Automatic verification of the remote laboratory NetLab

Hugh Considine, Andrew Nafalski & Zorica Nedic

University of South Australia
Adelaide, Australia

ABSTRACT: Remote laboratories provide access to equipment outside of a classroom environment. The NetLab remote laboratory allows students to conduct experiments in electrical engineering from anywhere in the world at any time. The current work focuses on improving the support available to students through the addition of an intelligent tutoring system (ITS). Research exists to show that the addition of an ITS to a remote laboratory can improve student learning. In a conventional real laboratory, the laboratory supervisor would be responsible for ensuring the correct functioning of the equipment, as well as warning students if some piece of equipment has become faulty and requires repair or replacement. In this article, the authors describe the development of an automatic verification system, which fulfils these two important roles in the remote laboratory context. It can run batch test scripts on the system at a time when no user is connected, providing early warning of any issues to the teaching staff. Even if the system cannot directly fix all issues, providing instructions to the human supervisory team and the students is potentially very useful.

INTRODUCTION

Remote laboratories (RLs) provide numerous advantages for both the learners and educational institutions [1][2]. The RLs using real equipment remotely combine the advantages of on-line access to real laboratories with real experiments as opposed to non-idealised equipment in a virtual/simulated laboratory [3]. The RLs provide flexibility, 24/7 access and access for people with physical disabilities. Safety is improved since there is no contact with electrically live equipment and there is the option to repeat experiments to correct/perfect results that is seldom available in real laboratories. In RLs usually one laboratory set is required, especially in booked systems. There is no need for costly supervision by humans and sophisticated scheduling, significantly reducing investment and running costs for administrating institutions.

Remote laboratories can be shared between institutions, allowing worldwide access to unique and/or expensive equipment otherwise not affordable or accessible. Many remote laboratories allow students to conduct experiments collaboratively with laboratory partners either sitting next to each other at the same computer or using different computers on the same campus or at home or indeed from any location in the world with Internet access. The RLs, such as NetLab provide a collaborative environment for distant users. This provides a unique opportunity for creating communities of practice, greatly enriching students' learning experience [3]. Explaining to students the advantages of using remote laboratories, including internationally, will increase students' motivation and engagement with the laboratory, with the prospect of being a part of a growing world community of remote laboratory learners.

As RLs are available 24/7, human tutors are unlikely to be available for assistance in a synchronous mode, thus leaving students with difficulties conducting experiments on their own. Therefore, the development of a 24/7 intelligent tutoring system (ITS) for the remote laboratory NetLab at the University of South Australia (UniSA), with a possibility of generalisation for other remote laboratories, is highly desirable. The process of the design and development of ITS has been reported in a number of publications [2][4-8]. The similarities of remote laboratory self-testing facilities with other electronic self-testing systems have been identified [9].

In this article, the focus is on auto testing of NetLab, so that errors in key functions are discovered and the supervisory team and users are notified. This particular module will form an important part of the intelligent tutoring system under development.

THE NETLAB

The remote laboratory NetLab, and its applications, have been presented in a number of recent publications [2][10-12]. NetLab is an open access system that requires only registration and can be used from anywhere in the world. It has been

in operation since 2002, constantly developed and improved to become robust, reliable and versatile. Thousands of users from nearly 60 countries use it on average of between 600 and 900 times per annum. To access NetLab, the following two addresses can be used: http://netlab.unisa.edu.au or http://netlab2.unisa.edu.au. The two systems are nearly identical.

The NetLab main Web site includes all information on the system, with examples of experiments. After registration, the user needs to book a time to use the system. NetLab is useful for performing experiments for electrical circuits with passive components.

DESIGN OF THE TEST SYSTEM

While there are few problems experienced with NetLab - considering the amount of usage - from time to time power interruptions or invalid input from users can put some part of the system into an unresponsive state that requires a physical reset. Additionally, it is possible for a user to create a circuit that trips some protection feature in the system and disconnects power to components. All testing of the NetLab remote laboratory has been done manually. During busy times, the system can remain in a faulty state as several novice users attempt to conduct experiments, until a student notifies a member of the teaching team or until a tutor logs in to manually test the system.

By automating the testing of key functions, any issues that have occurred can be identified much more quickly. The testing system connects to the NetLab servers in the same way that the NetLab client does, and sends exactly the same commands. When started, the main window of the test system appears as shown in Figure 1.

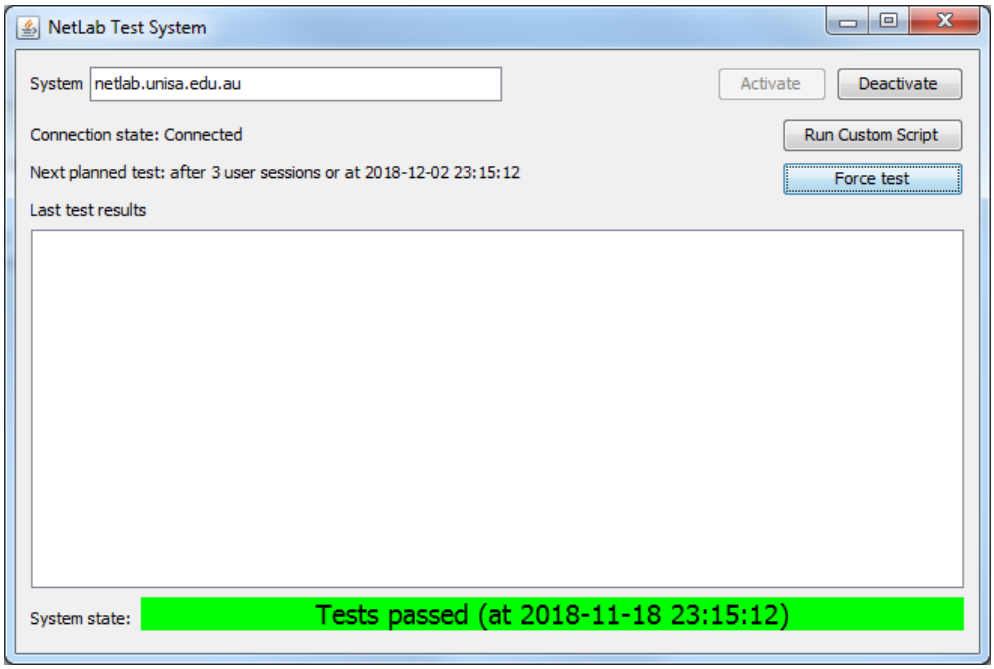

Figure 1: The system after a successful test.

The testing system normally remains dormant, connected to the NetLab server but not controlling it. The test system uses a special user account on the server, but is invisible to users. If the connection is interrupted, the test system will automatically reconnect. While connected, the test system monitors the other connections to the server.

The testing process is triggered by one of two events: user sessions ending or a length of time since the previous test. The testing process needs to be timed appropriately - to test too infrequently would allow issues to affect more users before being detected - and to test too frequently would add wear to the relays in the switching matrix of the NetLab system. Currently, the tests are set to begin after three user's login and logout events have been recorded, ensuring that tests are conducted regularly when the system is busy. The tests are also triggered when a period of two weeks has elapsed since the previous test, ensuring that the system is still tested during teaching breaks.

The testing will commence when a logout or time trigger is activated and no user is logged in. If a user is logged in, testing will be postponed and will commence immediately after all users have logged out.

When testing commences, both triggers will be reset. For example, if the third logout occurred one day after starting the test system, the time trigger would be reset to two weeks after the test, rather than triggering another test after 13 days. Similarly, if the system was in a quiet period and only two user sessions occurred in the two-week period, the time trigger would activate and reset the logout trigger. Another three user sessions would then need to occur to trigger the tests.

The test results are displayed on the main window of the system. Any detected issues are listed in detail in the *Last test results* section of the screen. The bottom of the window displays a traffic light indicator and a status message, showing whether the last test resulted in a pass or a fail. Test results are also e-mailed to the administrators of the NetLab system.

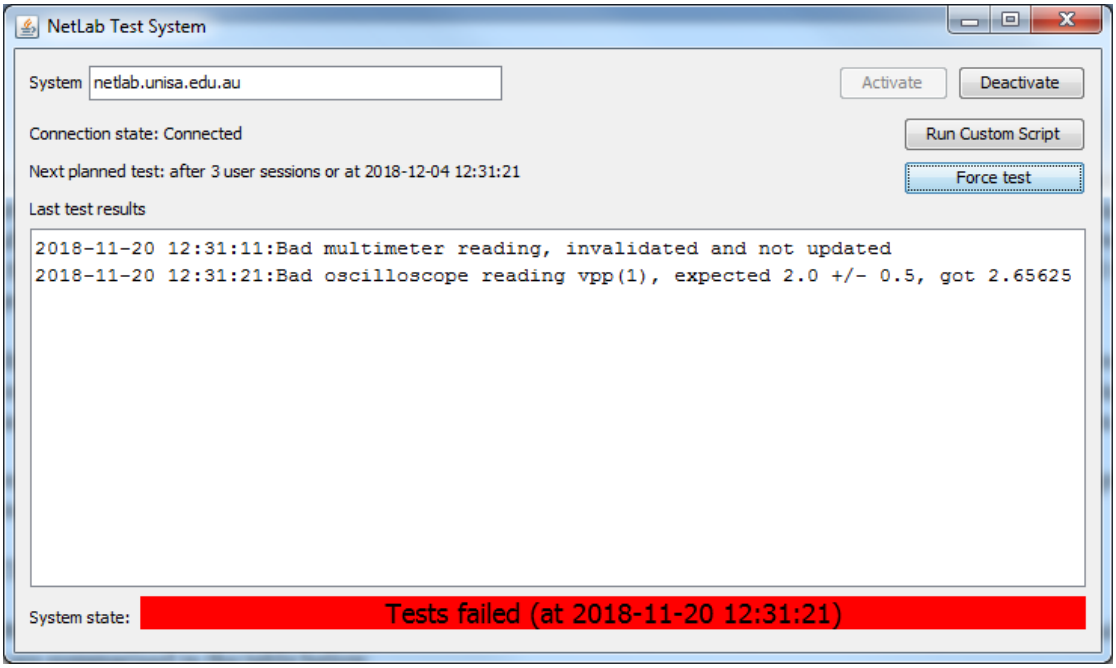

Figure 2: The system after a failed test.

Figure 2 shows the system after a failed test. The reasons for the failure are now shown. In this case, the failures were simulated by deliberate errors in the test script - no updated test readings are requested for the multimeter, and therefore none were received, and the oscilloscope readings did not match the values that the script was written to expect.

The test system also allows for scripts to be run interactively for testing purposes. Another window is provided, where scripts can be executed, and the commands and their results displayed in real time.

IMPLEMENTED TESTS

Tests are written using a basic scripting language. These scripts are run by an interpreter in the test system. Commands are summarised in Table 1.

Table 1 Test commands.

Command	Purpose
Control commands	
Repeat *n* ... End repeat	Commands placed between the *Repeat* and *End repeat* commands will be repeated *n* times. This command can be used to stress-test the system, repeatedly performing the same action in quick succession.
Wait *n*	Pause the test script for *n* milliseconds. This is used to reduce the rate at which commands are sent or to allow time for requested changes in the configuration of the laboratory equipment to take place.
Rate *n*	Adds an implicit delay of *n* milliseconds after every command. This is useful when running scripts interactively or to avoid multiple *Wait* commands, when dealing with slower parts of the system.
Device control commands	
fg	*fg* commands are used to control the function generator. Several variants of the command exist for setting the shape of the wave, the frequency, the amplitude of the signal, the DC offset, and the duty cycle of any square wave produced.
cro	*cro* commands control the oscilloscope. All oscilloscope functions can be controlled - the run and stop buttons, triggering, channel settings, and vertical and horizontal scale and positioning.
mm	*mm* commands control the digital multimeter. These can set the mode of the meter (AC or DC voltage or current, resistance or frequency).

Command	Purpose
Control commands	
build	The build command configures a circuit in the remote laboratory. A custom version of the NetLab circuit builder tool is used to generate XML code corresponding to the circuit drawn in it, and this code is then placed after the *build* command. The *build* command will wire the necessary components together, and set the values of any passive components used.
Device reading commands	
assert mm	When used on the multimeter, *assert* ensures that the last measured value received matches a desired value, within a specific tolerance. For example, *assert mm 5 0.1* will show an error message and cause the test to fail, if the value returned from the multimeter is less than 4.9 or more than 5.1.
assert cro	The oscilloscope *assert* command will perform some calculations on the raw oscilloscope data returned by the server and ensure that the result is correct. For example, *assert cro vpp(1) 2 0.5* will ensure that the peak to peak voltage of channel 1 is 2 +/- 0.5 V. If the peak-to-peak voltage measured by the oscilloscope is less than 1.5 V or greater than 2.5 V, an error message will be shown and the test will fail.
invalidate	The test system will always cache the most recent results returned by the server for the multimeter or oscilloscope. However, after a circuit change, these results may not be automatically updated. The *invalidate* command will discard any cached measurements, and request updated measurements from the NetLab server. Any *assert* commands following an *invalidate* command will be paused until new results are received. If the NetLab server does not send back results within a timeout period, an error will be displayed and any assert dependent on those results will fail.

The commands are combined into a test script, which is run when tests are triggered. Currently, the test script performs these tests:

1. Connects together the function generator, multimeter and oscilloscope.
2. Produces a known voltage on the function generator, and checks for this on the multimeter. Any issues at this stage indicate a fault with the function generator, multimeter or switching matrix.
3. Changes the function generator voltage, and ensures that the changed voltage is present on the multimeter. This ensures that the function generator has not become frozen on any particular voltage, and that the multimeter has the correct result because it measured it, not because it happened to have the correct reading by chance.
4. Checks the oscilloscope signals, and ensures that the oscilloscope is measuring the correct voltages. Any issues at this point, when previous tests passed indicate that the oscilloscope has stopped responding to commands. If issues occur at both this stage and the previous steps, an error with the function generator is likely.
5. Connects each of the four variable resistor boards to the multimeter. For each board, several different resistance values are configured, and the multimeter resistance reading is checked to ensure that each board is responding to commands and does not contain any burnt out resistors. Errors will highlight which board, if any, is faulty. If this test passes, but the earlier tests with the multimeter failed, an error with the function generator is the likely cause.
6. Builds several passive RC or RL filter circuits, each using a resistor board and one of the two capacitor boards or the inductor board. Incorrect voltages returned at this point show that the capacitor or inductor boards are not functioning correctly.

By analysing the output messages of the test system, the source of any issues can be quickly identified. If a component has been physically damaged by a remote user, the detailed server logs can be examined to identify how the component was damaged, and appropriate protections can be put in place to prevent a reoccurrence of the issue. Otherwise, the physical component can be checked to identify the cause of the problem.

CONCLUSIONS

During the previous year, several incidents highlighted the need for improved testing. Incidents have included power interruptions due to building work, software failures due to automatic updates, one physical failure caused by a connector that was poorly crimped eight years earlier, and an oscilloscope not responding due to a race condition in the oscilloscope code of the server. All of these issues would have been detected quickly by the testing system, and rectified far sooner.

An intelligent tutoring system will be implemented in NetLab in the near future. The testing system described in this article will help to maintain the remote laboratory in a healthy state, allowing accurate evaluation of the tutoring system. It will also improve the user experience for those using the remote laboratory.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the Commonwealth Government of Australia funding, received under the Research Training Programme (RTP), supporting the PhD project of one of the co-authors (HC).

REFERENCES

1. Gustavsson, I., Nilsson, K., Zackrisson, J., Garcia-Zubia, J., Hernandez-Jayo, U., Nafalski, A., Nedic, Z., Göl, Ö., Machotka, J., Pettersson, M. I., Lagö, T. and Håkansson, L., On objectives of instructional laboratories, individual assessment, and use of collaborative remote laboratories, *IEEE Trans. on Learning Technologies*, 2, **4**, 263-274 (2009).
2. Considine, H., Nafalski, A. and Nedic, Z., *Remote Laboratory Environments for Smart E-learning*. In: Uskov, V. Howlett, R.J. and Jain, L.C. (Eds), Smart Innovation, Systems and Technologies, 75, Smart Education and E-learning. Cham Springer International Publishing, 82-91 (2017).
3. Nedic, Z, Nafalski, A., Machotka, J. and Göl, Ö., Enriching Student Learning Experiences through International Collaboration in Remote Laboratories. Australian Learning and Teaching Council, Sydney, Australia (2011).
4. Considine, H., Nafalski, A, Nedic N and Zawko, T., Student interactions with the remote laboratory NetLab, *World Trans. on Engng. and Technol. Educ.*, 15, **4**, 344-348 (2017).
5. Garcia-Zubia, J., Cuadros, J., Romero, S., Hernandes-Jayo, U., Orduña, P., Guenaga, M., Gonzalez-Sabate, L., Gustavsson, I., Empirical analysis of the use of the VISIR remote lab in teaching analog electronics, *IEEE Trans. on Educ*., 60, **2**, 149-156 (2017).
6. Baker, R.S., Stupid tutoring systems, intelligent humans. *Inter. J. of Artificial Intell. in Educ.*, 26, **2**, 600-614 (2016).
7. Woolf, B.P., *Building Intelligent Interactive Tutors: Student-centered Strategies for Revolutionizing E-Learning*, Amsterdam, Boston: Morgan Kaufmann Publishers/Elsevier (2009).
8. Considine, H., Nedic, Z. and Nafalski, A., Assisting students in online experimentation. *Proc. 4th Experiment@ Inter. Conf.*, University of Algarve, Faro, Portugal, 47-51 (2017).
9. Zawko, T, Nafalski, A., Considine, H. and Nedic, Z., *Self-testing System Application to Remote Laboratory NetLab*. In: Auer, M.E., Guralnick, D. and Simonics, I. (Eds), Advances in Intelligent Systems and Computing. 716, Springer International Publishing Germany, 648-656 (2018).
10. Teng, M., Considine, H., Nedic, Z. and Nafalski, A., Current and future developments in remote laboratory NetLab. *Inter. J. of Online Engng.*, 2, **8**, 4-12 (2016).
11. Considine, H., Teng, M., Nafalski, A. and Nedic, Z., Recent developments in remote laboratory NetLab. *Global J. of Engng. Educ.*, 18, **1**, 16-21 (2016).
12. Baraniak, J., Pachowicz, K., Nafalski, A., Considine, H. and Nedic, Z., Determination of parameters of an equivalent circuit of a single-phase transformer using a remote laboratory. *World Trans. on Engng. and Technol. Educ.*, 14, **4**, 445-450 (2017).