Running Control Engineering Experiments Over the Internet

Carisa Bohus   Burçin Aktan*   Molly H. Shor†   Lawrence A. Crowl

Department of Computer Science
Oregon State University
Corvallis, Oregon 97331-3202

Technical Report 95-60-07

August 1995

Abstract

An important issue in engineering education is the availability of laboratory resources for student use. Using a computer network to link geographically distant students with laboratory teaching resources makes expensive and innovative equipment available to more students. At Oregon State University, we provide a working environment where remotely-located students can develop and run controllers on experiments in our control engineering laboratory. Remote users can watch the experiment in real time from a remote workstation, hear the sounds in the laboratory, and interact with other laboratory users. Remote power control, network reliability, and safety features are integrated into our experimental hardware and software design.

Key Words: control engineering education, remote control, distance learning, Internet.

This work was supported in part by Oregon Joint Graduate Schools of Engineering under NASA Grant NAGW-3965 and in part by the National Science Foundation under Grant DUE-9352734

*Department of Electrical and Computer Engineering

†Department of Electrical and Computer Engineering, Phone: 503-737-3168, Fax: 503-737-1300, E-mail: shor@ece.orst.edu

1 Introduction

Practical experience is a very important part of control engineering education, but it is resource intensive. Innovative control experiments can take time, money, and energy to design and to construct, and are often not fully utilized throughout the academic year. Sharing experiments remotely enables greater use of unique laboratory equipment, brings down the experiment cost per student, and makes more experiments available. Our goal with the remote lab paradigm is to provide remote access as effective as local access.

Control engineering instruction should combine theory and practice in each lesson. Students must be able to model systems adequately in order to develop controllers that enforce certain performance requirements. After a controller is designed and simulated on the model, observing the dynamics of a physical implementation gives the student valuable insight. Data collection and visual feedback are important aspects in control engineering instruction for the modeling, system identification, and testing stages. In the past, students had to be in the laboratory to gain practical experience; now they can be anywhere.

Distance learning is an emerging new paradigm where students, teachers, and equipment may be in geographically different locations. Second Best to Being There (SBBT) is a network application combining new and existing software and hardware to provide remote laboratory users the opportunity to conduct live experiments off-site. For SBBT, the Internet provides the communication infrastructure between students and the experiments (see Figure 1). This may be the first time an undergraduate laboratory has been made fully accessible using computer networking tools.

In the next section we discuss related work. Section 3 details the remote lab paradigm, and describes the trade-offs we made in our implementation. Section 4 provides an overview of the hardware. Finally, we conclude with the main lessons from taking our design to implementation.

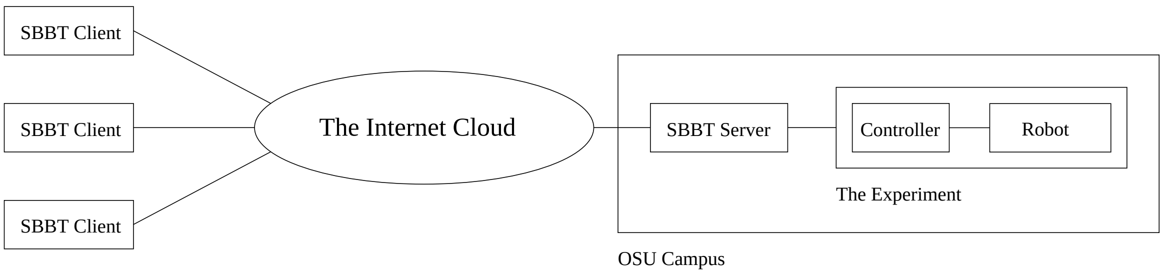

Figure 1: **Internetworking Context for the SBBT Application.** SBBT is a client/server application that enables distantly located students to control experiments on the Oregon State University campus.

2 Related Work

Three distinct areas of research stand in contrast to SBBT. They are telerobotic systems, virtual reality or simulation systems, and large multi-location process control automations. SBBT does

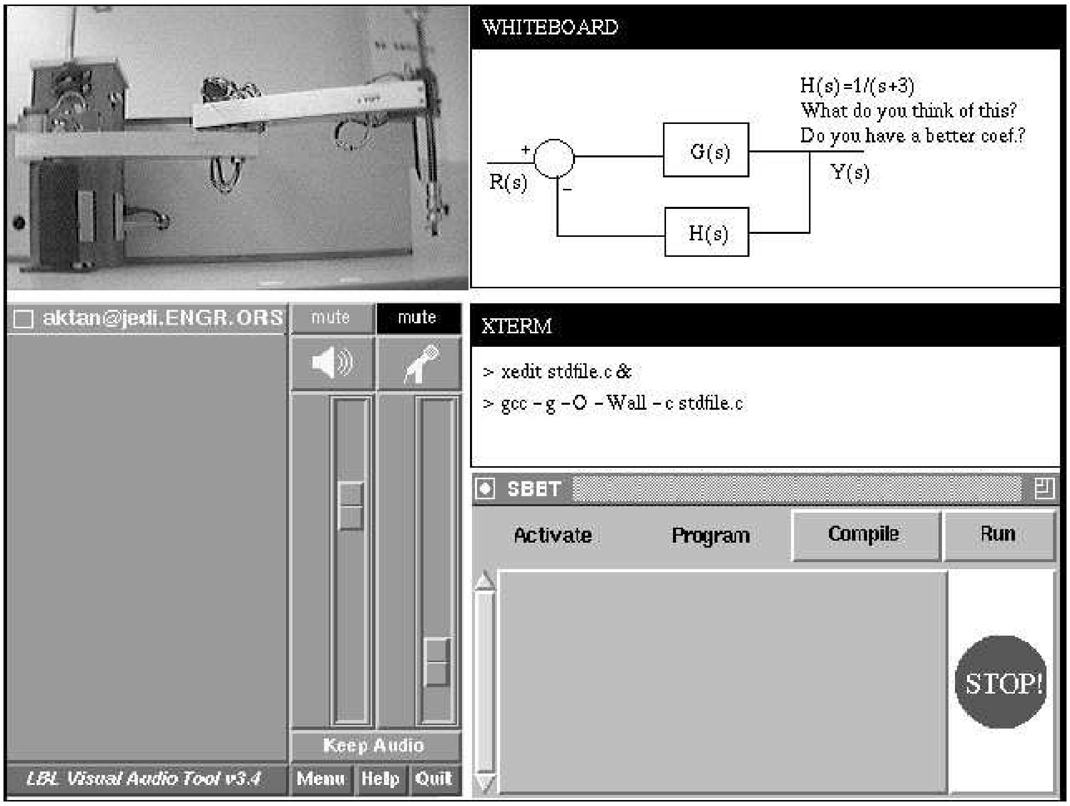

Figure 2: **The Remote Lab User Interface for SBBT.** This is the display the remote student will use to conduct experiments. Clockwise from the top left corner: (1) video window of the control experiment, a 3-DoF robot arm, (2) collaboration tool showing a block diagram and some discussion, (3) an Xterminal window representing the local development environment, (4) the lab environment control window with the panic stop button and (5) the audio configuration window.

not duplicate these important application areas; it gives students exposure to equipment they lack at their own site.

Telerobotics share some characteristics with SBBT; both operate over long distances, and both work with moving systems. But, telerobotics, where remote movement is generated by intentional force by the operator, is fundamentally different since it has an human operator directing the remote actions. For example, Lee and Lee [7], describe a real-time telerobotic system that was designed for use in outer space repair missions. The teleoperator uses force on a sensory device (e.g., a mouse or joystick) to direct the actions of a robot arm at the remote station. Another example of telerobotic systems are available on the World-Wide Web. These applications enable the distant user to move robot arms which tend a garden [11], or move blocks on a table [12, 10]. Again, the user clicks on a schematic, or still picture to move the robots, thus directing the action. In contrast to telerobotic systems, our system provides an environment for experimentation with control code, which is downloaded to the equipment and then run on the equipment itself. That is, instead of transmitting control signals, the remote student transmits control programs which are run, enabling the equipment to handle tasks autonomously.

Robotic systems are being developed to accomplish increasingly complex tasks. Appropriate training on complicated equipment is critical. Miner and Stansfield [9], describe a method to fulfill these training requirements. They designed a virtual reality robotic control system to train operators more efficiently in real-time operation. Virtual reality can be thought of as an interactive user interface to a simulation system. Simulation provides a cost-effective way to learn systems that are expensive or potentially dangerous to staff with inexperienced personnel. However, simulation means working in a closed universe, which omits some physical realities. There will always be an important place for simulation systems, but they cannot completely substitute for experience with actual systems.

Researchers [3, 6], from five institutions have developed a factory automation testbed for distributed telerobotics research. The five sites, which are connected by the Internet, provide an experimental environment for standardizing device interfaces, testing protocols, distributing tasks, developing user interfaces, and supporting collaborations. SBBT, in contrast, is designed to provide a more intimate setting, where one student can design and carry out an experiment entirely on his or her own. The emphasis in our design is to focus the remote user's attention on the experiment, not on the system that supports the experiment.

Several distance learning projects-in-process are noted in appendix A.

3 Paradigm and Application

Imagine a control engineering student, developing a controller for a 3-DoF (Degree of Freedom) robot arm. Her school doesn't have a robot arm, but a university 85 miles away does. She sits in front of a computer screen and downloads her code to the distant university robot arm experiment over the network. As soon as the control code is compiled, linked and loaded, she can watch her code run on the robot, in real-time. She observes that the controller performance does not meet her design goals, modifies her controller code and runs it again, repeating this until she is satisfied. She is working in a remote laboratory using a new paradigm.

The remote lab paradigm is based on a software user interface and a hardware configuration. In this section, we describe our approach and implementation of the user interface; the hardware configuration is discussed in Section 4. There are five main functional parts (*components*) to the user interface: (1) the experiment, (2) lab environment control, (3) lab presence, (4) collaboration, and (5) safety. We developed an open architecture for the overall system, with a variety of implementation choices for each component, balancing costs and functionality needs.

3.1 The Control Engineering Experiment

This component is basically unchanged from what it would be conventionally. Common experiments in our lab are x-y positioning tables, robot arms, DC motor control, and magnetic suspension systems. Criteria to consider when selecting an experiment for remote use fall into three categories: economics, logistics, and appearance.

If substantial time, human, or financial resources have been dedicated to design and build an experiment, it is a worthy candidate to make available to remote students. The cost of simply replicating an experiment should be compared to the effort and expense of installing SBBT, keeping in mind that most SBBT costs occur only once, while replication costs scale. If replicas cost more than SBBT, it makes sense to use SBBT.

The logistic considerations, which can be automatic or manual, are (1) remote power control, (2) safety for people and property in the lab, (3) ability to run without human intervention, (4) a stable start position, (5) at least one reset position,[1] and (6) the ability to download control code. The importance of finding the proper solution to each of these concerns should not be underestimated. Although time-consuming, representational views from students, professors, and supporting technicians are critical to a successful experiment. At each stage, from design to implementation, all procedures and interfaces should be reviewed. These evaluations serve not only to verify understanding, but also to provide informal training.

Regardless of the type of system the controller runs on (e.g., personal computer, real-time operating system on a workstation, DSP board, or embedded controller), the ability to download the experimental control code from the remote site is mandatory.

If video equipment is available, consider what visual information the experiment will offer to the remote user. From watching the experiment, will the student get a good grasp of the overall behavior? For a robot experiment this is obvious, in the case of a DC motor, perhaps alterations such as notches on the rotor, will give additional information. In general, any experiment that has movement is a candidate. Other appearance criteria, mostly for demonstration purposes, is whether the equipment is unique or interesting to watch.

We had three candidate experiments to test the feasibility of SBBT: an inverted pendulum, an x-y table, and a robot arm. We ruled out the pendulum experiment because it required modifications for the reset operation (a separate set of arms that would close and set the pendulum upright). Because the pendulum and the x-y table were in active use by other students, we anticipated contention for their use, which would have made development difficult. We chose the 3-DoF robot arm. It was a good first choice since it contained no loose pieces, was easily reset, and would provide visually interesting demonstrations. In all cases, we could download control code to each of the experiments, the mandatory logistic condition.

3.2 Lab Environment Control

Since the student is not in the room where the experiment is located, this component carries out the functions students normally perform themselves in person, such as resetting the apparatus. In effect, this piece replaces the student in the lab. The main functions required for permitting remote operation and control of an experiment are (1) controlling the power to the experiment, (2) resetting the experiment, (3) downloading control code, (4) collecting data during experimental runs,(5) determining equipment status, and (6) interrupting normal operation. These tasks are so general that every experiment requires them; additional functions are necessary only to specific experiments. We present a table of the fundamental operations in Appendix B.

[1]The reset position may be the same as the start position.

During the design of this component we maintained a list of every possible feature a student might find useful. We ranked the items to determine a minimal but complete environment. By tracking every idea we were able to stay focused on the minimal set knowing all ideas were captured and could be reevaluated and implemented in a future version.

We implemented this component as a client/server network application. The user interface was implemented as an X11 window, with buttons to click on, rather than using command line directives (see Figure 2, Part 4). This user-interface method keeps command details that may change or be mistyped out of the user's way. We wanted to emphasize ease in conducting the control engineering experiment and not burden students with learning a new system.

3.3 Lab Presence

Real systems with moving parts are exciting to work with. Being remotely connected, trust needs to be established that student initiated actions are indeed being relayed to the distant site. Some feeling of the lab environment must be communicated to the student. Of the five basic senses, sight and sound can be transmitted over computer networks easily.

SBBT strives to give users a genuine sense of actually being in the lab. Real-time video over the network was an early candidate for giving remote users live information as their experiment runs. We also evaluated a network audio tool. Video and audio tools are often bandwidth-intensive, which the existing network may not tolerate gracefully. Depending on the experiment, text feedback may be a low-cost alternative to video. We chose the network tools for video and audio (`vic` [8], and `vat` [4], respectively) developed at LBL (Lawrence Berkeley Laboratories), based on their configurability to conserve bandwidth and their contribution as a window into the lab (see Figure 2, Parts 1 and 5).

3.4 Collaboration Support Tools

Anecdotal evidence shows that students working in the same space exchange a lot of information. Distance learning nearly guarantees that students will not be sharing the same space. Without explicit support for communication with others, the sense of isolation is real. Not only must the communication tools exist, but there needs to be a reason to use them.

One solution to help remote students get acquainted with each other was to establish a *social protocol* for scheduling the equipment, rather than relying on a computer program to arbitrate access to the SBBT system and experiment. Turning over the use of a class experiment is a very natural crossroad where two students must essentially *meet* and negotiate. From this exchange, ideally students will linger and talk about their findings, help each other, and form collegial relationships.

To support collaboration, we used a tool called `wb` (whiteboard) [5], also developed at LBL. This tool is a piece of the CRT screen shared among all the participants, which receives text and drawings simultaneously at all locations connected to the SBBT session (see Figure 2, Part 2). Other options for collaboration are additional video cameras, email, and the UNIX program `talk`. Making as many collaboration tools available as possible allows the users to develop their own interactive environment.

3.5 Safety

Overall, the system needs to be safe. Not only do people and property in the lab need to be protected from unanticipated experiment movement, but in the case of any mishap the student will not be there to supervise. On the other hand, people learn from the natural process of making mistakes. An instructional laboratory must allow a range of errors and still be a safe environment.

We surveyed professors, technicians and students, both at our university and others, for their experience with effective laboratory safety. Using this information, we identified safety procedures and mechanisms to ensure that students and equipment in the lab would be protected. Some hazards are difficult to communicate to a remote user. The smell of wires burning, for example, is quite distinctive, and yet it is not the kind of information easily transmitted to a remote user over the Internet. Other mechanisms should be installed and monitored for specific hazards. See Appendix C for a compilation of hazard analysis for engineering teaching laboratories.

We established fail-safe mechanisms in three areas of operation: (1) automatic mechanical controls, (2) SBBT system control, and (3) student control. These measures combine to create a comprehensive safety barrier.

On an automatic mechanical level, we outfitted the lab with safety mats to protect those working in it. If the motors are on, indicating the experiment is live, any pressure on the mats automatically shuts off the motors moving the robot arm. Alarms warn local users when the power to the motors is being turned on.

Next, the SBBT system maintains a constant heartbeat signal. The SBBT session server sends a signal to the client program running at the remote site. The client acknowledges this special heartbeat signal by sending it back. If too many heartbeats are missed, indicating a loss of network connectivity or an excessive delay in the network, the power to the experiment is automatically turned off.

The third safety measure is under user control. The remote student, seeing trouble, can stop the experiment immediately by clicking on their screen's virtual panic stop button (see Figure 2, Part 4). There is also a physical stop button in the laboratory on the experimental apparatus.

4 Hardware Configuration

The basic control engineering experiment consists of the controller and the system being controlled. To make the experiment accessible to a remote student, another level of equipment control was introduced, consisting of a Motor Control Interface (MCI) and UNIX workstation. The MCI provides remote power control and enables the implementation of safety features. The workstation supports network video and audio and connects the controller and the MCI to the Internet (see Figure 3). Our hardware configuration enables overall connectivity and comprehensive safety features.

4.1 Motor Control Interface (MCI)

The custom-built MCI handles all of the safety features through basic power access. The MCI receives orders from the workstation using its own serial line. The MCI relays on/off directives to the experiment's power supplies. The robot motor power on/off functions are used by normal

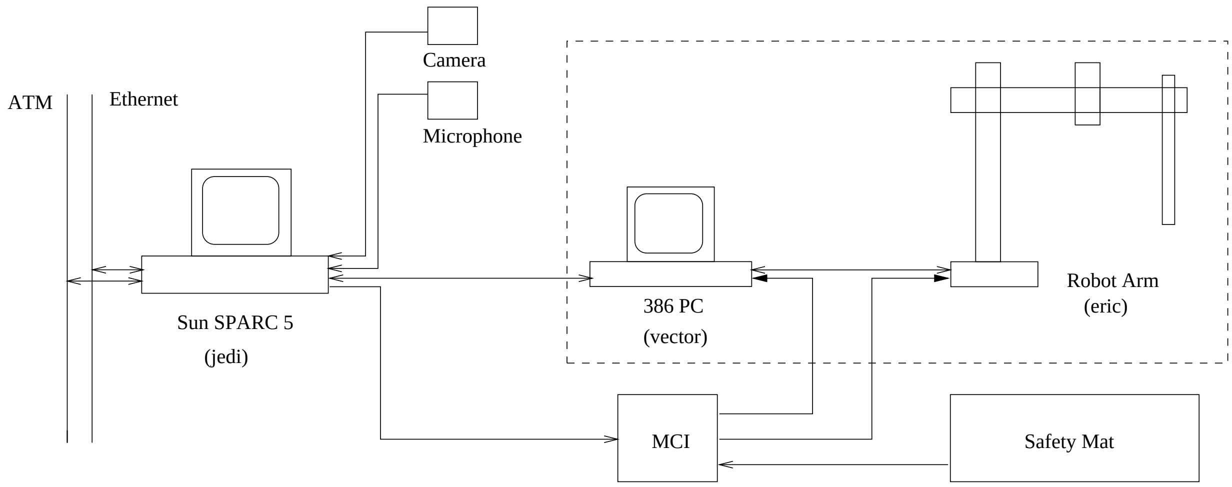

Figure 3: **SBBT Hardware Configuration.** The inset box shows the conventional hardware setup for a control engineering experiment. Outside this box, the workstation is used for remote connectivity and the custom-made Motion Control Interface box for power control.

---

start and stop, by the local and remote panic stop button, by the safety mats, and by the system heartbeat. The PC power on/off is used to reboot the PC. The MCI makes it possible to have the robot available 24 hours a day, since it provides the ability to turn the equipment on and off.

4.2 Workstation to PC Communications

Dependable communications between the workstation and PC are mandatory. All the source code and data communications occur via this link. The communication is implemented through a full duplex, RS232 serial connection. Tasks are performed by shell scripts on the workstation and corresponding batch files on the PC. By implementing these directives at the operating system level, we gained flexibility and portability. These communications are transparent to the user, who can observe the effect of his or her controller by video and audio feedback and data returned from the PC.

5 Conclusion

We learned several lessons developing the remote lab paradigm:

- Audio and video are valuable. They entice use, support collaboration, and give the remote user the feeling of being in the laboratory.
- All software tools under consideration should be evaluated thoroughly for their operational details. We spent some time making efficient configuration choices to accommodate the high-bandwidth tools such as audio and video.

- Safety features are not only necessary but may also permit other remote functionality. For example, the MCI supports remote power control and initialization of controller devices.
- An open architecture helps both in porting the system to new environments and new experiments, and also allows rapid prototyping. For example, we used shell scripts to implement the user directives. As a result, incorporating new commands became trivial.
- Collaboration can be encouraged. We made the controversial choice to rely on a social protocol for experiment scheduling, giving students the opportunity to interact.
- Make as many communication tools available as possibile. More collaboration tool choices lets students create effective environments to learn from each other. In a developing paradigm, we felt it would be wiser to offer more choices than to have a regimented approach.
- Working with a multi-disciplinary team provides many valuable perspectives during the architecture review, as well as promoting the development of a common vocabulary. Our system was developed by computer scientists, electrical engineers, and control engineers, with input from mechanical and industrial engineers. Representation of technical support, students, and professors from complementary fields provides a basis for a system that everyone will enjoy using.

SBBT is an educational motivator to master all the necessary steps from application of appropriate theory to design, since it is fun to use. We developed a paradigm for remote lab use and demonstrated the feasibility through an implementation. Although we concentrated on one experiment, the remote lab paradigm clearly applies to a wide range of experiments. The remote lab paradigm provides access to equipment otherwise out of reach.

Acknowledgement

We wish to thank Juliet Hyams for her contributions.

References

[1] Announcement in *Currents*, a publication of the Electrical and Computer Engineering Department, Carnegie Mellon University, Spring 1995.

[2] Mike Driscoll. From conversations, GPIB bus project, Spring 1995.

[3] Sean Graves, Larry Ciscon, and J.D. Wise. A modular software system for distributed telerobotics. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, pages 2783–2785, Nice, France, May 1992.

[4] Van Jacobson and Steve McCanne. vat (Beta release). Available by ftp from ftp.ee.lbl.gov under conferencing/vat (e-mail contact van@ee.lbl.gov), 1994.

[5] Van Jacobson and Steve McCanne. wb (LBL whiteboard) version 1.59, Beta release. Available by ftp from ftp.ee.lbl.gov under conferencing/wb (e-mail contact van@ee.lbl.gov), 1994.

[6] George V. Kondraske, Richard A. Volz, Don H. Johnson, Delbert Tesar, Jeffrey C. Trinkle, and Charles R. Price. Network-based infrastructure for distributed remote operations and robotics research. *IEEE Transactions on Robotics and Automation*, 9(5):702–704, October 1993.

[7] Sukhan Lee and Hahk Sung Lee. Modeling, design, and evaluation of advanced teleoperator control systems with short time delay. *IEEE Transactions on Robotics and Automation*, 9(5):607–623, October 1993.

[8] Steve McCanne and Van Jacobson. vic (VIC 2.6 BETA). Available by ftp from ftp.ee.lbl.gov under conferencing/vic (e-mail contact mccanne@ee.lbl.gov), 1994.

[9] Nadine E. Miner and Sharon A. Stansfield. An interactive virtual reality simulation system for robot control and operator training. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 1428–1435, 1994.

[10] Ken Taylor and James Trevelyan. A telerobot on the world wide web. In *Proceedings of the 1995 National Conference of the Australian Robot Association*, 1995.

[11] The Tele-Garden: A tele-robotic installation on the WWW. On the Web at http://www.usc.edu/dept/garden, 1995.

[12] Australia's telerobot on the web. On the Web at http://telerobot.mech.uwa.edu.au, 1995.

A Related Projects-in-Progress

The Internet provides an environment for many kinds of resource sharing. We would like to acknowledge these works in progress:

Carnegie-Mellon University [1] and Portland State University [2] are developing network accessible electronics laboratory resources. These remote labs feature GPIB tools for electrical engineering experiments with no moving parts.

The University of Essex, Department of Electrical Systems Engineering, is in the initial stages of creating an interactive electronics hardware laboratory for use over the Internet, specifically over over the WWW.

While this list is far from exhaustive, it shows there is confidence in using the Internet to provide students with more kinds of learning opportunities.

B Lab Environment Control

This table shows the main function names and actions for the SBBT control window.

Main Functions	**Explanation**
sbbt	Start up the application for a work session. If the experiment is already in use, a communication session is set up between the parties so scheduling can be negotiated.
quit	Release all SBBT resources.
stop	Immediate shutdown of motors.
reset	Put the experiment in a predefined, stable state.
download	Transfer control code or data to the target controller.
reboot	Turn off power for several seconds to force the PC through a reboot sequence.
compile	Compile and link the control code on the target machine.
run	Execute the most recently compiled control code.
getdata	Transfer experiment output data to the user.

C Hazards Analysis

Through surveys, we have compiled the following list of hazard categories and safety precautions.

Hazards Categories

	Hazard	Possible Causes
1.	Hazard due to students learning	The software or controller (hence robot) may not do what students think has been programmed. The hardware or software initialization or calibration may be off, resulting in unintended robot action. The gains or controller may be inappropriate, resulting in instability of the closed-loop system, hence undesirable robot behavior.
2.	Hardware setup incomplete or incorrect	The hardware connection may be loose, missing, or improperly prepared, resulting in incorrect or noisy signals from or to robot, resulting in unintended controller or robot action. The robot may not be left in operable condition by the on-site users, causing problems in hazard category 1 or rendering the robot inaccessible to distance users.
3.	Observers in the way	A person or object may obstruct the robot in the lab, resulting in an accident during normal robot action.
4.	Environment hazards	A person tripping over cables may pull down and damage the attached equipment.
5.	Unsafe experiment start-up	Someone may turn on power before all switches and mechanical parts are on the safe power-on positions, disrupting safe power-on sequence.

Safety Precautions

	Electrical
1.	Check all switches for default positions before main power is turned on.
2.	Verify all electrical connections are intact, including cards.
3.	Verify that no electrical conducting media (wires, people) are touching any live wire.
4.	Check power-on sequence.
5.	Verify startup transients do not damage equipment.
	Mechanical
1.	Verify no person can obstruct any moving parts.
2.	Verify no wires or cables are in the way of human or machine movement.
3.	Verify no piece of equipment can obstruct other equipment.