Robotics & Automation Engineering Journal
ISSN: 2577-2899

Juniper Publishers
key to the Researchers

Review Article
Volume 3 Issue 4 - August 2018
DOI: 10.19080/RAEJ.2018.03.555618

Robot Autom Eng J

Remote Access Laboratory for Delivering a Junior Level Computer Engineering Course

Mansour Tahernezhadi*, Abul KM Azad and Aashish Nimbekar

Moulay Ismail University, Morocco

Submission: June 06, 2018; **Published:** August 06, 2018

***Corresponding author:** Mansour Tahernezhadi, College of Engineering and Engineering Technology, Northern Illinois University, USA, Email: mtaherne@niu.edu

Abstract

This paper presents the design and development of an embedded processor driven online laboratory facility to support a junior level computer engineering course in an Electrical Engineering or a Computer Engineering program. The developed experiments can be accessed remotely over the web using a suitable graphical user interface (GUI). The GUI also provides the remote users with live video feed using an IP Camera-providing them with real time feedback. Also, remote users can upload their own program to test and analyze the interfaces made with the Arduino board. The remote user is also provided with the ability to get real-time feedback and download their data for offline analysis. In addition, the system has a logic analyzer and a switching circuit. The logic analyzer analyzes data flow between two controllers, while the switching circuit can be used to reconfigure hardware connections remotely.

Introduction

Laboratories require the user to directly interact with the equipment for performing certain actions like turning the switch on/off, adjusting the rotary knob, and receiving the audio, visual or tactile feedback. In a remote laboratory, the user has access to control the equipment in the laboratory through a user interface provided by the remote infrastructure. The remote infrastructure bridges the gap between the user and the laboratory equipment [1]. Much has been said and done concerning online experiments in science and engineering. In general terms, online labs comprise a range of cyber-physical resources, including various types of test and measurement equipment. Online labs also complement the theoretical content offered by e-learning platforms. They can provide access to real workbenches supporting web access (remote experiments), to simulation environments (virtual experiments), or to a combination of both (mixed-reality experiments). However, most of the work done in this area was essentially technical in nature and supported by engineering or computer science institutions with a small or non-existent participation by educational sciences. As a result, the literature concerning pedagogical evaluation of online labs is very scarce, and the pro and con arguments are largely based on common sense without supporting evidence from field trials.

Many of the remote laboratories designed so far use a stand-alone computer to control the equipment and a dedicated server to provide access to the users [2]. These laboratory designs have a complex system design, and a lot of hardware and software tools are required to put these systems together. To overcome design complexity, development boards can be adopted to perform the task of the stand-alone computer. The rationale for the Labs-on-the-web project consisted of preparing such trials and gathering evidence to identify the benefits and limitations of offering web access to workbenches used by engineering students. Practical experience is a significant learning tool across all areas of study. Traditionally, practical knowledge is gained in educational laboratories, but over the years, the nature of these traditional laboratories has changed.

Traditional laboratories in engineering are hands-on for practical examination and validation of underlying theories. As such, there is a need for customization to enable access from a remote place over the internet to achieve comparable experiential functionalities as traditional labs. We have three types of laboratories: normal laboratories, virtual laboratories and remote laboratories [3]. Normal laboratories are traditional laboratories in which students perform tasks on original hardware tools, also known as hands-on experience. Virtual laboratories provide software simulations for original experiment set-up. Remote laboratories enable the user to perform a range of scientific experiments over the internet without being near the actual equipment. The main problem in designing remote laboratories is to find an efficient platform for communication between the remote client and the intended equipment [4].

A number of research groups have developed remote laboratories using software tools like LabVIEW and MATLAB in different teaching, research and industrial fields. LabVIEW software is a flexible programming environment that can help in building a remote laboratory [5]. To control the experiment set-up from a remote area using LabVIEW, the control panel should be shared on web browsers. A plug-in is also required to run the experiments [6,7]. The control panel, also called a Graphical User Interface (GUI), contains all the input and output controls of an experiment arranged in an easily understandable way. When a user requests control of the experiment, LabVIEW enables a secure connection between the user and server using TCP/IP Ethernet protocol. While the experiment tools are being used, it will keep incoming requests in queue. The main drawbacks of this system are that LabVIEW is not open source software, it is an expensive tool, and clients should download a plug-in, which is about 400MB.

The other approach to building remote laboratories is using MATLAB combined with Simulink software [8]. In this case, Java programming technology is used for development of an application. The Java program uses a multi-threading approach to improve performance and handle multiple MATLAB sessions at any given time. The JMatLink library enables connection to MATLAB from Java servlets. These can be accessed from the web browser via HTTP. The servlet functionality can open and initialize the MATLAB engine, start all simulations, obtain results and send them to the client application. Of course, finally it enables closing the connection with the MATLAB application. The client application is dynamically generated by the web server. It facilitates user interaction with the remote system. Except for PHP, we can use an AJAX approach that enables asynchronous modification of the web page. Drawbacks of this method are that

A. MATLAB, like LabVIEW, is an expensive tool;

B. it is not an open source program for controlling the experiment and the clients are forced to download java applets/Ajax software tools for their browsers; and

C. MATLAB is no longer supporting remote laboratory tools [8].

In contrast, the primary goal of this project is to design a remote laboratory, including cost effectiveness and ease of access, and to expand the ability of students to engage in hands-on learning. Free and Open source software such as Python and Linux Operating System play a vital role for designing these types of laboratories [9]. In this paper, the problem is segregated into two tasks:

a) establishing a link between experiment set-ups and the server PC and

b) establishing a separate link between remote clients and the server PC.

The server PC plays a vital role for secure connection, speed and queues. This work advances three main contributions:

a. Development of a remote lab platform

b. Development of a Logic (bus) analyzer; and

c. Development of a switching circuit.

This paper is organized as follows. While Section 2 provides coverage of the problem, Section 3 delves into the proposed platform for a remote access laboratory in computer engineering. Section 4 describes the graphical user interface for the proposed platform. Finally, conclusions are presented in Section 5.

Problem Statement

There is a high demand for access to remote laboratories by the students. The idea is to develop a laboratory setup in a University that can be accessed remotely from any location using a computer. The major challenges with laboratories are that they cannot be accessed outside of normal operating hours, not all equipment is available in a single laboratory, and managing access to all students in the laboratory if difficult. These problems can be addressed by implementing a remote laboratory. The main aspect of a remote laboratory is to have two-way communication between the remote user and the laboratory equipment. To accomplish that, an application had to be created to integrate the server side and the user side. A solution is to use a freely available general-purpose programming language like Python. Additional tools like JavaScript and HTML (Hyper Text Markup Language) make the implementation much easier. Providing a GUI facilitates communication between the user and the equipment and addresses the problem of access. Using these open-source tools makes it cost effective and more efficient. The proposed system provides a GUI over the web to access the remote laboratory. The GUI was developed using open source tools like HTML, CSS, and JavaScript.

One of the major challenges with traditional laboratories is that they cannot be accessed outside of normal operating hours due to management and safely issues, thereby depriving the intended users from much needed flexibility in terms of time and space. This problem is further exacerbated by inefficient utilization of the lab, resulting in equipment lying idle most of its working life. This problem can be addressed by introducing online laboratories in which one can access the laboratory facility over the web to perform laboratory experiments, while the experimental system (hardware or software) can be placed to a single or distributed location. There is a high demand for online laboratories to support engineering and engineering technology courses.

Figure 1 shows the generic concept of an online laboratory in which the experimental setup and the local computer server

How to cite this article: Mansour Tahernezhadi, Abul KM Azad, Aashish Nimbekar. Remote Access Laboratory for Delivering a Junior Level Computer Engineering Course. Robot Autom Eng J. 2018; 3(4): 555618. DOI: 10.19080/RAEJ.2018.03.555618.

are located within the laboratory and remote users gain access to the facility over the Internet. Online laboratories can be classified into three distinct categories: remote, virtual, and a combination of the two. A remote laboratory offers remote access to an experimental facility involving real hardware hosted within a campus laboratory. A virtual laboratory provides a simulated experimental system that may be hosted either on a server(s) or in the cloud. A mixed reality laboratory offers a combination of real hardware and simulated experimental system, sometimes known as hardware-in-the-loop.

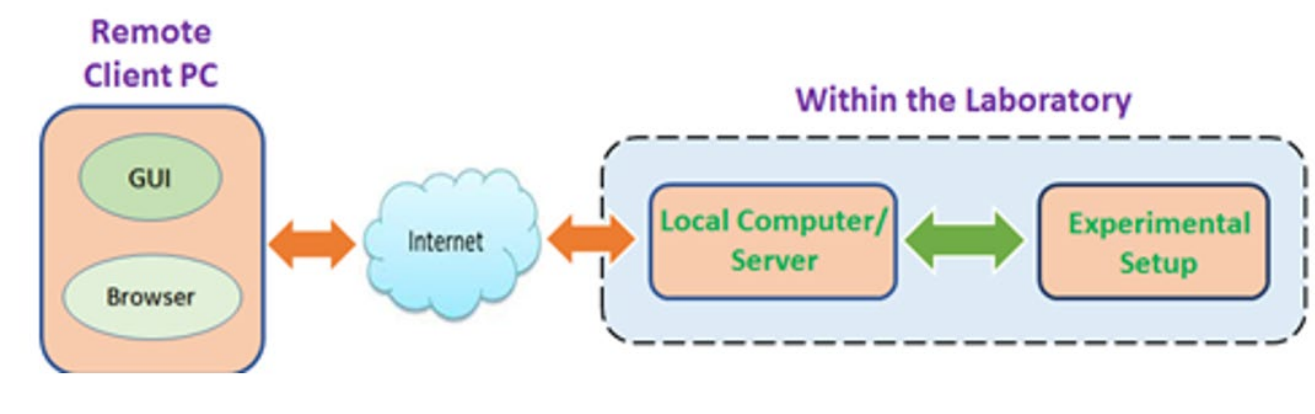

Figure 1: Generic concept of a remote laboratory.

The main aspect of a remote laboratory is to have two-way communication between the remote user and the laboratory equipment. Many online laboratories designed to date use a computer for equipment control and a dedicated server to provide access to the remote users [10]. This approach uses a complex system design and involves several hardware and software tools that can be eliminated. To overcome design complexity, embedded processor boards can be adopted to perform the task of a computer. On the server side, applications can be created by utilizing open source software like Python, JavaScript, and HTML. The purpose of the current work is to develop an online laboratory facility that will support a junior level computer engineering course.

Developed Platform

The primary goal of the proposed platform is the design and development of a remote laboratory system that will be cost effective, easy to access, and engaging for students. The target course involves teaching embedded processor programming utilizing an Arduino board [11]. Open source software plays a vital role in this development. The system can be divided into three areas: experiments needed for the course, a logic analyzer, and a switching circuit. Figure 2 shows the main components of the proposed platform.

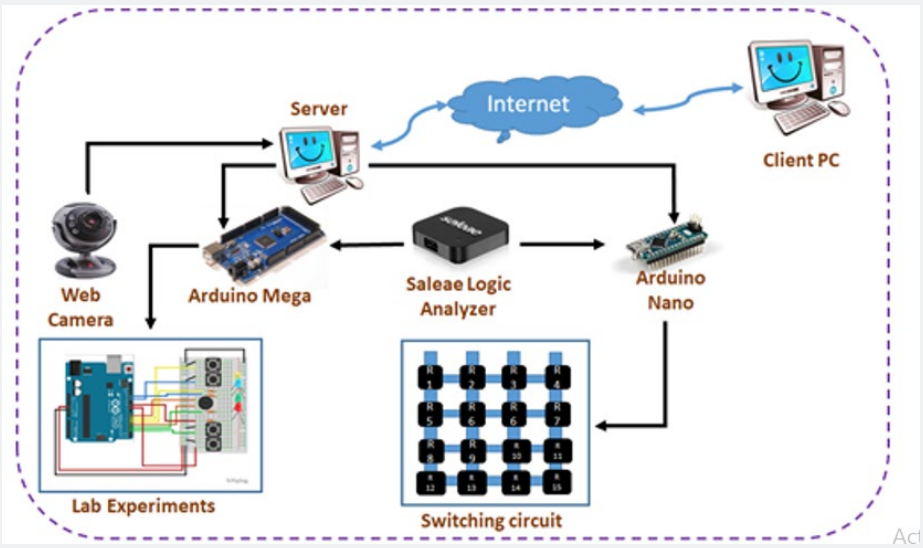

Figure 2: Showing main components of the designed system.

How to cite this article: Mansour Tahernezhadi, Abul KM Azad, Aashish Nimbekar. Remote Access Laboratory for Delivering a Junior Level Computer Engineering Course. Robot Autom Eng J. 2018; 3(4): 555618. DOI: 10.19080/RAEJ.2018.03.555618.

The system is comprised of a server, an Arduino Mega board, a Saleae logic analyzer, an Arduino Nano board, and a webcam. The server, hosting the GUIs, is the gateway to the outside world, and runs with the Linux operating system. The Arduino Mega board is the main control hardware and communicates with the laboratory experiment setup, a logic analyzer, and a switching circuit [12]. The webcam is directly connected to the server to provide real-time video of the experimental facility. It also passes the audio from the experiment, so the remote user can hear the sound. This allows the users to have visual and audio feedback from the experimental system.

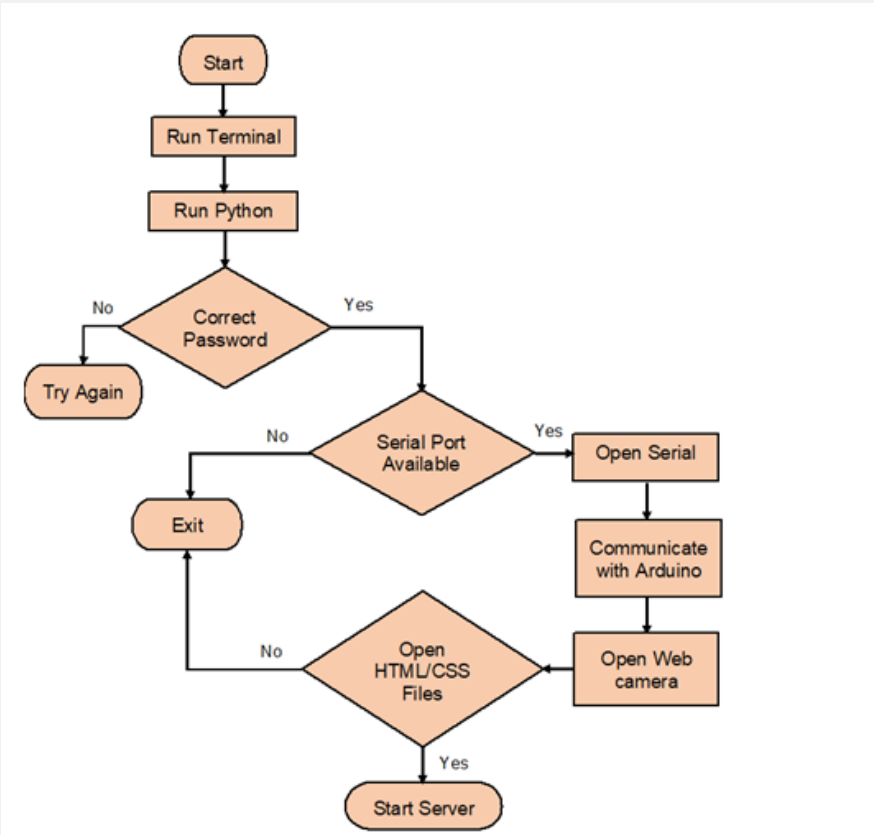

Figure 3: Software interaction diagram within the server.

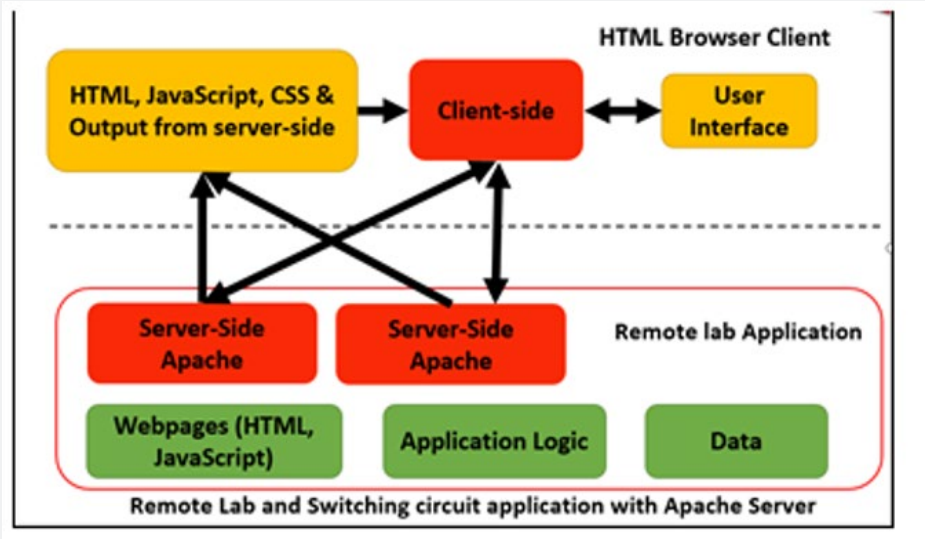

Figure 4: Server startup flowchart.

How to cite this article: Mansour Tahernezhadi, Abul KM Azad, Aashish Nimbekar. Remote Access Laboratory for Delivering a Junior Level Computer Engineering Course. Robot Autom Eng J. 2018; 3(4): 555618. DOI: 10.19080/RAEJ.2018.03.555618.

Figure 3 shows the program data flow in the server system. The server is connected to the experimental set-up via a serial cable for bi-directional communication. The server provides access to the remote users over the Internet, whereby the remote user can upload the program over the Internet and interact with the experimental setup to perform laboratory exercises. The server utilizes a Linux operating system, while Python is used for developing remote user applications [13]. When a user uploads a program to the server, it handles the incoming files and stores them in its database. Subsequently the files are transferred to the Arduino board and are executed to perform the desired task involving sensors and actuators. A Python script is utilized to develop a file management system. It also deals with the file names within the server. With this arrangement there is no need for any plug-ins in the remote user's system. The only thing needed is an Internet connection and a web browser since a starting Python takes all the modules required for the operation using an import method. It loads all the files before starting the server, and whenever a request is made, the server script utilizes the HTML/CSS files in order to transfer the files to the client's browser as a webpage. It will then exit and shows the error type messages if anything is wrong in the procedure. Figure 4 shows the server startup flowchart. After initialization, it requests login credentials. Upon successful login, it opens a serial port for communicating with the Arduino board as well as opens the web camera [14]. At this point, the remote user navigates to an HTML page that contains all the GUI controls for the experiments.

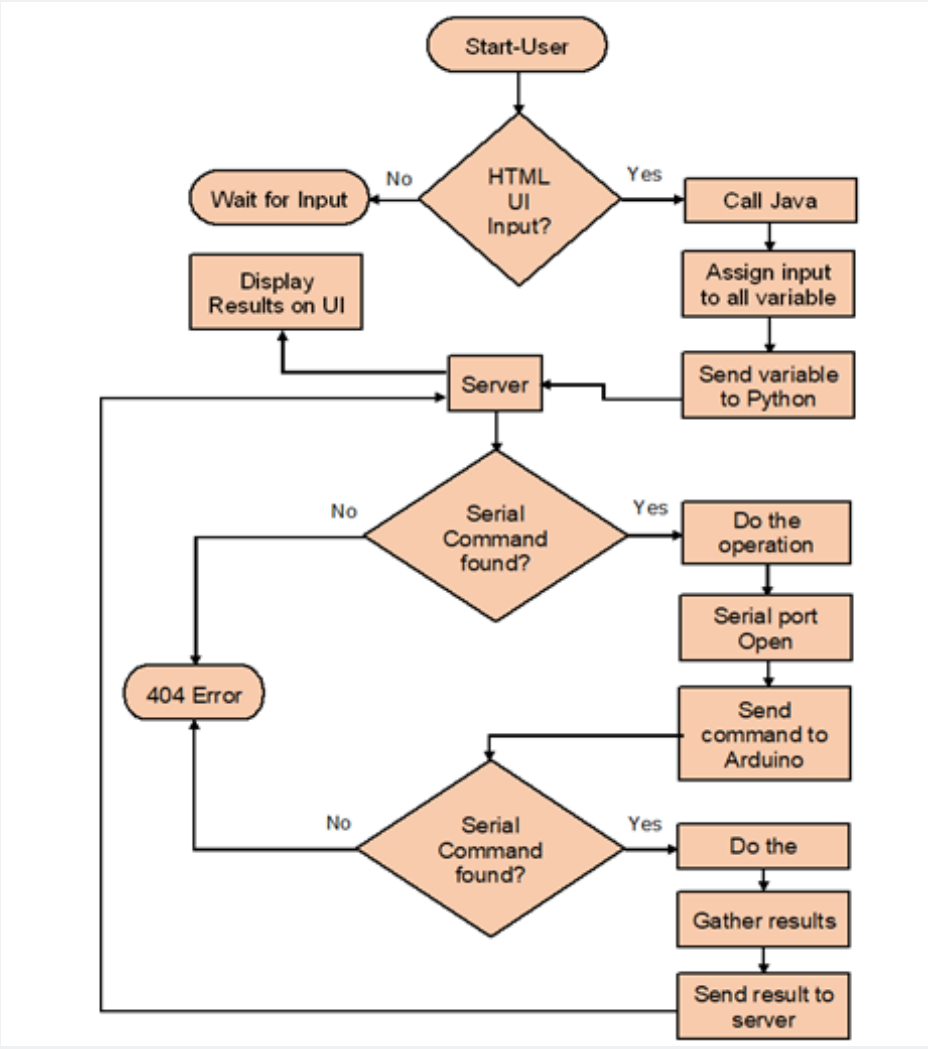

Figure 5: Program dataflow between client and server.

The program dataflow between the client and the server is shown in Figure 5. The server establishes a new connection whenever a remote user tries to connect using the web address. After getting input from the remote user, the HTML code calls the Java Script written in JQuery. JQuery assigns the input request to a variable. The assigned variable is passed to the Python server and the Python script compares the client's input command with the pre-defined commands, and if they are equal, then Python sends the appropriate pre-defined characters to the Arduino board [15]. If the input command does not match the pre-defined commands, then Python raises an error on the client's web page as "404 not found."

How to cite this article: Mansour Tahernezhadi, Abul KM Azad, Aashish Nimbekar. Remote Access Laboratory for Delivering a Junior Level Computer Engineering Course. Robot Autom Eng J. 2018; 3(4): 555618. DOI: 10.19080/RAEJ.2018.03.555618.

The procedure of remote flash can be divided into two parts: handling and storing the uploaded file in the server database and uploading that file to the Arduino Mega [16]. When users try to upload their own program to experiment on a remote flash, the server should transfer it with the file name, format and size of incoming file. If all of the requirements are satisfied, then the remote flash should store the incoming file in the database for next steps. To upload a file, the user must have an upload option in the GUI. The upload procedure can be a Hypertext Preprocessor (PHP) or a HTML based on the designer's knowledge. The steps to design an upload page using HTML are described below (Figure 5a):

```
<form enctype="multipart/form-data" action="/upload" method="post">
File: <input type="file" name="file1" />
<input type="submit" value="upload" </form>
```

Figure 5a

The security of incoming files is an important issue when dealing with the internet. HTML offers different encryption methods to solve this concern. The browse button allows the user to choose a file from the user's system. When the user presses the Upload button, the next step is to check the file format and name of incoming file. The incoming file name should be the same as in the Python script. However, users may not rename their files, so users need to process the incoming file in Python to change its name and to check the file format. Arduino only takes '. ino' formatted files in its flash process. Once the incoming file is processed, it should be stored in a particular folder by replacing the previously uploaded file (Figure 5b).

```
file1 = self.request.files['file1'][0]
original_fname = file1['filename']
extension = ".ino"
fname = "blink"
final_filename= fname+extension
output_file = open("/home/azad /" + final_filename, 'w')
output_file.write(file1['body'])
```

Figure 5b

AVRDUDE is one of the compiler tools available that helps to flash a file into an Arduino board. There are different Arduino boards available. While flashing the recently stored file into the Arduino board, the specifications should be declared. AVRDUDE checks these specifications and the port number in a text file named 'Makefile'. For every flash user need to create a Makefile in the same folder in which the user's file is stored. The Makefile should appear as shown below (Figure 5c).

```
BOARD_TAG = Mega
ARDUINO_PORT = /dev/ttyACM0
ARDUINO_LIBS =
ARDUINO_DIR = /usr/share/arduino
include ../Arduino.mk
```

Figure 5c

By sending simple commands like 'make' and 'make upload' to the AVRDUDE from Python starts the compiling of the recent program in the specified folder. An 'OS' module allows Python to manipulate the operating system features. The compiler returns errors if there are any syntax errors; if not, it uploads the file to the Arduino board. The success or failure messages should be displayed for the user. To do that users have to get the terminal procedure lines while the file is executing and save that as a text file. This will be the easiest way of showing error messages on the client's browser. The last line of the code below does the same job; it takes all the lines of the terminal window into an outputfile.txt and saves it in a specific memory path of the server. The outputfile.txt contains all error messages and successful messages. The Python Tornado server can directly read the lines from the file and writes the data on the web page [14] (Figure 5d). The 'output' text file is further filtered and processed to remove unwanted lines. Python writes a 'Success' message if there are no errors, otherwise it will display an error line, including the error category and line number in the user's program.

How to cite this article: Mansour Tahernezhadi, Abul KM Azad, Aashish Nimbekar. Remote Access Laboratory for Delivering a Junior Level Computer Engineering Course. Robot Autom Eng J. 2018; 3(4): 555618. DOI: 10.19080/RAEJ.2018.03.555618.

```
os.system("make clean")
os.system("make")
os.system('sudo make upload 2>&1 | tee ~/outputfile.txt')
```

Figure 5d

Graphical User Interface

A user-friendly GUI is an important factor for an engaging online laboratory. With this in mind, the facility includes a number of GUIs that are used to interact with the remote users. Figure 6 shows the GUI that is used for performing laboratory experiments by remote clients. There are three major activities one can perform through this GUI. The first functionality involves the test programs that are provided within the facility. The second functionality is the remote user program where the users can upload their own program to the Arduino board and verify the correctness of the program. The third functionality will help the users to execute the uploaded program. There is a total of eight test programs provided for remote users (five for individual laboratory experiments and three for projects). To run a test program, a remote user needs to click on a button under the 'Test Codes' area and observe the execution. In terms of uploading a user file, a remote user needs to click on the 'Select file' button and that will open an upload window. The user then selects the desired program from the system and clicks on the 'Upload' window. The user program then uploads to the remote Arduino board. The program is then executed, and the user can verify the correctness of the program.

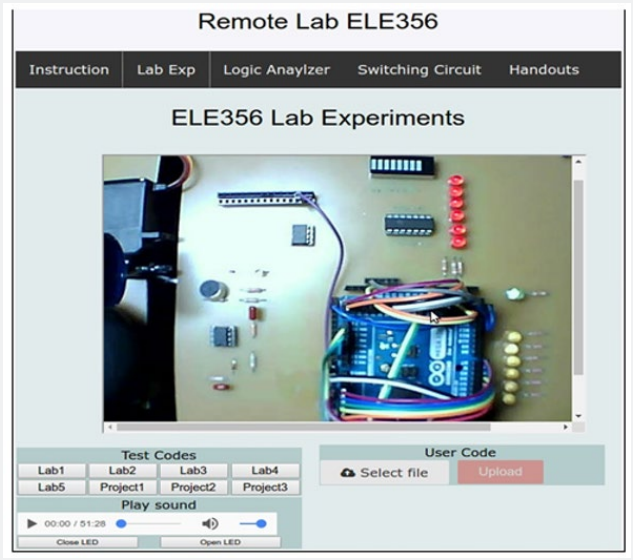

Figure 6: GUI for lab experiment.

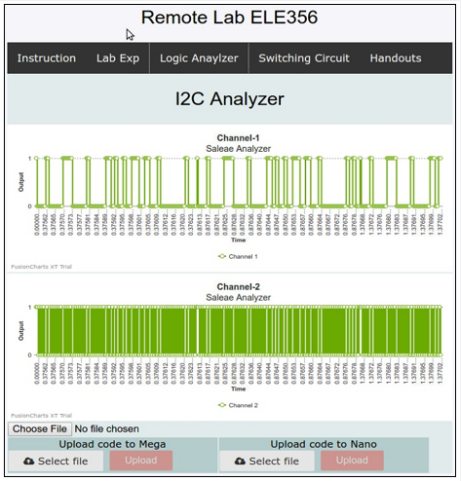

Figure 7: GUI for I2C bus analyzer.

How to cite this article: Mansour Tahernezhadi, Abul KM Azad, Aashish Nimbekar. Remote Access Laboratory for Delivering a Junior Level Computer Engineering Course. Robot Autom Eng J. 2018; 3(4): 555618. DOI: 10.19080/RAEJ.2018.03.555618.

As shown in Figure 2, two controllers are utilized with the developed system (an Arduino Mega and an Arduino Nano). These are connected over an I2C bus. A Saleae's logic analyzer monitors the I2C bus traffic between the controllers. The GUI developed for this monitoring is shown in Figure 7. Within this GUI, a remote user can upload a program to Arduino Mega or Arduino Nano. One can also upload a program to Arduino Mega or to Arduino Nano from the bottom window. Remote users can also decide which controller they would like to use as the master and the slave.

A switching circuit was developed to allow the remote user to reconfigure a circuit connection without any physical intervention [17]. As depicted in Figure 8, a relay switching module is organized as a matrix that can be used to modify a circuit configuration. A bank of relays is used to create a 4X4 relay matrix. A number of capacitors and digital potentiometers are connected between the relays. A look up table shows the value for a given potentiometer. The switching circuit consist of 16 relays, seven capacitors, two digital potentiometers, and two shift registers. The two shift registers (74HC595) can be configured as an 8-bit serial-in or parallel-out with output latches. In other words, one can use the shift registers to control eight outputs at a time, while engaging a few pins of the microcontroller.

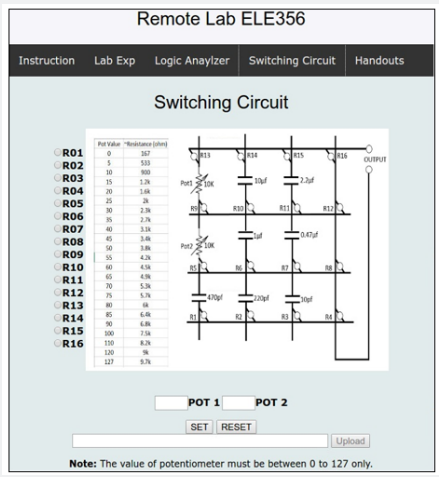

Figure 8: GUI for switching circuit.

The following describes the operation of the switching circuit. If a remote user wants to use a 10uf capacitor, then relays R14, R10 and R12 should be ON to get the output at the output pins. If a remote user wants to use digital potentiometer Pot1, then relays R13, R9 and R12 should be ON and the user must put a value in the Pot1 box to set a value (range 1-127). Each value corresponds to the resistance amount provided within the look-up table. The user can make series and parallel connections by selecting the proper relay combination. As an example, if a remote user wants to use potentiometer Pot1 and Pot2 in series, then relays R13, R5, and R8 should be ON. If a user wants to connect a 2.2uF capacitor in parallel with the Pot1 potentiometer, then relays R15, R13, R9, R11 and R12 should be ON and the user should put the resistance value in the Pot1 box. Similarly, the user can try multiple combinations of capacitor and resistance using the proper relay combination.

Conclusion and Future Work

This paper illustrated the design and development of an online remote laboratory facility that can be used for delivering a laboratory course for a junior level computer engineering course. There are a number of available tools with the advent of evolving technologies. This project mainly used open source software, so the cost is reduced and the design complexity is minimized compared with traditional systems. The proposed system utilizes an Arduino Mega board that acts as an embedded processor to allow students to run and test their code from a remote location. A GUI was also designed such that each lab can be monitored or controlled individually, whereby students can remotely run all the experiments. An IP camera was also integrated into the system to provide live streaming of the system over all the GUIs. The second part of the project involved integration of two apache servers for two Arduino boards (Mega and Nano). This enabled the users to initiate and monitor the I2C bus traffic between the two controllers. Specifically, the users can program any of the boards as a master or a slave and monitor the I2c bus traffic on the GUI. The GUI also provides the ability to save data files for post offline analysis. Last but not least, with the help of Arduino boards and shift register ICs, the user can select and turn on different combinations of relays to choose and activate different components in a given circuit. All in all, the proposed remote laboratory platform serves as an effective experiential platform for students to learn about Arduino programming, its associated hardware experiments, and Python programming from anywhere and at any time.

How to cite this article: Mansour Tahernezhadi, Abul KM Azad, Aashish Nimbekar. Remote Access Laboratory for Delivering a Junior Level Computer Engineering Course. Robot Autom Eng J. 2018; 3(4): 555618. DOI: 10.19080/RAEJ.2018.03.555618.

References

1. (2018) About Remote Laboratory.
2. Gravier J, Fayolle B, Bayard MA, Lardan J (2008) State of the Art About Remote Laboratories Paradigms- Foundations of Ongoing Mutations, International Journal of Online Engineering 4(1): 19-25.
3. William BY (2013) Remote Laboratory Approaches.
4. Hashemian R, Riddley J (2007) A Method to Design, Construct and Test Digital Hardware all in Classroom Environment. In Proc 37th Annual Frontiers in Education Conference, FIE2007 Conference, Milwaukee, WI, USA, pp. 11-13.
5. (2015) LabVIEW System Design Software.
6. Zubia JG, Alves GR (2011) Using Remote labs in Education, University of Deusto, Bilbao, Spain, pp. 1-455.
7. Azad AKM, Pramod PK (2014) Developing Control Experiments as a part of a Remote Laboratory Facility, ASEE Annual Conference & Exposition. Indianapolis, USA.
8. Esche SK, Chassapis C (2012) On Infrastructures for Educational Online Laboratories; In: Azad AKM (Ed.), Internet Accessible Remote Laboratories: Scalable E-Learning Tools for Engineering and Science Disciplines, IGI Global, PA, USA.
9. Vakati S (2012) Design of remote laboratory application using Python, M.S. Thesis, Department of Electrical Engineering, Northern Illinois University, USA.
10. Azad AKM, Sharma MS (2013) Internet Accessible Remote Experimentation with Integrated Learning Management System. ASEE Annual Conference. Atlanta, USA.
11. Arduino M (2018) Arduino Mega.
12. (2016) The Arduino Mega 2560 is a microcontroller board.
13. Python (2015) Python Software Foundation.
14. Tornado (2015) Tornado Stable.
15. Boldt E (2013) Python Web UI with Tornado.
16. Pratyaksa D (2013) Programming and uploading Arduino sketch without IDE.
17. Nilsson K (2014) Development and Evaluation of Openlabs and the VISIR Open Electronics and Radio Signal Laboratory for Education Purpose, Blekinge Institute of Technology, Sweden.

DOI: 10.19080/RAEJ.2018.03.555618

How to cite this article: Mansour Tahernezhadi, Abul KM Azad, Aashish Nimbekar. Remote Access Laboratory for Delivering a Junior Level Computer Engineering Course. Robot Autom Eng J. 2018; 3(4): 555618. DOI: 10.19080/RAEJ.2018.03.555618.