab at the client end is used for implementing the algorithms. Java is used at both the server as well as client end for communication over the Internet thereby connecting the client with the server.

For the above solution, we need a dedicated copy of scilab running at the server end for every SBHS. One way to do this is to host it on multiple computers with unique IPs. Hence the number of SBHS we want to host requires as many computer's and public IPs thereby making it expensive. Moreover, it also limits its scalability. The other way to do this is to host multiple java and scilab servers on the same computer. Hosting many copies of Scilab simultaneously requires a powerful computer for the server.

For these reasons we decided to take scilab off the server computer and to use java alone to communicate with

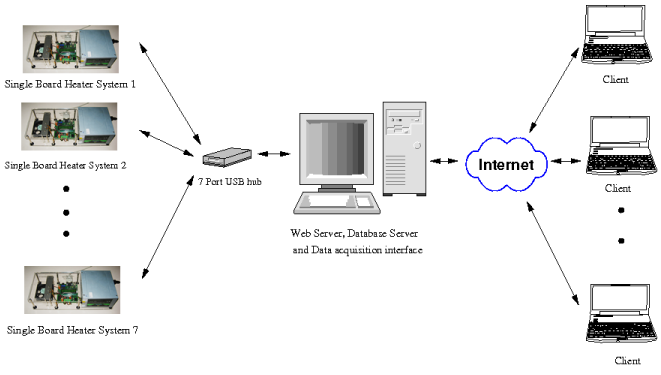

Fig. 4. Virtual control lab hardware architecture

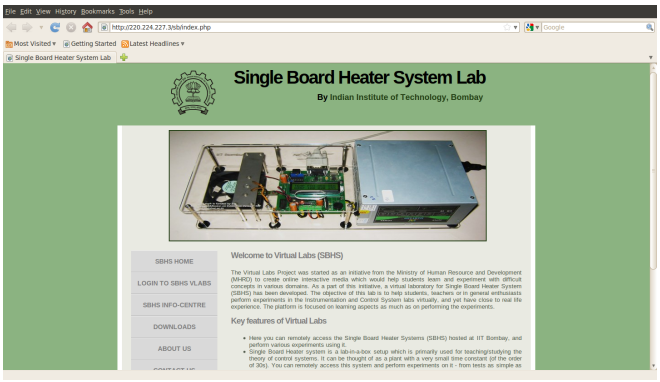

Fig. 5. Home page of SBHS V Labs

the SBHS directly. Java also communicates with the client computer. We connected seven SBHS systems to a USB port through a serial port hub. This architecture was implemented on a Windows Operating System. We faced the following difficulties in this solution.

- When we connected more than one serial hub to a PC, the port ID could not be retrieved correctly. Port ID information is required if we want a student to use the same SBHS for all their experiments during different sessions.
- The experiments required time stamping of the data communicated to and from the server. But this time stamping was not linear and suffered instability.

This made us to completely switch to FOSS with Ubuntu Linux as the OS and is the current structure of the Virtual lab as shown in Figure 6

V. Current Architecture

A. Hardware

The architecture of the virtual single-board heater system lab as shown in Figure 4 involves 7 single-board heater systems connected to the server via a 7-port USB hub. The server computer is connected to a high speed internetwork and has enough processing capability to host data acquisition, database, and web servers. The internetwork connected client computer needs only the Java runtime engine and Scilab installation.

A similar architecture but with 14 units connected to the server via two 7-port hubs has been successfully tested on intranet for the undergraduate Process Control course and the graduate Digital Control and Embedded systems courses conducted at IIT Bombay. Currently, the intranet and internet versions of the lab, connected to 14 and 7 units respectively, are available 24x7 for experimentation. One unit each from both the servers is integrated with a camera to facilitate live video streaming. This facility will be extended to all the units in future. This gives the user a feel of remote hands-on.

B. Software

The current software architecture of this virtual control lab is shown in Figure 6. The server computer runs on

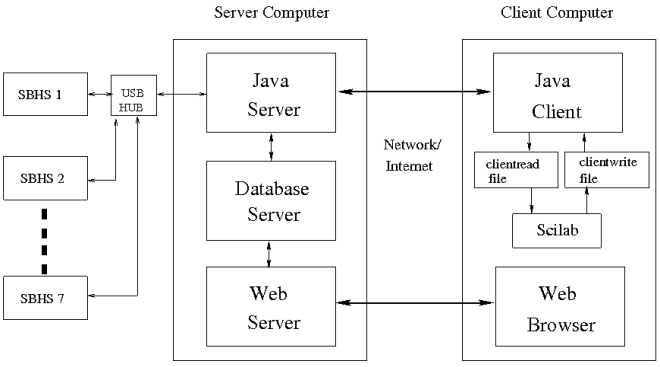

Fig. 6. Current Architecture of SBHS Virtual Labs

Ubuntu Linux 10.04 OS. It hosts a LAMP (Linux-Apache-MySQL-PHP) server. The MySQL-based database server has the details of all the registered users, their slot details, authentication keys to allow remote access, etc. It also hosts a PHP based web server shown in Figure 5 that has pages for registration, login and slot booking [8]. The server computer has Java based data acquisition interface and Java Remote Method Invocation (RMI) API for live data sharing. It takes care of serial port communication with the control setup, and live-data transfer across the networked client computers. The Java server also uses a v4l4j (Video4Linux4Java) package for capturing video streams from the camera connected to the SBHS and for sending it to the client. The client end has control algorithms running in Scilab and communicates over the Internet through the client Java RMI.

The steps to be performed before and during each experiment are explained next.

1) Registration: A client willing to perform the experiments needs to register with us by entering the personal details on the Registration page. This sends an account activation link to the clients mailbox. Upon clicking the link, the account gets activated.

2) Slot booking: The client can now login and book slots to perform the experiments. Each slot lasts for an hour with 55 minutes for experimentation and 5 mins for resetting the setup. A client can book up to 2 slots, per day, in advance. Besides this, if the current slot is empty, it can be booked as a free slot. For each slot being booked, the client gets a

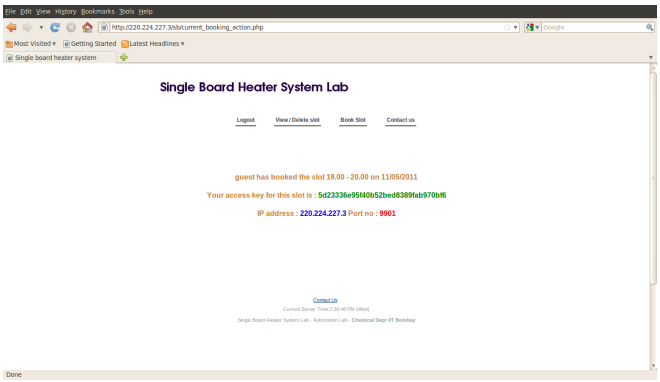

Fig. 7. Screenshot of Slot Booking Page

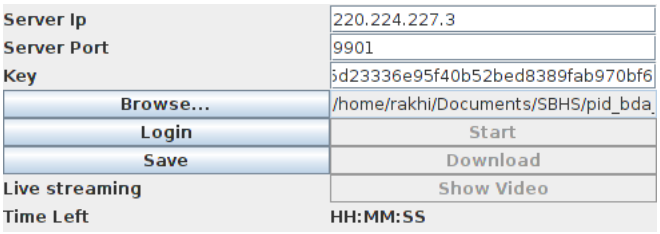

Fig. 8. Screenshot of Client Java Login Window

port number and an access key. The interface depicting this is shown in Figure 7.

3) Port number: In order to maintain consistency in performing the experiments, each client is alloted the same setup every time s/he requests for the slot. To facilitate this, each setup has been alloted a machine I.D. For example, if the machine I.D. of the unit is 5, the port number for communication would be 9905. This eliminates time and effort spent in plant modelling, each time, due to change in the setup.

4) Access key: An access key is an alphanumeric 32 character random key generated by the server for authentication. Before starting the experiment, the user needs to launch Java RMI client, and enter the access key along with the port no. The client is allowed to perform the experiments only if these fields are found to be genuine. The client login window facilitating this is shown in Figure 8 .

5) Starting the Java client: After keying in the server IP, port number and access key, select the path where the clientread and clientwrite files reside. We will explain the significance of these files later. Now click on `login` button and click `start`. This starts the communication with the server. The client java login window shows the time remaining in the slot.

6) Video streaming: At present, one unit connected to the servers (both internet and intranet) has a camera mounted above the LCD. This allows the client to view the change in the heated plate temperature with the change in heater and fan inputs. The video streaming of the capture display is shown in Figure 9. The clients with lesser bandwidth have an option to refrain from video streaming. To start video streaming, s/he should press on the `show video` button.

Fig. 9. Screenshot of Video Stream

7) Executing the Scilab code: The client needs to download an example code from the SBHS home page. This is available under `Downloads`. The client can modify this code to implement his/her own algorithm. S/he should not edit the part of the code which is used for communicating with Java. This part is clearly marked in the code.

8) File handling: This subsection explains the significance of clientread and clientwrite files. During the experiment, the client writes heater and fan inputs to the clientwrite.sce file. These values are read by the Java RMI client and sent over the network. Java server at the other end reads these values and gives them to the SBHS through its data acquisition interface. After feeding the values, it reads the temperature of the heated plate and sends it to the Java client. The Java client writes all these values i.e. heater, fan and temperature along with the timestamp to the clientread.sce file. Scilab client reads the latest temperature value and does the calculation of heater and fan inputs in accordance to the logic developed for the experiment. The live data streaming involves heater and fan inputs being sent by the client and temperature response in turn being sent by the server. The server timestamp accompanying the server data could be used for real-time control of the setup.

9) Concluding the experiment: Once the client is done with the experiment, s/he can download the log files by clicking on `Download` button available on the Client Java login window. Then close this window. S/he can now use the log files for analysis.

C. Other Implementation Issues

1) Mapping of machine id with the USB port number: Whenever a SBHS unit is plugged in to a USB port, the dev id (/dev/ttyUSB*) assigned to it by OS is different. A script is run which creates a table of the mapping between the machine I.D. and the USB dev id. This is done to ensure that the client gets the same machine I.D. i.e. the same SBHS each time.

2) Auto log off problem: In some ISPs, the network gets disconnected if it is inactive for some time. To handle this situation, we are running a shell script which checks for internet connectivity by pingging google periodically. If the network is down then it will try to reconnect.

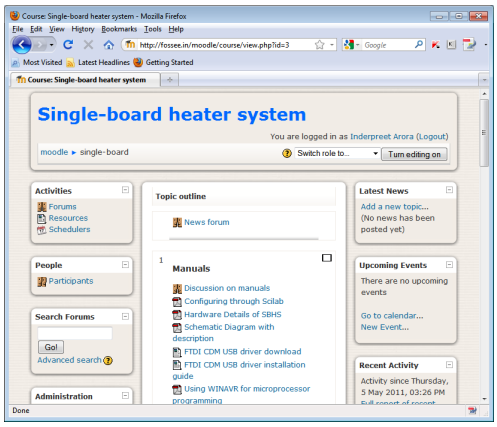

Fig. 10. Fossee moodle webpage to support SBHS activities

3) Video devices: The performance of the video devices is limited by the bandwidth of data bus connecting the device to the processor. By default a single video camera eats up 60% of the USB bandwidth. We have to compromise this by reducing the resolution and video quality. Using 320X120 resolution and JPEG compression value of 50 , we are able to reduce this usage to 30%. So a single root hub supports 3 devices. With 2 USB root hubs available by default for most of the computer's, only up to 6 devices can be supported. To make one computer support more video devices, we are exploring the possibility of using a multiplexer that would time division the captured video from each of the devices.

D. Support

The SBHS support activities are being funded by National Mission on Education through Information and Communication Technology [10]. The major support activities include conducting workshops in several colleges across the country, active discussion through Fossee moodle and spoken tutorials for self-learning.

1) Workshops: The SBHS team has been conducting workshops to popularise the utility of SBHS. The course content is around local and remote access of the SBHS. More than 50 college teachers from several Engineering colleges of the country have attended the workshops and have started using the setup for the relevant courses running in their colleges.

2) Fossee moodle: Fossee moodle is a web interface that allows discussions, post queries, upload files, upload grades, etc. Also, there is a facility to provide e-mail notification whenever there is a post on the course website. Thus, such an interface allows to address common problems of many users at the same time. It also provides a platform for the users at different locations to share their experiences during the experiments. The codes and manuals for about 8 experiments are available at [9].

3) Spoken tutorials: A Spoken tutorial is a screen capture along with the audio that could be used to teach any computer application. Currently, the spoken tutorials for various Free and Open source softwares are available in different Indian languages at [11]. The scripts and tutorials for SBHS are in pipeline.

VI. Experiments

About 10 experiments have been performed using SBHS locally while about 5 of them through remote access. The open loop experiments are not affected much with a moderate amount of network traffic. However, key learning is while implementing the control algorithms and observing the effect of network delay. The results of a few experiments are presented next.

A. Identification using step response data

As reported in [1], the step test experiment has been conducted by giving 2 step changes of 5 PWM units each and allowing the temperature profile to stabilize each time. The process model is then identified using this step response data [12], [13]. Notably, the remotely performed experiments reported in this paper are done on the same unit reported in [1].

For the given two step changes, we got the following models [1]:

$$G_1 = \frac{0.62}{44s+1}e^{-10s},\ G_2 = \frac{0.65}{50s+1}e^{-10s}$$

B. PID control

The efficacy of the PID controller, in discrete form, is explored next.

$$u(t) = K\left[e(t) + \frac{1}{\tau_i}\int_0^t e(t)\,dt + \tau_d\frac{de(t)}{dt}\right] \quad (1)$$

With the data acquired after every T_s seconds, the integral and derivative terms can be approximated as,

$$\int_0^t e(t)\,dt \approx T_s\sum_{i=0}^{t} e(i) \quad (2)$$

$$\frac{de(t)}{dt} \approx \frac{e(t)-e(t-1)}{T_s} \quad (3)$$

Using the reaction curve method for the models of the two step response experiments stated earlier, we arrive at the following values:

$$L_1 = 10, \tau_1 = 90, K_1 = 0.62$$
$$L_2 = 10, \tau_2 = 105, K_2 = 0.65$$

1) Local implementation: This experiment implements the PID controller locally i.e. the user has a unit connected to his/her computer.

Using the first set and the relations $K_p = 0.9/RL$, $\tau_i = 3L$, we arrive at:

$$u = \text{ubias} + K_p\left[\text{et} + \frac{\text{eti*Ts}}{\tau_i}\right]$$
$$K_p = 13.06, \tau_i = 30$$
$$\text{et} = \text{setpoint - temp, eti = eti + et}$$

where eti is the accumulated error. The response for this controller is depicted in Fig. 11. This has been described in greater detail in [1].

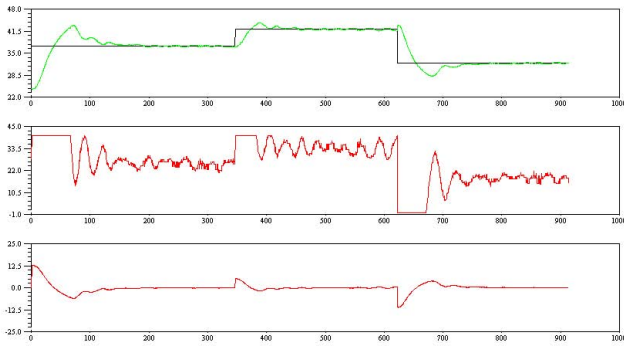

Fig. 11. Local PI controller implementation: temperature, heater duty, error

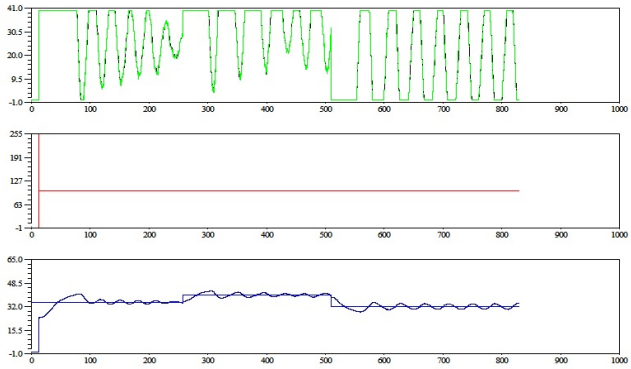

Fig. 12. PID control with local PID tuning parameters and remote implementation: heater duty, fan speed, temperature

2) Over intranetwork: The control effort for the same implementation but over network is found to be oscillatory. The experimental result is depicted in figure 12. Next, we re-tune the controller to get better results.

In order to implement an acceptable control algorithm, we either need to compensate for time delay or tightly tune the controller. The source of time delay is network traffic. With multiple applications running on the same machine and the sampling done every 0.4 s, a small amount of time is also spent in reading from and writing to a file at both the ends.

We implemented the controller with the tuning parameters, $K_p = 2$, and $\tau_i = 50$. The real implementation results are shown in figure 13. The tracking is found to be satisfactory.

The possibility of real-time control using a disturbance observer [14], [15] is being explored.

C. Two Degrees of freedom Controller

A Two Degrees of freedom Controller (2-DOF) strategy [16] was implemented remotely on SBHS through virtual lab. A block diagram for the same is as shown in the Figure 14. After performing the step test, the second order transfer funtion obtained was

$$G(s) = \frac{1.029}{(71.16s + 1)(s + 1)}$$

with time constant $\tau_1 = 71.16sec$, $\tau_2 = 1sec$ and gain $K = 1.029$. For rise time = 20 seconds and Overshoot of 5%, the

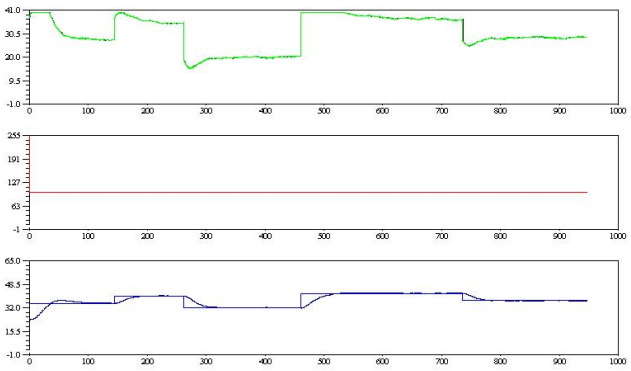

Fig. 13. Tuned PID controller and remote implementation: heater duty, fan speed, temperature

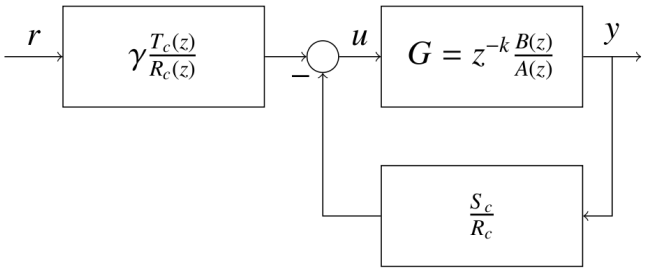

Fig. 14. Two Degrees of freedom Controller structure

various controller parameters obtained are

$$\begin{aligned} R_c &= 0.0143941 - 0.0345042z^{-1} + \\ &\quad 0.0265766z^{-2} - 0.0064665z^{-3} \\ S_c &= 0.0045849 - 0.0045528z^{-1} \\ T_c &= 1. - 0.9929982z^{-1} \\ \gamma &= 0.004589 \end{aligned}$$

The response of the controller implented on SBHS virtually is as shown in the Figure 15.

D. Identification of PID parameters using Relay Feedback Technique

Relay feedback auto-tuning method developed by Astrom and Huggland is one of the simplest and most popular auto-tuning technique for process control applications [17]. In this method, a simple experimental test is used to determine ultimate gain and ultimate period.

A feedback controller is temporarily repalced by an On-Off controller (or a relay) as shown in the Figure 17. When the control loop is closed, the control variable exhibits a sustained oscillations which is a property of On-Off controller.

Relay feedback method has been implemented on Single board heater system using virtual labs.

From the Figure 16, we can observe the sustained oscillations in temperature. Using the expressions derived by [17], the following parameters are obtained: The Ultimate period Pu = 25 seconds. The Ultimate gain Kc = 5.66. The PID parameters are then calculated from Ziegler-Nichols

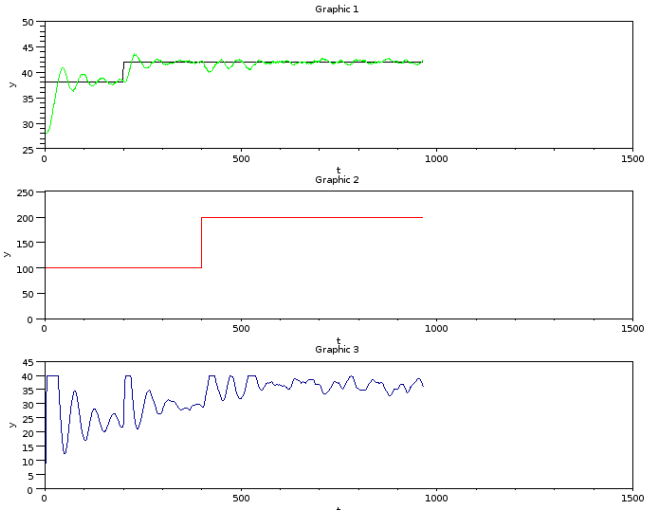

Fig. 15. 2-DOF controller implementation: temperature, fan speed, heater duty

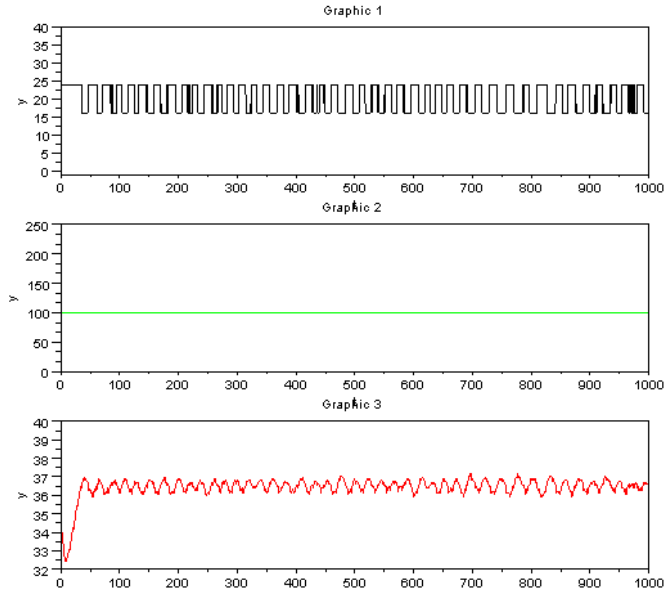

Fig. 16. Relay feedback implementation: heater duty, fan speed, temperature

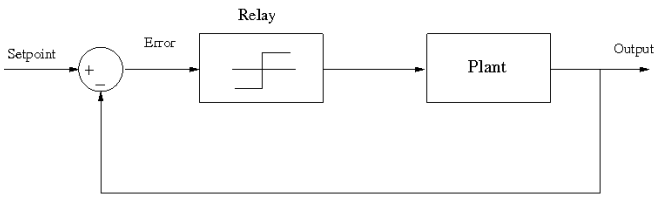

Fig. 17. Conventional relay feedback system

Type of controller	K_c	τ_i	τ_d
P	$0.5K_{cu}$	∞	0
PI	$0.45K_{cu}$	$0.8334P_u$	0
PID	$0.6K_{cu}$	$0.5P_u$	$0.125P_u$

TABLE I
ZIEGLER-NICHOLS CONTROLLER SETTINGS BASED ON THE CONTINOUS CYCLING METHOD

controller settings shown in Table I. The PID parameters are obtained as K_c = 3.3939394, τ_i = 12.5, and τ_d = 3.125.

VII. CONCLUSIONS

In this work, we presented a virtual laboratory that is based entirely on free and open source software. This solution is scalable as it is affordable. It is also in line with the objectives of the funding agency [10] that aims to promote open educational resources.

The solution presented in this work allows about 100,000 hours of experimentation time in a year with one or at most two server PCs working with about 20 SBHS. Thus, this configuration has the potential to cater to the million students who enroll in engineering colleges in India every year.

The main shortcomings of the approach followed in this work are due to the delay introduced in the loop by the Internet. We are at present studying the efficacy of delay observers [14], [15] to address this issue.

REFERENCES

[1] I. Arora, K. M. Moudgalya, and S. Malewar, "A low cost, open source, single-board heater system," in *International Conference on E-Learning in Industrial Electronics*. AZ, USA: IEEE, November 2010.

[2] K. M. Moudgalya and I. Arora, "A Virtual Laboratory for Distance Education," in *Proceedings of 2nd Int. conf. on Technology for Education, T4E*. IIT Bombay, India: IEEE, 1–3 July 2010.

[3] C. Ramos-Paja, J. M. R. Scarpetta, and L. Martinez-Salamero, "Integrated learning platform for internet-based control-engineering education," *IEEE Trans. on Ind. Elec.*, vol. 57, pp. 3284–3296, 2010.

[4] A. Balestrino, A. Caiti, V. Calabr, and E. Crisostomi, "DCL: a real time portable distributed control telelaboratory," in *18th Mediterranean Conference on Control and Automation, MED'10 - Conference Proceedings*. IEEE, 2010, pp. 185–190.

[5] S. Odeh, "Building reusable remote labs with adaptable client user-interfaces," *Journal of Computer Science and Technology*, vol. 25, pp. 999–1015, 2010.

[6] J. Gabriel and M. T. Restivo, "Hands-on Using On-line Engineering: The Trend to Better Solutions," in *Proceedings - ICELIE 2009, 3rd IEEE International Conference on e-Learning in Industrial Electronics*. Porto: IEEE, 2009, pp. 64–68.

[7] A. Leva and F. Donida, "Multifunctional remote laboratory for education in automatic control: The CrAutoLab experience," *IEEE Transactions on Industrial Electronics*, vol. 55, pp. 2376–2385, 2008.

[8] Virtual labs project, "Single board heater system," http://www.co-learn.in/web_sbhs, seen on 11 May 2011.

[9] "Fossee moodle," http://www.fossee.in/moodle/, seen on 10 May 2011.

[10] K. M. Moudgalya, "Introducing National Mission on Education through ICT," http://www.spoken-tutorial.org/NMEICT-Intro, 2010.

[11] "Spoken tutorials," http://spoken-tutorial.org/Study_Plans_Scilab, seen on 10 May 2011.

[12] K. J. Åström and T. Hägglund, *PID Controllers*. Instrument Society of America, 1995.

[13] D. E. Seborg, T. F. Edgar, and D. A. Mellichamp, *Process Dynamics and Control*. John Wiley & Sons, Inc., 2004.

[14] C. J. Kempf and S. Kobayashi, "Discrete-Time Disturbance Observer Design for Systems with Time Delay," in *Proceedings of the 4th International Workshop on Adv. Motion Contr.* Tsu-city, Japan: IEEE, 1996.
[15] R. W. Jones and M. T. Tham, "Disturbance observer design for continuous systems with delay," *Asia-Pacific Journal of Chemical Engg., Wiley Interscience*, vol. 2, 2007.
[16] K. M. Moudgalya, *Digital Control*. John Wiley & Sons, Ltd, Chichester, 2007.
[17] K. J. Åström and T. Hägglund, "Automatic tuning of simple regulators with specifications on phase and amplitude margins," *Automatica*, vol. 20, pp. 645–651, 1984.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the funding support from the National Mission on Education through Information and Communication Technologies [10].

The authors would like to thank Ankit Bahuguna, Harsh Yadav, Piyush Joshi, Kushal Thakkar, Swanand Kulkarni, Aditya Sengupta, Sushanth Poojary, Tanuj Bhojwani and Sitesh Patel for their inputs for a few software modules.