New Orleans JAZZED about Engineering Education

Paper ID #15869

Design and Development of Remote Testbeds Using Python

Prof. Abul K. M. Azad, Northern Illinois University

Abul K. M. Azad is a Professor in the Technology Department of Northern Illinois University. He has a Ph.D. in Control and Systems Engineering and M.Sc. and B.Sc. in Electronics Engineering. His research interests include remote laboratories, mechatronic systems, mobile robotics, and educational research. In these areas, Dr. Azad has over 100 refereed journal and conference papers, edited books, and book chapters. So far, he has attracted around $1.7 million in research and development grants from various national and international funding agencies. He serves on editorial boards of a number of professional journals and is Editor-in-Chief of the International Journal of Online Engineering. He is active with various professional organizations (IEEE, IET, ASEE, and ISA) as well as a member of board of Trustees of CLAWAR Association.

Dr. Reza Hashemian P.E., Northern Illinois University
Mr. Suresh Vakati

Design and Development of Remote Testbeds using Python

Abstract

With the emergence of the Internet of Things (IoT), the development of remote testbeds are gaining momentum with an intention to use them for teaching and for laboratory activities. Remote testbeds allow one to perform experiments on a real hardware over the Internet from a remote location. There are a number of software packages used in the design and development of remote testbeds. This paper will describe the use of Python for such a development. To demonstrate Python's effectiveness, the paper will describe two case studies. One of them is a remote vacuum cleaner and the other is an embedded processor system with remote programming capability.

1. Introduction

It is vital to provide laboratory activities to maximize learning in STEM disciplines. Traditionally, students perform experiments by being present in a laboratory and working with physical systems. However, when considering the financial involvement, manageability, and accessibility, this arrangement is not always effective. With the emerging technologies in computing hardware and software, researchers and academics are leaning toward Internet accessible remote testbeds to replace, or supplement, existing laboratory experiments.[1] Remote testbeds can maximize utilization time, increase collaboration among universities and research centers, and provide access to expensive experimental resources.[2]

Researchers are using various hardware and software technologies for remote testbed developments.[3,4] Thinking broadly, a remote testbed system can be divided into a few major components: the experimental system, interface of the experiment with a local computer, graphical user interface (GUI), and server for remote access and access management. A number of software tools can be used for interfacing, GUI development and server applications. Some of these tools are LabVIEW, Matlab, and .NET. These are expensive proprietary software, and the first two have some limitations in terms of flexibility in development and browser adaptability.

With these in mind, this paper will report the design and development of experimental testbeds using Python, in which a single software can provide computer interfacing, GUI development, and remote access. This paper will describe the development of two remote testbeds using Python. The testbeds are a mobile platform with self-navigation and an embedded system with remote programming capability.

The first testbed is a self-navigated mobile platform fitted with a vacuum cleaner. The mobile platform is wirelessly connected to a server for operation and control. The system is fitted with a number of sensors to implement self-navigation around obstacles and has an onboard microcontroller system for control. A GUI was developed to facilitate the interaction between the system and a remote user. Within the second testbed, an embedded processor (Arduino board) was interfaced with a number of output devices (liquid crystal display, seven segment display, light emitting diodes, and a stepper motor). Using a GUI, remote clients can run a few

pre-developed programs provided within the system. They can also upload their own program to the embedded processor board. The testbed has the capability to compile an uploaded program and send the compilation report to the remote user. If needed, the remote user will then debug the program and upload again. Finally, the user can verify the program implementation through a video feedback provided within the testbed.

2. Remote Laboratory Technologies

To explore technologies involved in the development, Figure 1 shows a conceptual structure of a remote testbed. The main components can be identified as the experimental setup, local computer/server, Internet cloud, and remote clients. The experimental system is connected with a local computer/server, which plays the role of a gateway between the experiment and the remote computer for clients. There should be some middleware that facilitates the information exchange between the local and remote computers.

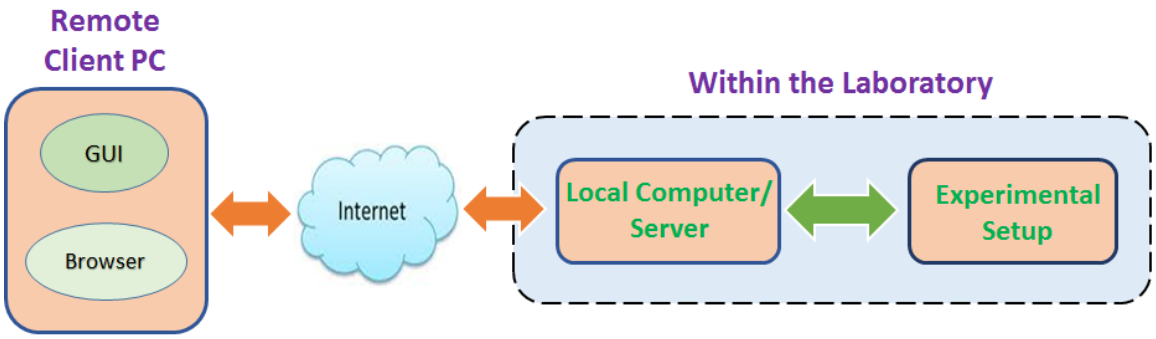


Figure 1: Concept diagram of a remote testbed.

From a technical point of view any experimental system that can be interfaced with a computer via a transducer (sensors and actuators) is a potential candidate for a remote testbed. These can be electrical, mechanical, industrial, chemical, and even biological systems. Of course given the nature of an experiment, appropriate measures should be taken to ensure the safety and integrity of the system and the environment around it. Within the given structure, the local computer/server gathers all the system input, processes those to produce commands for the desired output, and passes them to the experiment to drive the actuators. In almost all of the cases, developers provide a GUI to facilitate a client-friendly interaction so someone with very little technical knowledge can operate the experiment. Connections between an experiment and a local computer/server can be wired or wireless depending on the technology used and the nature of the experiment. The next phase is to make the GUI available over the Internet so clients can have access to the GUI from a remote computer.

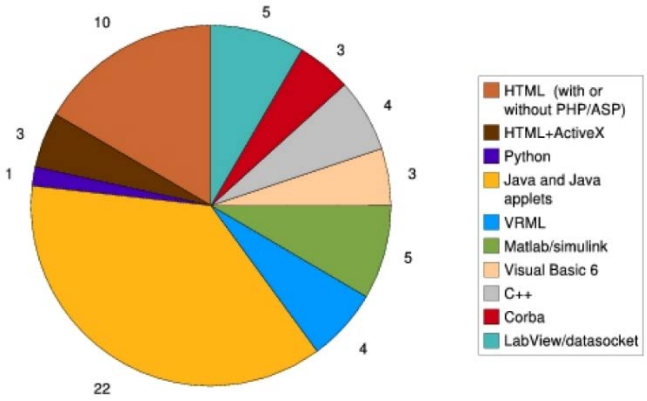


Figure 2: Distribution for software usage for remote testbed developments (Gravier [5]).

A number of software technologies are being used to develop remote laboratories. Gravier and his coworkers reviewed approximately 60 reported initiatives and identified a breakdown of the software used for remote laboratory implementation.[5] A pie chart showing the breakdown of the use of software packages is provided in Figure 2.

3. Python for Remote Laboratory

One of the goals of this development was to use an open source software package for interfacing, GUI, and middleware development. With this in mind, Python was selected as a candidate for a number of reasons. Python is one of the scripting languages available on the market for software development particularly suited to Internet applications. As Python software is an open source tool, it has a wide variety of modules, exceptions, high-level dynamic data types, classes and interfaces to huge libraries. Python code tends to be more compact and more readable than MATLAB code and LabVIEW graphical programs, as Python determines the scope of a block based on its indentation. Python incorporates dictionaries, which are like arrays, except that the index is not an integer. Dictionaries are very popular in engineering and scientific programming because a dictionary is an efficient way to implement symbol tables. Python has huge number of applications, and the Python Package Index has thousands of third party modules for Python.[6] For Web development, Python offers many choices like Django, Pyramid, Flask, Bottle, Plone, and djangoCMS. Python is also widely used in scientific engineering and numeric computing. It has SciPy, Pandas, and Ipython modules for mathematics, science and engineering fields. Python is also used in designing desktops GUIs and for software development.

4. Case Studies

This section will illustrate two case studies of developing remote testbeds using Python. These are a remote vacuum cleaner and remote programming of an embedded processor. The remote vacuum cleaner has an upright mechanical system fitted with sensors and actuators for motion control. The sensors and actuators are connected with a local computer/server via a wireless link. The control mechanisms are implemented within the local computer/server, and all the user controls are provided within a GUI. The second system is an embedded processor system and can be programmed remotely for driving a number of output devices. This system is used for classroom activities in an embedded processor course. As the processor, an Arduino board is

used for this implementation along with an LCD, LEDs, a seven segment display, and stepper motor as output devices. A remote user can upload a program and can validate its performance from a remote location. The system is also designed to debug the uploaded program.

4.1 Remote Vacuum Cleaner

The remote vacuum cleaner consists of a mobile platform fitted with a dc vacuum cleaner along with a number of sensors for navigation. A general block diagram of the system is shown in Figure 3.

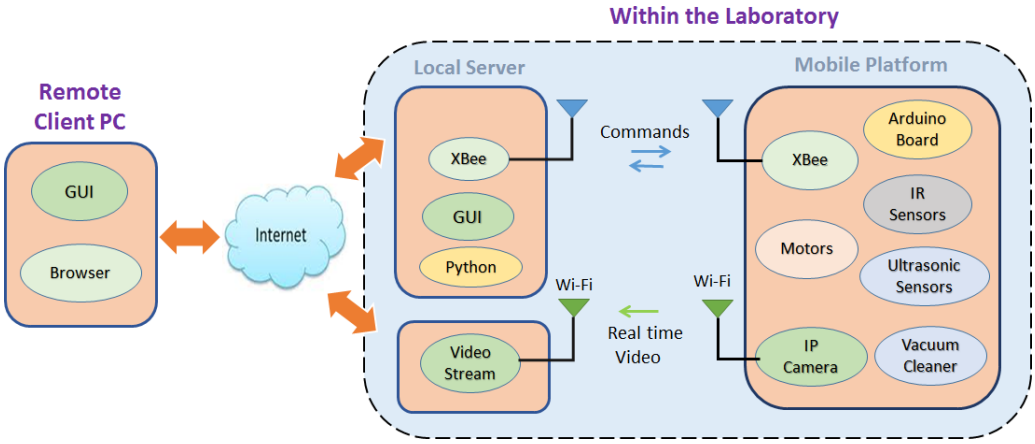
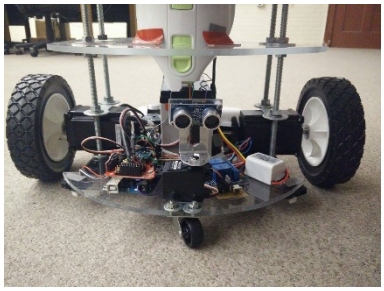


Figure 3: System diagram of the remote vacuum cleaner.

The system has three main components: a mobile platform as the remote testbed system, a local server is the gateway between the testbed and remote clients and the remote client. The mobile platform consists of a drive system, sensors for navigation, an embedded processor (Arduino board) for local control and data management, an XBee for wireless communication with the local server, and an IP camera for real time video. The IP camera has its own communication route via a WiFi channel. The video is then embedded within the GUI for user monitoring. Images of completed mobile platform are shown in Figure 4.

(a) Overall system.



(b) Close up view of electronics.

Figure 4: Images of the mobile platform.

Designing a GUI to provide users with excellent visual composition is a vital part of remote testbed designs. The goal is to improvise and enhance the visual experience between the human eye and computer. Considering the issues for an effective GUI, HTML was used as a software tool for testbed GUI development. Along with HTML, Cascaded Style Sheets (CSS) were provided to improve the overall visual experience. The GUI contains all the input and output selection buttons and variables, which can be adjusted by a client. Figure 5 shows an image of the developed GUI showing all the functionalities. There is an on/off switch provided at the top right corner of the page. Just to the right side of this, there is a mode selection switch. The system can be operated in two modes: auto and manual. On the top right corner there is a vacuum on/off switch for that can activate and deactivate the vacuum cleaner. In the manual mode one can use the arrows for motion control and the red square button to stop. In the auto mode the system travels on its own guided by a set of IR sensors for wall detection and a pair of ultrasonic sensors for obstacle detection. In addition, users can adjust the speed and allow proximity to an obstacle. On the mid left, there is an arrow to determine the distance from the nearest obstacle at any point. The distance will display at the bottom. At the bottom right, the safety status is displayed to show the danger from a nearby obstacle.

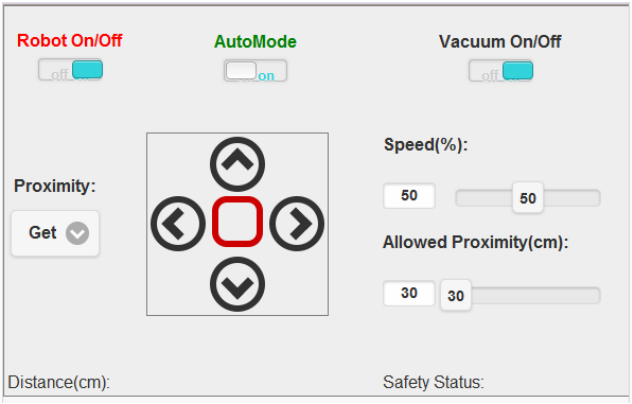


Figure 5: Image of the GUI used for the remote vacuum cleaner.

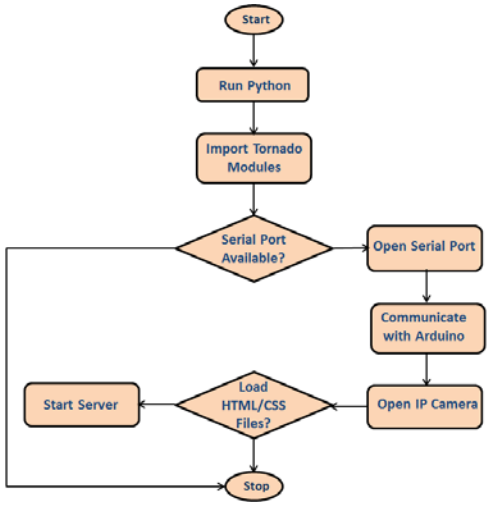


Figure 6: Program flow for the server.

Figure 6 is a flowchart showing a high level view of the program execution cycle for the server system. To start the server, the Tornado Python script should be executed.[7] At the beginning, Python takes in all the modules required for operation using import method. It intiates the opening of a serial port. There will be an error message if the serial port is already being used or is not avaialble. The server communicates with the Arduino board once it opens the serial port. The server script requires all of the HTML/CSS files when a request is made from the client side. The files are loaded before starting the server, and whenever a request is made, it will transfer the files to the client's browser as a webpage. In case of any irregularity in the procedure, it will exit the loop and will display an error message.

When a client enters the URL address on a browser, this directs the user to an HTML page that contains all of the GUI controls of the experiment. Clients can choose different inputs from the GUI, and based on the inputs, the server responds to the user. The program dataflow between the client and the server is shown in Figure 7. The server establishes a new connection when a user enters the URL. It waits until the user sends some input data. After receiving an input from the 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. If they match, Python then sends the appropriate pre-defined characters to the Arduino board. If the input command does not match the pre-defined commands, then the Python raises an error on the client's web page as "404 not found."[8]

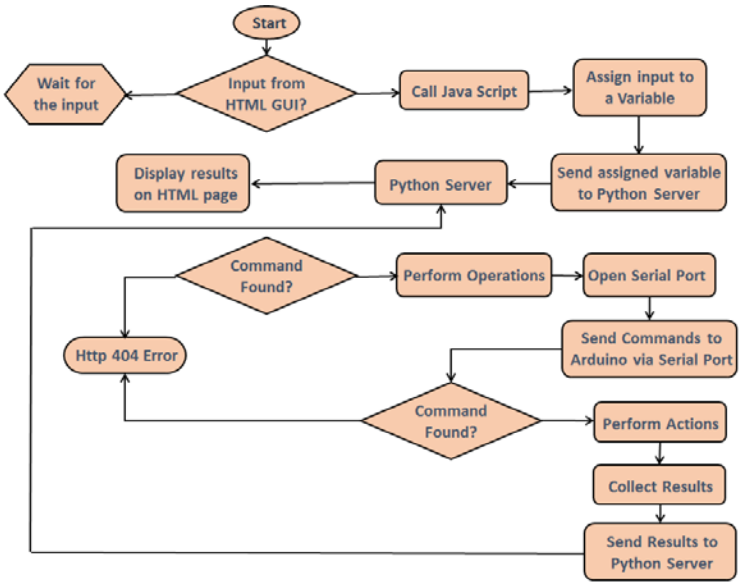


Figure 7: Program flow for handling a client request.

4.2 Remote Programming of Embedded Systems

The second case study describes the development and operation of an embedded system with remote programming capability. With the growth of the Internet of Things (IoT), embedded processors are becoming an integral part of electronic systems. Considering this, almost all engineering and technology programs offer one or two courses on embedded processors at

various levels. Within a given hardware system, embedded processors can be programmed to perform various activities and clients can explore them to design and verify different ideas and concepts.[8] Remote programming capability for an embedded system opens a new horizon for clients in which they can use the facility 24/7 and enhance the learning process.

This section provides the development process and describes the features of an embedded system with remote programming capability. An Arduino board was used for this development along with a number of output devices. The output devices are an LCD, LEDs, a stepper motor, and a seven segment display. The same as in the previous test case, Python was used for interfacing, GUI development, and web services. A block diagram of the developed system is provided in Figure 8.

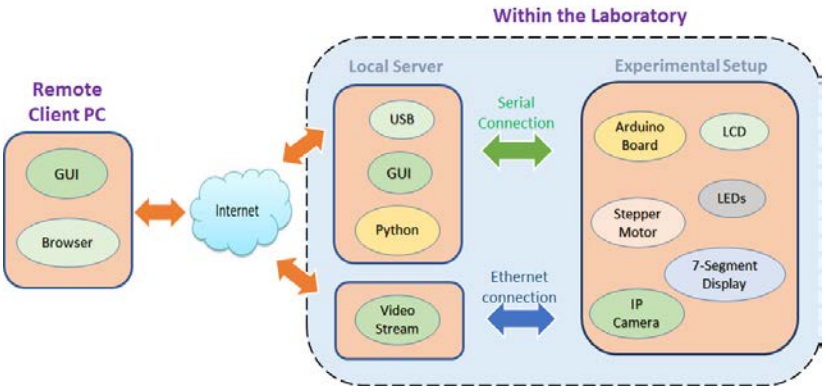


Figure 8: Program flow for handling a client request.

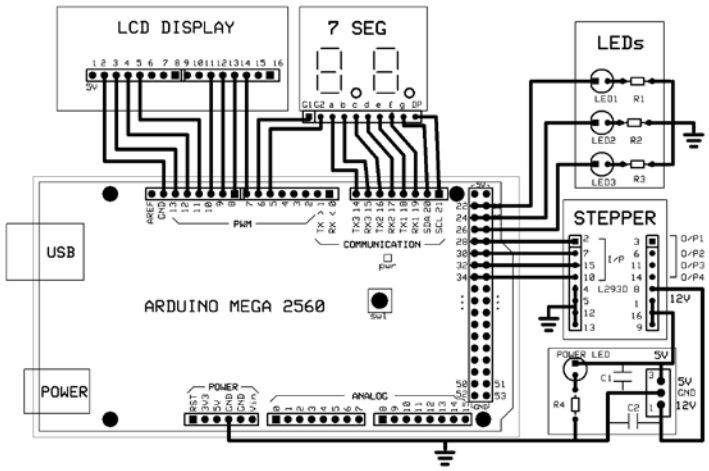


Figure 9: Hardware connection diagram for the remote programming system.

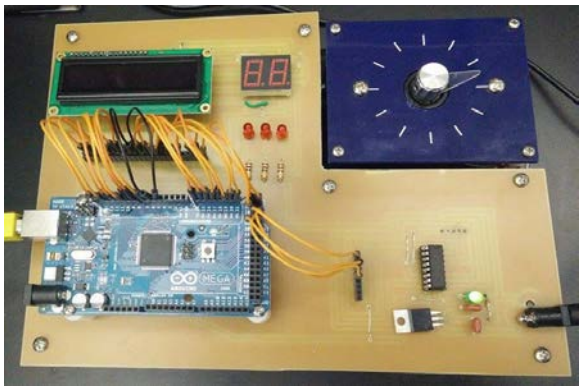
The main idea of this project was to design an embedded processor system that could be programmable from remote locations. An Arduino Mega-2560 microcontroller board was used as the embedded processor.[10] All of the output devices were connected via the Arduino board.

Figure 9 shows a schematic diagram of the hardware system, while Figure 10 shows images of the completed system.

Serial communication was established between the server and experiment set-up to achieve remote programming. When a user uploads a program to the server, it handles the incoming files and stores them in its database to process and then transfer to the processor board. A Python script helps to handle the files and their names in the server. With this arrangement, the remote user does not need any plug-ins to operate the remote testbed. The clients' PC just needs an Internet connection and a latest version browser like Firefox, Opera, Chrome, or Safari.

The server is a computer that handles all incoming clients and responds to the users for their inputs. This experiment mainly deals with uploading files, so the CPU must be 2.66 GHz 128 cache or above, the RAM should be 2GB or above, and the minimum database space should be 10GB. The server is installed with an Ubuntu 14.10 version operating system rather than Windows, which allows easy access to the installed software tools from the terminal. Python is an open source software that provides a variety of modules to develop server applications. A Python script using a Tornado module can turn any computer into a server, and this feature was utilized for this development.[7]

(a) IP camera is directed to the experiment.



(b) Image of the developed experiment.

Figure10: Pictures of the final designed system.

The process for the remote programming of the Arduino can be divided into two parts: handling and storing the uploaded file in the server database and transferring the file to the Arduino board.[11] The upload request through a GUI can be executed with a PHP or HTML procedure on the server side. In this project, an HTML approach was implemented. The lines of code used for this implementation are shown below:

```
<form enctype="multipart/form-data" action="/upload"
method="post">
```

```
File: <input type="file" name="file1" />
<input type="submit" value="upload" </form>
```

Keeping the experimental system functional is of highest priority, and there is a procedure in place to verify the suitability of an incoming file. HTML offers different encryption methods to address this issue. The incoming file name should be the same as in the Python script. To ensure this, there is a script to process the incoming file in Python to change its name as well as to check the file format. Arduino only takes '.ino' formatted files in its uploading process. Once the incoming file is verified, it should be stored in a particular folder by replacing the previously uploaded file. The piece of the code used for this purpose is provided below:

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

AVRDUDE is one of the available compiler tools that allows users to upload a file into an Arduino board. When uploading the recent 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 upload, there should be a new Makefile within the folder in which the user's file is stored. The Makefile should look like the following:

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

By sending simple commands like 'make' and 'make upload' to the AVRDUDE from Python, it starts compiling the recent program in the specified folder. The 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 back to the user. To do that we have to identify the terminal procedure lines while the program is executing and save that as a text file. This will be the easiest way of showing the error messages on the client's browser. The last line of the code below serves this purpose. It collects all of the lines from the terminal window into an outputfile.txt and saves it in a specific memory path on the server. The outputfile.txt contains all the error messages as well as the successful messages. The Python Tornado server can directly read the lines from the file and writes the data on the web page.[7]

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

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 message including the error category and line number of the code (Figure 11).

(a) Failure message. (b) Succcess message.

Figure 11: Success and failure message.

The GUI was designed using HTML and CSS; an image of the GUI is provided in Figure 12. On the left hand side, there are buttons for various activities. The first four buttons from the top initiate the running of four fixed programs that are already saved on the embedded processor board. The next button triggers a temperature measurement of the room and the temperature will display at the bottom of the page. The next button initiates a remote upload process that has been described earlier. With this provision the client can upload the program for the Arduino board and test and validate its performance on real hardware from a remote location. An IP camera is used to provide a real time video of the experimental setup for the clients. The last button on the menu is for resetting the Arduino board.

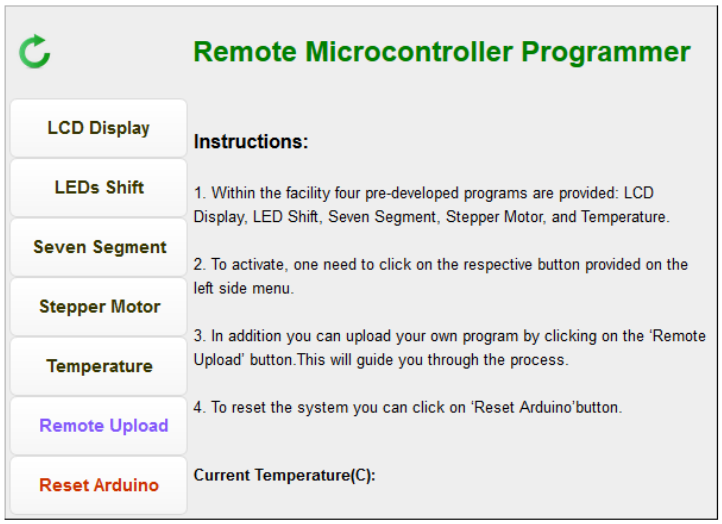


Figure 12: An image of the developed GUI.

Figure 13 and Figure 14 show images of the GUI after executing the LCD display and the seven segment display, respectively.

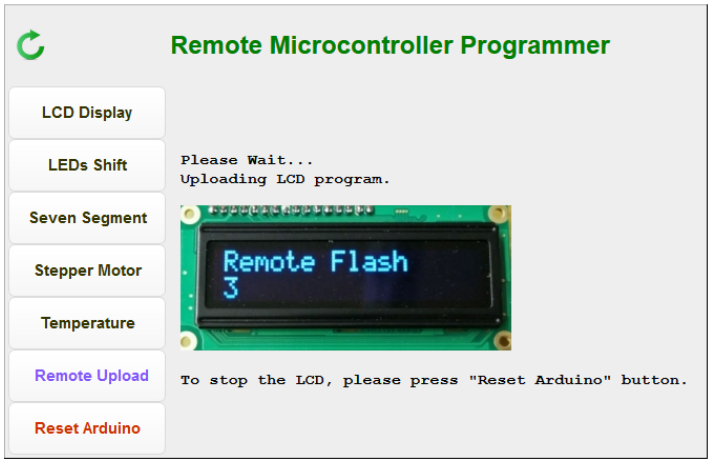


Figure 13: GUI with the LCD programming action.

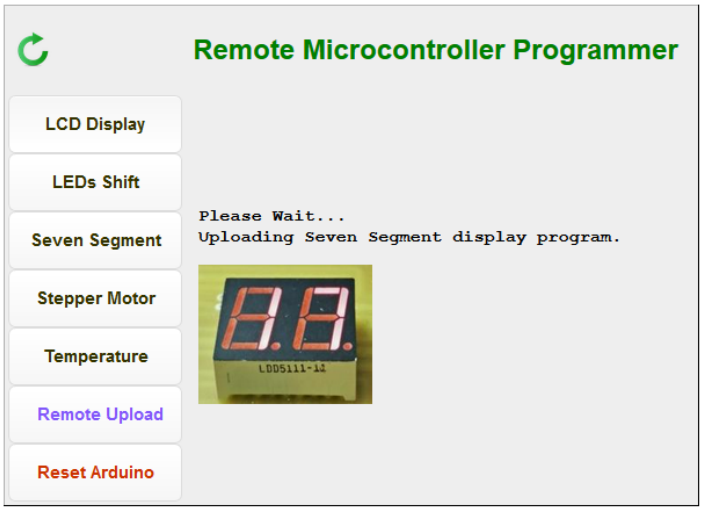


Figure 14: GUI with the seven segment display.

5. Remote Access and Pedagogy

The developed facility will be linked with a secure web portal. The web portal part of this facility involves a number of issues: system access levels, user profile and password control, documentation, experiments, weekly surveys, and administrative activities. The web portal part is independent of the experiments and can accept any form of experiments without much change. The only thing that has to change is the experiment related documentation.

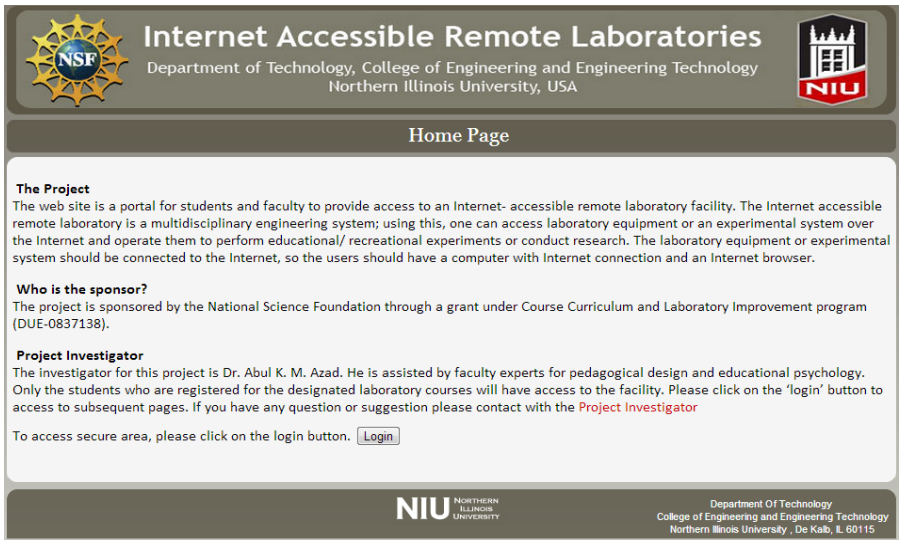


Figure 15: Homepage with client login access.

The system access level controls the level of access by a facility user. There will be two levels of access to the system. One will be as a client and the other as an administrator. Students will be allowed with client level of access. With this status, they can perform or view an experiment, change password and demographic details, respond to experience sampling method (ESM) questions, and complete the weekly survey questionnaire. An administrator level of access will allow management of experiments and monitor and gather access profile and survey data. An image of client login page is shown in Figure 15; while an image for the laboratory experiment page is shown in Figure 16.

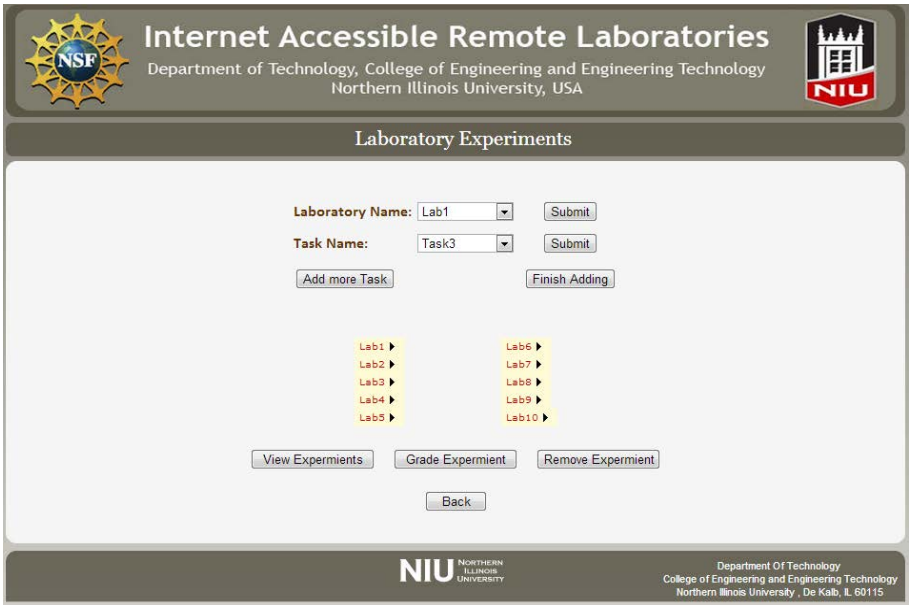


Figure 16: Web access portal for laboratory experiments.

An image of the web application page available via the Laboratory Management Server is shown in Figure 17. This application page was developed using .NET technology connected to an SQL Server. The page provides a portal to the remote laboratory facility while facilitating all of the requirements mentioned earlier.

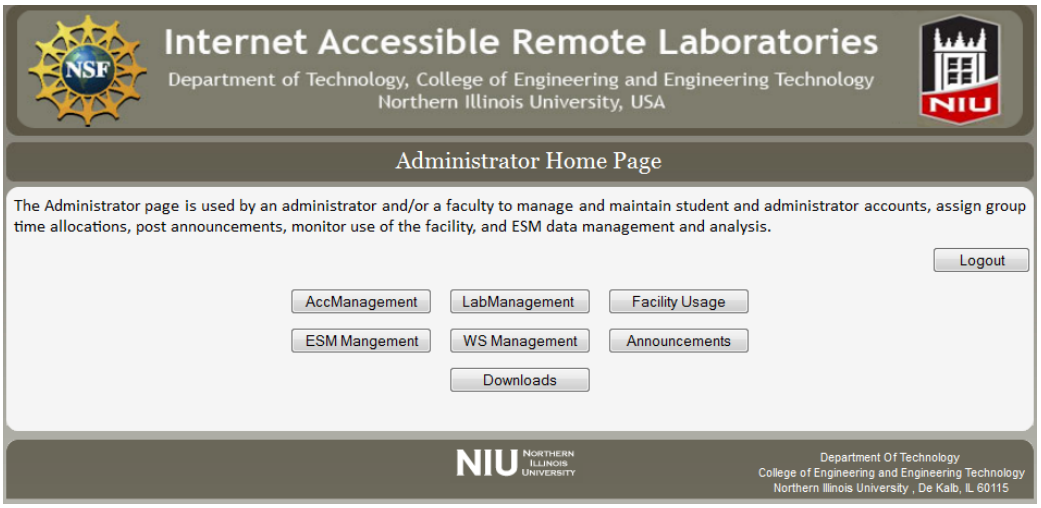


Fig. 17. An image of administrator home page.

The developed online laboratory experimental system is intended to use for a senior level undergraduate course on embedded system. The topic involves introduction of Arduino system, use of Arduino software tools, transducers and signal conditioning, input/output operations, pulse width modulation and timers, and communication protocols. All these will be implemented using the developed system that has a stepper motor, liquid crystal display, 7-segment, and light emitting diodes. On this one needs to prepare necessary experiment handout and instructions.

6. Conclusions

The first part of the paper discusses the background of a remote laboratory along with its makeup. This is followed by the development of remote testbeds using the Python as the middleware and GUI preparation software. Two case studies are provided to demonstrate the effectiveness of Python as the software for remote testbed development. The first case study is a remote vacuum cleaner, while the second is an embedded processor system with a provision for remote programming. The remote vacuum cleaner can be operated in auto and manual modes. In auto mode, the system operates autonomously in a pre-programmed route while avoiding obstacles. When in manual mode, one can direct the system in various directions using a few buttons. The embedded processor system allows the clients to run four fixed programs as well as upload their own program to run any of the output devices. With the remote uploading facility, users can operate any of the output devices to enhance their programming skills. There are two important features of this facility. One is the use of a single software package for communicating with the experiment and development of the GUI as well as data processing. The second one is the simplicity for client access to the system; a client simply needs a browser without any plugin to be installed.

Acknowledgment

The authors would like to thank the National Science Foundation (NSF) for its support for the reported work. This paper is based on an NSF TUES (Transforming Undergraduate Education in Science, Technology, Engineering, and Mathematics) project, award number DUE-1140502. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References:

[1] Gomes, L. and Bogosyan, S. (2009). Current Trends in Remote Laboratories, *IEEE Transactions on Industrial Electronics*, **56**(12), Pages 4744 – 4756, October 2009

[2] Chen, S., Huang, Y, Zhang, C. (2008), Toward a Real and Remote Wireless Sensor Network Testbed, *Third International Conference Wireless Algorithms, Systems, and Applications*, WASA 2008, Dallas, TX, USA, October 26-28.

[3] Axaopoulosa, P. J., Moutsopoulos, K. N. and Theodoridis, M. P. (2012). Engineering education using a remote laboratory through the Internet, *European Journal of Engineering Education*, **37**(1), pp. 39-48.

[4] Tripathi, P. K., Mohan, J., and Gangadharan, K. V. (2012). Design and Implementation of Web based Remote Laboratory for Engineering Education, *International Journal of Engineering and Technology*, **2**(2), pp. 270-278.

[5] Gravier, J. Fayolle, B., Bayard, M. A. and Lardon, J. (2008), State of the Art About Remote Laboratories, *International Journal of Online Engineering*, **4**(1), pp. 19-25.

[6] *Python.* (2015). Retrieved from Python Software Foundation, https://www.python.org/

[7] *Tornado.* (2015). Retrieved from Tornado Stable: http://www.tornadoweb.org/en/stable/

[8] Boldt, E. (2013). Python Web UI with Tornado. Retrieved 01 04 2015, from http://robotic-controls.com/learn/python-guis/python-web-ui-tornado

[9] Pratyaks, D. (2013). *Programming and uploading Arduino sketch without IDE.* Retrieved from linuxcircle.com http://www.linuxcircle.com/2013/05/15/programming-and-uploading-arduino-sketch-without-ide/

[10] *Arduino Mega.* (2015). Retrieved from Overview of Arduino mega: http://arduino.cc/en/Main/Arduino BoardMega

[11] The Arduino Mega 2560 is a microcontroller board. (2014). Retrieved from Atmel Corporation: http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega6401280-1281-2560-2561_datasheet.pdf