Design and Implementation of Experiments with Real-Time Shared Architecture using Different Mobile Systems

http://dx.doi.org/10.3991/ijoe.v11i2.4133

Bahaa I. Kazem[1,2], Mazin Rahman Ali[1]
[1] University of Baghdad, Iraq.
[2] University of Missouri-Columbia, USA.

***Abstract*—This work presents a new design for an internet laboratory at University of Baghdad (UoB-iLab) that can be used by students off-campus to execute real time experiments in the on-campus. The UoB-iLab architecture is designed to be used with limited internet bandwidth and concurrent access by users and administrators. The Arduino ® Microcontroller was used to establish the interface platform with various laboratory devices using C and C++ programming languages to control input/output signals for a specific lab device. The web interface for the system was developed using HTML, PHP, and AJAX along with JAVA to build graphical real-time interfaces for the experiments that can work on any operating system platform (Windows, Mac OS, Linux, Unix, BeOS) on PCs and mobile devices. Several experiments were designed to check the reliability of the suggested UoB-iLab architecture to deal with different types of input and output configurations during the real experiments session while using several types of sensors and actuators. The suggested design of UoB-iLab can be used to implement several types of real-time experiments to improve the practical skills for engineering college students at the University of Baghdad and other universities without the need for high resource investment.**

***Index Terms*—iLab, remote control, web-based laboratory.**

I. INTRODUCTION

The rapidly expanding use of information and communication technology in an educational environment and its reflection on the teaching and learning styles make it is possible for schools, universities, and other educational institutions to offer web-based online laboratories as an alternative to the traditional laboratory experience.

The educational institutions that would have struggled to or been unable to provide a sufficient traditional laboratory experience and equipment can use distance-learning technology that provides the opportunity to incorporate online laboratory experiences in their course curriculum. Consequently, more students in technical and science courses are able to participate in valuable laboratory experiences. Even universities without such concerns can benefit from online laboratories as a complement to their existing hands-on and traditional facilities [1].

Certainly, online laboratories are not a perfect replacement for traditional laboratories. The students do not get hands-on experience setting up the experiments and have limited debugging abilities. However, an online laboratory system ensure that the laboratories remain open at night to serve students in any part of the world at a minimum cost where traditional and expensive equipment is not available.

Online laboratories are flexible and can be accessed by a large number of users from anywhere at any time. The flexibility lets students conduct their experiments at a convenient time and location. It also allows institutions to keep their laboratories open for much longer, which is usually not possible with the traditional hands-on laboratory. Furthermore, universities can use online laboratories to expand their offering of available laboratory experiments with minimal additional cost so that the entire class of students can perform the same experiment on shared equipment.

A lot of research has been conducted by researchers [1]-[33]. They have used different types of microcontrollers to connect and control laboratory devices and web-based platform engines as shown in Appendix B.

The first iLab project was started at MIT in 1998 by one of the authors [2]. The initial inspiration for the first iLab came from the frustration that MIT's courses on semiconductor devices did not contain a laboratory component. Traditionally, students in these courses were exposed only to theoretical device models presented in lectures and course texts. At the same time, an Agilent 4155B semiconductor parameter analyzer (shown in Fig. 1) was available with spare capacity in a graduate research lab. While the underutilization of the Agilent equipment seemed to provide an opportunity to have this tool also used in education, there was no way to accommodate the students taking courses using a single piece of equipment in the crowded research lab.

Students in an upper level electrical engineering course in the fall of 1998 were the first to try this system, i.e., Microelectronics WebLab. By the following spring, the hardware and software combination had proved to be reliable to the point that an undergraduate class of nearly 100 students employed the online lab for an assignment.

According to the available research, there are several limitations with using the available iLab systems, especially with low infrastructure locations. They include:

The student is required to have LabView Runtime installed. While this can be a perfectly affordable requirement on a personal computer, it can become a problem if the student tries to use the system on a computer where he does not have administration permissions, such as a computer room at the university or a Cyber Cafe.

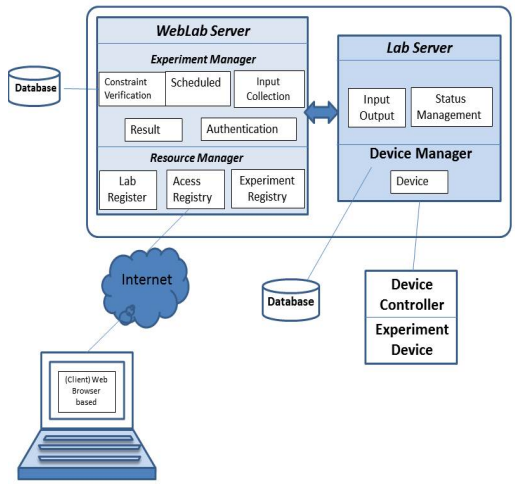

Figure 1. iLab Architecture [7].

LabView Runtime is not available for all platforms. While covering Microsoft Windows could be enough in most cases, it fails to achieve the ongoing trends of using mobile devices (including tablets such as iPads or Samsung Galaxy Tabs) more often than the computer. Implementing m-learning with a remote laboratory based on LabView Runtime is plainly impossible.

The communication is not encrypted and hard to encrypt. The fact that it does not use HTTPS makes it difficult to create a tunnel that the web browser can understand and to make it possible to send secure messages to the server. Depending on how affordable the risk of malicious users breaking the remote hardware is, it might become a problem.

The communication is based on a TCP socket. This makes the deployment difficult when trying to support other web platforms (such as Apache or IIS). If it is required to have two servers in the same machine, a non-standard port must be opened, both by the IT services of the host university and by the client system (which might not be even possible if the user is at a foreign university).

In this work the required web-based interface platform using basic programing languages (JAVA, HTML, AJAX, and Processing Language – PHP) was developed. Arduino® Microcontroller will be used for interfacing the laboratory devices since it is inexpensive and has several good features when compared with commercially available microcontrollers.

The suggested UoB-iLab system in this work needs to have following features:

- Open-source development model that gives users the ability to improve the product.
- Compatibility with major PC operating systems like Linux, Mac, or Windows.
- Support of running real-time applications. This special feature is possible due to use of a separate AVR microcontroller programmed in assembly language.
- Can be extended to a wide range of multidisciplinary applications where innovators can use it as a base to develop advanced projects, as it can deal with different types of input/output protocols.

The developed UoB-iLab was used to design several basic experiments to be used by Mechatronics engineering students at the Al–Khwarizmi Engineering College at the University of Baghdad. The students can access and run real-time experiments from any location at any time without any special application or software to be installed on their PCs. Also, UoB-iLab can be integrated with other international resources of iLab around the world.

II. iLAB ARCHITECTURE OVERVIEW

From the perspective of the internet shared architecture, online experiments fall into two broad categories [7].

- A batch experiment in which the entire course of the experiment can be specified before the experiment gets running. Experiment execution takes place in machine time.
- An interactive experiment in which the user monitors and controls one or more aspects of the experiment during execution. Experiment execution takes place in real time.

iLab architecture consists of three layers: Client, WebLab server, and experiment device. The WebLab server is responsible for managing lab server experiments for the students who will perform experiments, as shown in Fig. 1.

The WebLab server plays an important role in the management of the relationship between the lab server at laboratory side and the client (user) at the other side.

The roles of the WebLab server include management of authorization, authentication, and registration, scheduling, and periodization of users.

The lab server's role is to control the laboratory devices to implement a specific set of experiments according to the client's (user's) needs.

III. UOB-ILAB ARCHITECTURE

The suggested design for UoB-iLab must be reliable and easily implemented at the University of Baghdad with the following design features:

The system can be used by mechatronics engineering students at the Mechatronics Engineering Department at the University of Baghdad

The design of the UoB-iLab architecture can be used to support secured authentication for clients and administered with low cost and limited internet bandwidth.

The suggested design for (UoB-iLab) can be implemented using an Arduino® Microcontroller platform with a web interface.

The design and implementation of the required web interface can be used to control and interact with electronic and mechanical devices.

Figure 2 shows the basic design hardware and software components for UoB-iLab. The required programing and interface languages to complete data transformation and acquisition between the user interface and laboratory device are also listed in Fig. 2.

The major laboratory activities that need to be considered to complete an experiment in an integrated manner are as follows:

- Switching the process of equipment and devices.
- Getting the result values from the sensors.
- Modifying the value of the process variables.
- Achieving the concept of the feedback control system.

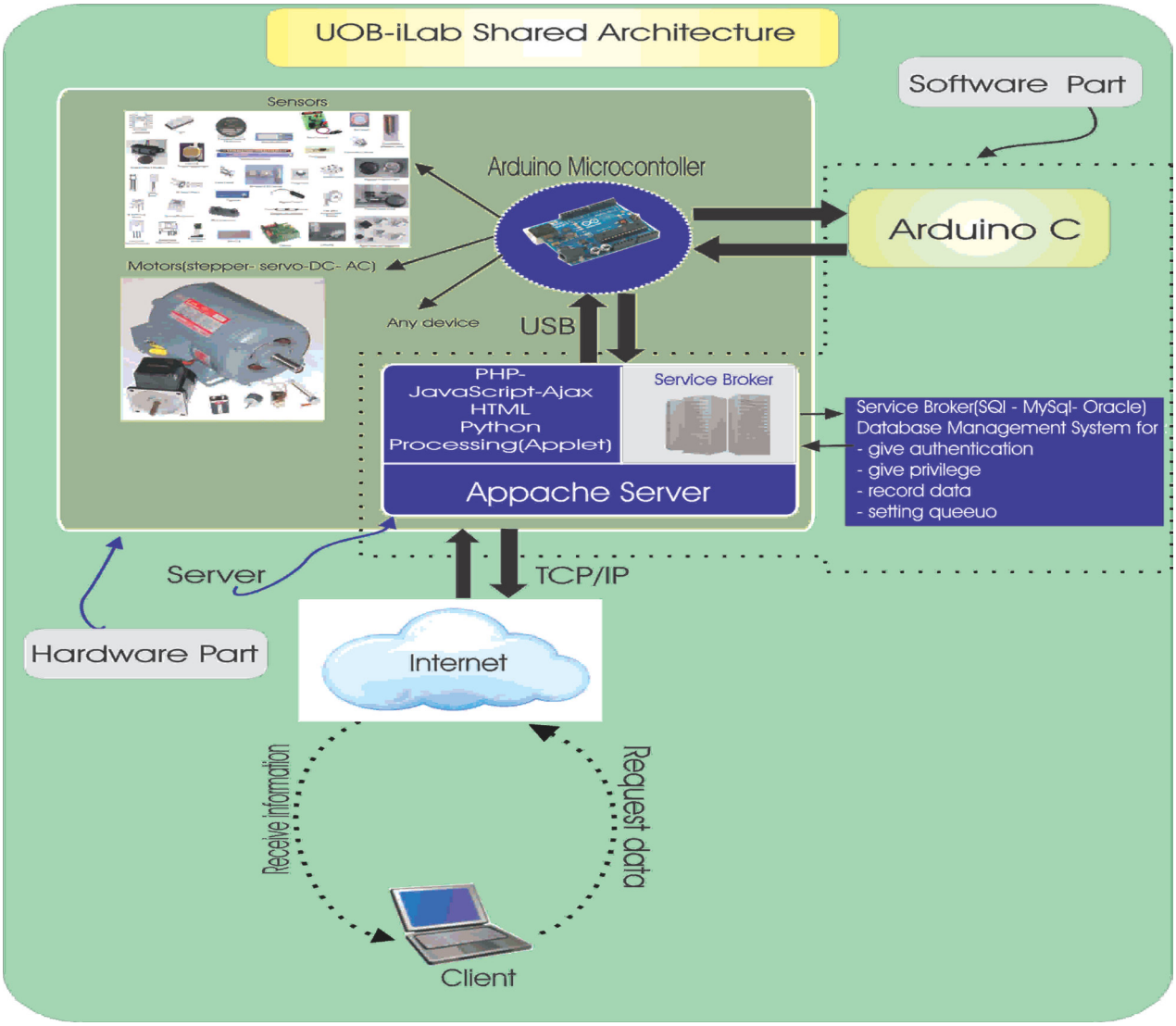

Figure 2. UoB-iLab Architecture.

IV. MICROCONTROLLER DESIGN AND PROGRAMMING

The Arduino® Microcontroller board is provided with a USB plug to connect to a computer and a number of

connection sockets that can be wired to external electronics such as motors, relays, light sensors, laser diodes, loudspeakers, microphones. They can be powered through the USB connection from the computer or by using a 9V battery. They can be controlled from the computer or programmed by the computer and then disconnected and allowed to work independently [34].

Arduino is an open source code environment that makes it easy to write codes and upload them on the input/output (I/O) board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on processing, avr-gcc, and other open source software. The physical computing platform is based on a simple I/O board and a development environment that implements processing. Arduino is composed of two major parts [35]:

Part 1: the Arduino board, which is the piece of hardware used to build objects.

Part 2: the Arduino IDE, the piece of software to be run on computer.

The Integrated Development Environment (IDE) is a cross-platform application written in Java and is derived from the IDE for the processing programming language and wiring projects. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and it is also capable of compiling and uploading programs to the board with a single click. Experiments are monitored in real time.

There is typically no need to edit files or run programs on a command line interface. A program or code written for Arduino is called a “sketch” [34].

Arduino programs are written using C or C++. The Arduino IDE comes with a software library called "Wiring” from the original wiring project, which makes many common input/output operations much easier. Users only need to define two functions to make a runnable cyclic executive program:

setup(): a function that runs once at the start of a program and can initialize settings

loop(): a function repeatedly called until the board powers off.

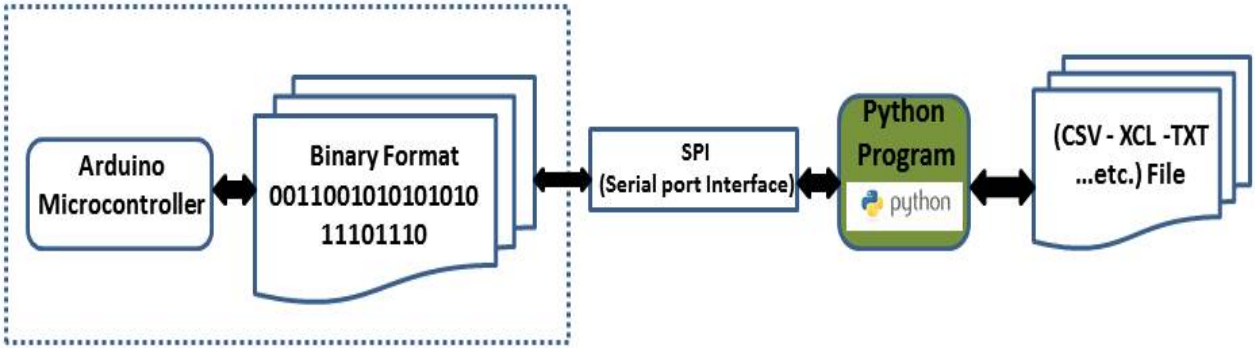

Figure 3. Steps of codes witting for an Experiment.

V. INTEGRATED RELATION BETWEEN ARDUINO MICROCONTROLLER AND PYTHON IN UOB-ILAB

Python programs are indeed often deployed in the context of larger applications. For instance, to test hardware devices, Python programs may call out to components that give low-level access to a device. Similarly, programs may run bits of the Python code at strategic points to support end-user product customization without the need to ship and recompile the entire system's source code. Python's simplicity makes it a naturally flexible control tool. Technically, although, this is also just a common Python role, many (perhaps most) Python programmers code standalone scripts without ever using or knowing about any integrated components. It is not just a control language [36].

To support communication with the Arduino board, a method of interfacing is required. Standard Arduino boards provide RS232-style serial connection to older boards over a physical RS232 connection. But with such a port that is no longer common to modern computers, recent boards include a USB for serial functionality. With Python, it is trivial to read and write to the serial devices as if it were a file on a Windows system, but there is also a wrapper library called pySerial that works well across all operating systems [35]. After installing pySerial, reading data from Arduino is straightforward:

```
>>> import serial
>>> ser = serial.Serial('COM11', 9600)
>>> while True:
…       print ser.readline()
'1 Hello world!\r\n'
'2 Hello world!\r\n'
'3 Hello world!\r\n'
```

Writing data to Arduino is easy too:

```
>>> import serial #
>>> ser = serial.Serial('COM11', 9600)
>>> ser.write('5')
```

It is necessary to connect to the same device that is connected to from within the Arduino development environment. A symlink between the longer-winded device name and COM11 to cut down on keystrokes has been created. Note that the example above will not work on a Windows machine, as the Arduino serial device takes some time to load. When a serial connection is established, it resets the Arduino.

Any write() commands issued before the device initialization will be lost. A robust server-side script will read from the serial port until the Arduino declares itself ready, and it will then issue write() commands. Alternatively, it is possible to work around this issue by simply placing a "'time.sleep(2)" call between the serial connection and the write call. The major reason for using the python program in UoB-iLab is to convert the result data from Arduino microcontroller from binary form to another form such as CSV–TXT- XCL…, which enables the other program to analyze it easily, as shown in Fig. 3.

VI. DESIGN AND IMPLEMENTATION OF EXPERIMENTS USING UOB-ILAB

Several basic experiments were designed and implemented at UoB-iLab using the same procedure given in Section IV to test system reliability and ability to capture input/output relation in real time. Figure. 4 shows steps to build any new experiments using an Arduino Microcontroller at UoB-iLab.

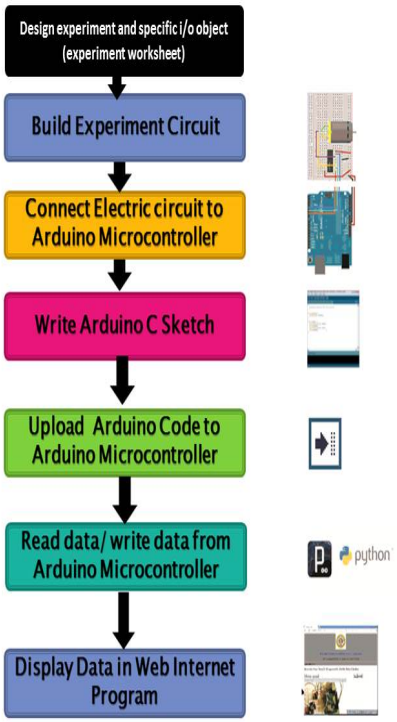

Figure 4. Steps to build any new experiments using the Arduino microcontroller at UoB-iLab.

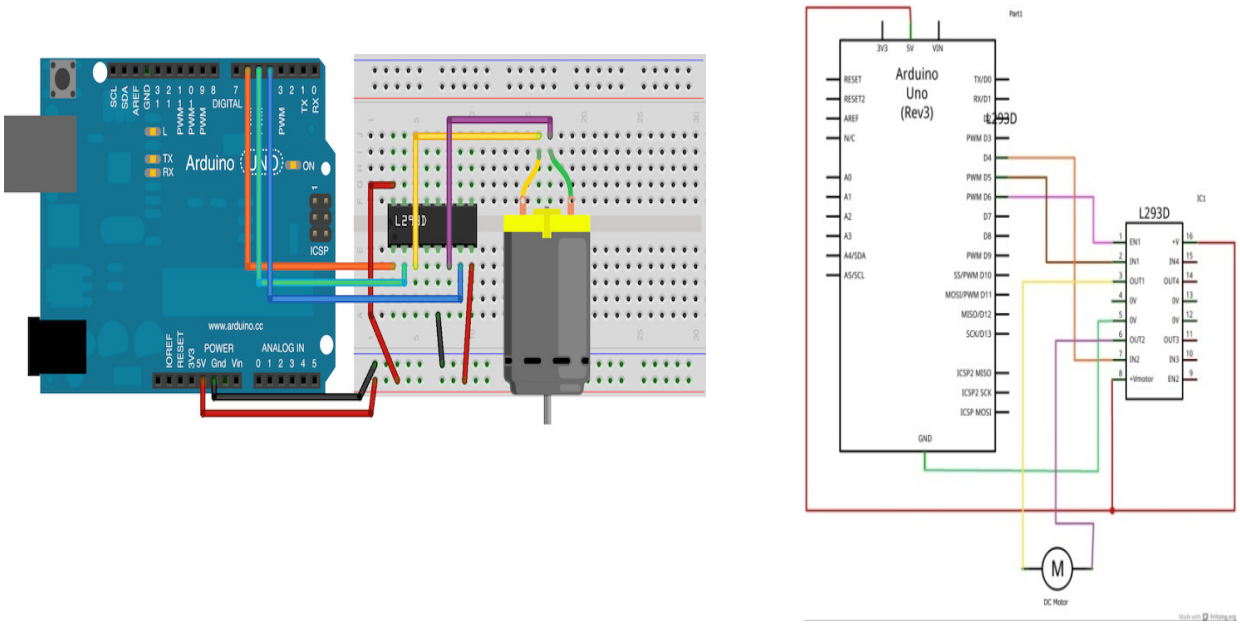

Figure 5. Schematic diagram for experiment(Controlling a DC motor).

A. *Remote control of DC motor*

Objectives of the Experiment:

The aim of this experiment is to show the capability of Arduino in controlling the speed of a DC motor via internet in UoB-iLab project and the method of using the Python program to read and write data to the Arduino platform using a USB port by converting the results from binary to text file.

The hardware requirements needed are as follows:

- Arduino Uno board
- Breadboard
- DC motor (5V)
- L293D motor driver
- Wires

The hardware configuration needed is as follows:

The motor and motor drivers must be correctly wired to the IC. First, it is necessary to plug the L293d motor driver into the middle of the breadboard. It can be started by connecting the power for this integrated circuit. After this, pins 8 and 9 must be connected to the 5V pin of the Arduino board and pin 5 must be connected to the ground pin of the Arduino board.

There are still three input pins and two pins to be connected. The output pins can be connected to the terminals of the DC motor. The output pins used for this are pins 3 and 6.

The first input pin to connect is pin 1, which is called the Enable pin. This is the pin used to turn the motor on and off and to change the speed of the motor. This pin is connected to pin 6 of the Arduino board. The schematic diagram that presents the design procedure for the new experiment is shown in Figure 5.

Finally, pins 2 and 7 of the L293D need to be connected to pins 4 and 5 of the Arduino board, respectively. These pins will be used to change the direction of the motor.

The software requirements needed are as follows:

Arduino C IDE

- Python (with pySerial library)
- PHP
- HTML
- JavaScript

The electrical circuit needed is as follows:

The complete schematic of the diagram for the DC motor control system is shown in Fig. 5.

The L293D circuit needs to work properly before performing any remote command operation. To check the proper function of the remote command operation, a simple command such as making the motor accelerate and decelerate in both directions can be performed. The script code is given in Appendix A. The setMotor function has two inputs: direction and the speed of the motor. The first step is creating two digitalWrite() operations to set the direction of the motor: one pin of the L293D circuit will receive 5V, and the other one will be set at 0V. Finally, the analogWrite() command on the enable pin is used to change the motor speed using PWM. With this function, it is easy to command the motor to move in both directions. For example, to accelerate in reverse, the following script can be used:

```
1    // Accelerate reverse
2    for (int i = 100; i < 250; i++){
3      setMotor(false, i);
4      delay(10);}
```

The motor speed for accelerating, decelerating, and reversing the rotation direction can be controlled by uploading the above script to the Arduino board.

At this point, the DC motor speed control using the Arduino® Microcontroller has been implemented. Three different software parts need to be developed to integrate it with the web interface system. First, it is necessary to slightly modify the Arduino sketch so that it can receive data from the host computer. The second part will be written in Python, which will make the interface between the computer and the Arduino board, by reading and writing data using text files. Finally, HTML, PHP, and JavaScript

together can be used to create a nice web interface for the project.

Figures 6 and 7 show the architecture of the experiment, and the role of the Python program in converting data from Hex to txt type using the USB Port and enabling the PHP program to perfectly read it.

From this experiment, it can be concluded that the Arduino® Microcontroller can be integrated with high-level data structured languages like Python and the standard scripts for web development to complete low-cost, reliable, real-time web- based laboratory system.

B. Proportional-Integral-Differential (PID) Photoresistor Controller Experiment

Objectives of the Experiment:

A photoresistor is a resistor whose resistance decreases with increasing incident light intensity.

A PID is a controller widely used in industrial control systems. A PID controller calculates an error value as the difference between measured process variables and a design setpoint. The controller attempts to minimize the error by adjusting the process through the use of a manipulated variable as shown in Fig. 8 There are three primary components to think about in a PID control loop. Each component is prefixed with a gain constant and, when added together, gives the instantaneous control value that can be used to drive the system.

Typically, a voltage is generated to control the system, so that each component can be thought of as contributing a particular voltage to the system's final output.

The voltage corresponding to the current state of the system (position, temperature, etc.) is called "Process Variable" or PV. The PV is the value that is passed to the PID control loop as feedback about the state of the system. Also, a set point voltage (SP) that corresponds to the state to be reached by the PV will be determined. Basically, the PID loop will drive the system to be a state in which the SP and PV are equal. A control voltage (u), which corresponds to the instantaneous voltage value, will be used to

Figure 6. A mock-up of the user interface for remote control of the DC motor.

drive the system toward its SP voltage. The control voltage u can be thought of as what is actually sent to the system to steer it to the desired point. Defining u (t) as the controller output, the final form of the PID algorithm as follows [37]:

$$u(t) = MV(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt}e(t)$$

The constants Kp, Ki, and Kd are used to set the sign and contribution gain of each part of this equation. e (t) is the "error" corresponding to SP and PV.

The variable t corresponds to the current time in the system under control and is simply a variable of integration. The proportional portion of the equation takes into account how far away PV is from SP. The differential part takes into account how fast the system is moving (if it is moves too fast near our SP, it will overshoot) and can be used to reduce the proportional portion if it is moving too fast or to speed up if it is experiencing resistance despite its proportional contribution.

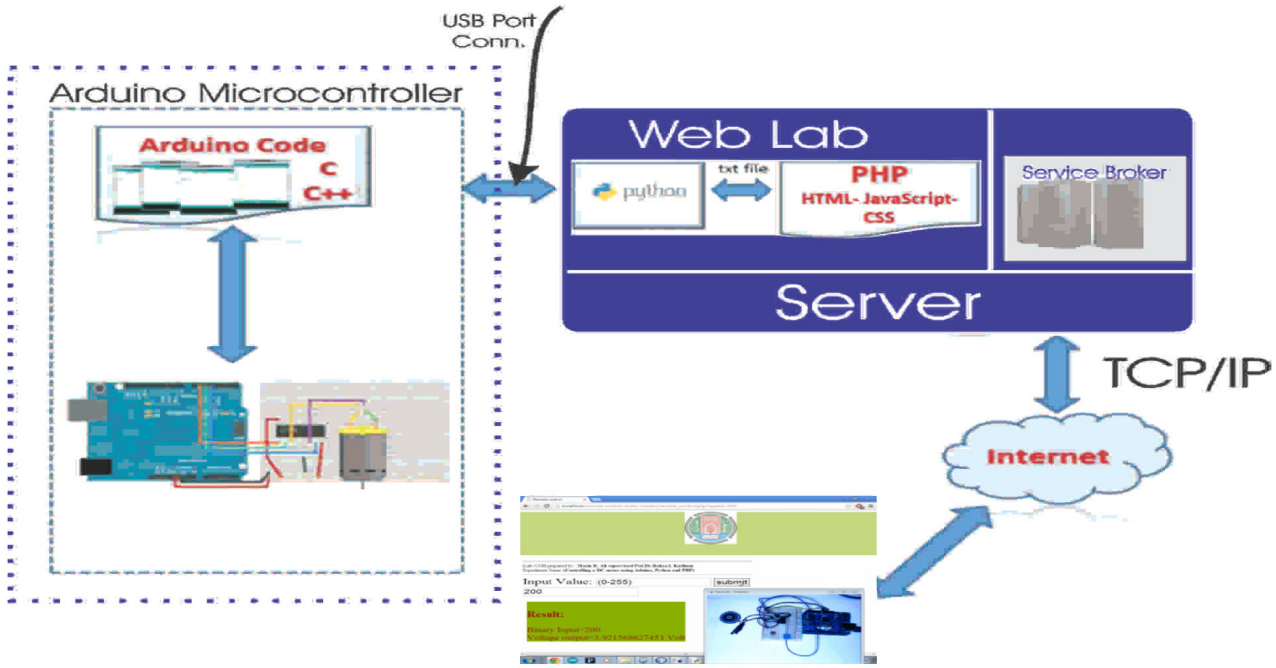

Figure 7. Architecture of experiment (Remote control of DC motor) with UoB-iLab.

The integral part of the equation takes into account how long the system has been off the set point, contributing more to system output the longer it is missing the SP. This is important because P and D contributions will typically lead the PV value to sag slightly above or below the SP value.

The hardware equipment needed is as follows:

- Arduino UNO R3.
- Breadboard
- Resistance 1 K
- Resistance 220 ohm.
- Photo resistance(10 K)
- Potentiometer (10 K)
- Light Diode (Led)

The software requirements are as follows:

- Arduino C IDE.
- Processing (Java) IDE.

Electrical circuit:

Figure 8 shows the schematic diagram for the PID controller for the brightness control system.

Writing a new PID control loop is not difficult, but there are many details to take into account. Beauregard [38] is good place to start with an existing library, a set of ready tools that can be easily used. First it is necessary to install the library.

First, download the library from following link: https://github.com/br3ttb/Arduino-PID-Library/archive/master.zip

The system may suffer from external actions called disturbances, such as shading or excessive light in the sensor or in the environment or even blocking the sensor.

The disturbances are automatically controlled by the Arduino_PID, i.e., when the incident light on the LDR (simulated by a shadow on the LDR) is decreased, the system will automatically increase the brightness of the LED, trying to maintain a constant ambient lighting.

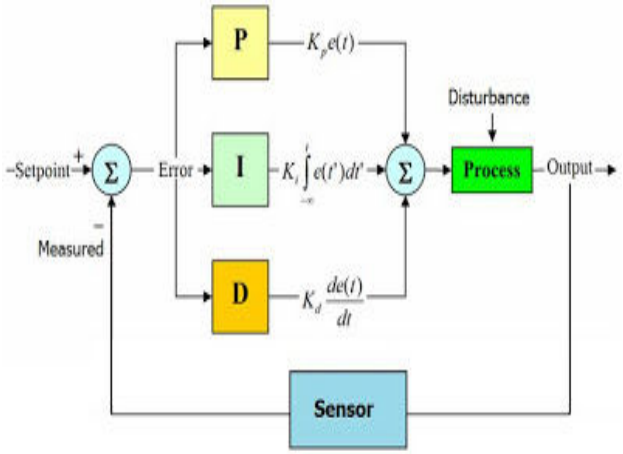

Figure 8. Block diagram of PID Controller in a feedback loop.

When the incident light on the LDR (a flashlight may increase the brightness for simulation purposes) is increased, the system will reduce the brightness of the LED, trying to maintaining a constant ambient lighting.

Another detail of this particular test is that the set point value is controlled by a potentiometer, and using the set point configuration via PID_FrontEnd is no longer needed.

The code can be downloaded from the following link: https://github.com/bcomnes/315-lab-microcontroller/blob/master/code/pid_led_set_serial/pid_led_set_serial.ino

To start with, it is necessary to use the following:

PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);

This line sets up the PID library to use a PID process called myPID. At any time, the program call

myPID.Compute();

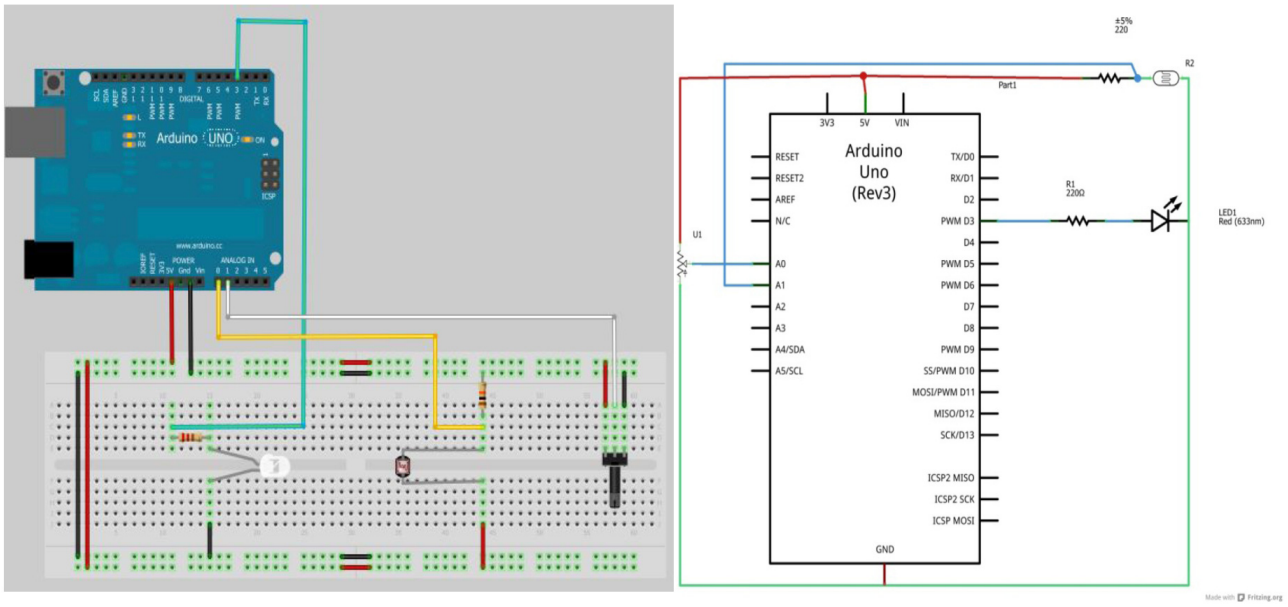

Figure 9. Schematic diagram for (PID) Controller Experiment)

Subsequently, myPID will take the variables Input, Setpoint, Kp, Ki, and Kd and use them in its calculation. It will then write whatever it determines to Output. DIRECT simply tells the PID the direction the system is working in. For example, if the PID loop tries to make the LED dimmer when it should actually get brighter, then it is necessary to change DIRECT to REVERSE.

Time stamps have been used to keep track of when some serial line feedback has been provided and to decide whether to wait or to write what the system PID loop is doing.

It is necessary to check if any commands have been sent to the Arduino every time the serial line has been updated. A very simple command parser has been created that will take 3 float variables separated by commas and set the value of Kp, Ki, Kd, respectively, with the new values. This will help in gain tuning without having to reflash the Arduino card every time it is necessary to change the tuning value.

To complete the experiment setup, the following schematic is needed:

The photo resistor needs to be placed so that it aims into the LED output or to attach the LED to a wire to vary how close it is to the photo resistor.

The disturbances are automatically controlled by the Arduino_PID, i.e., when we have a decrease in the incident light on the LDR (simulated by a shadow on LDR), the system will automatically increase the brightness of the LED, trying to maintain a constant ambient lighting.

When there is an increase in the incident light on the LDR (a flashlight may increase the brightness for simulation purposes), the system will reduce the brightness of the LED, trying to maintaining a constant ambient lighting.

Another detail of this test is that the set point value that is controlled by a potentiometer is no longer needed for using the set point configuration via PID_FrontEnd.

The following functions were used to control the experiment session:

TOGGLE_AM	changes the PID mode to automatic or manual
SETPOINT	desired amount of light in the environment(from 0 to 1024)
INPUT	the actual value of the ambient brightness (measured by LDR and returned as feedback)
OUTPUT	the control value returned by the Arduino as a PID controller (brightness control of the LED attached on pin D3)
Kp	Proportional control's constant
Ki	Integral control's constant
Kd	Derivative control's constant
TOOGLE_DR	changes the PID's direction (if output grows as grows the entry or the reverse of it)
SEND_TO_ARDUINO	send data to the Arduino
PID Input / Setpoint	Graphical Window for the setpoint (desired value for brightness) and input (measure of brightness LDR) curves.
PID Output	Graphical window for the control output of the LED. In the pictures, it can be notice the setpoint settled to 950 (green line), the input variable (red line) and the PID control output variable (blue line) to control brightness.

As shown in Figs. 10 and 11.

Where the error = SP−PV, and the error is applied to the PID controller that generates the MV for process control.

PV = Process Variable;
SP = Set Point;
MV = Manipulated Variable.

Each controller type (i.e., P, I, and D) can be used independently or jointly as follows:

- P controller (often used)
- PI controller (often used)
- PD controller (rarely used)
- PID controller (most used)

Figure 13 shows the effect of changing the set point value (from 950 to 700 and then to 950 again), where the PID controller tracks the changes in the input variable.

It can be observed that this system has some oscillation in the control (characteristic of each controlled system, whether it is first order, second order, etc.).

Figure 13 shows the effect of the disturbance input on the system response, which is very helpful information for the students when they try to understand the properties of any control system.

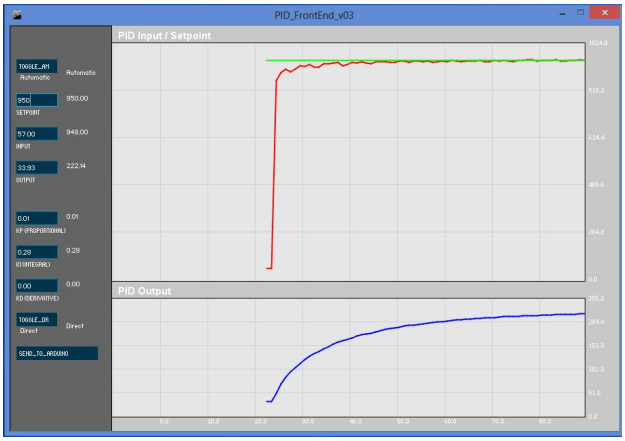

Figure 10. PID Brightness Controller.

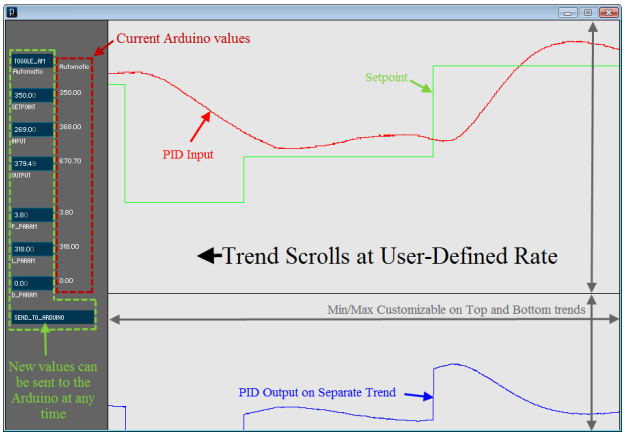

Figure 11. system responses for PID Brightness controller.

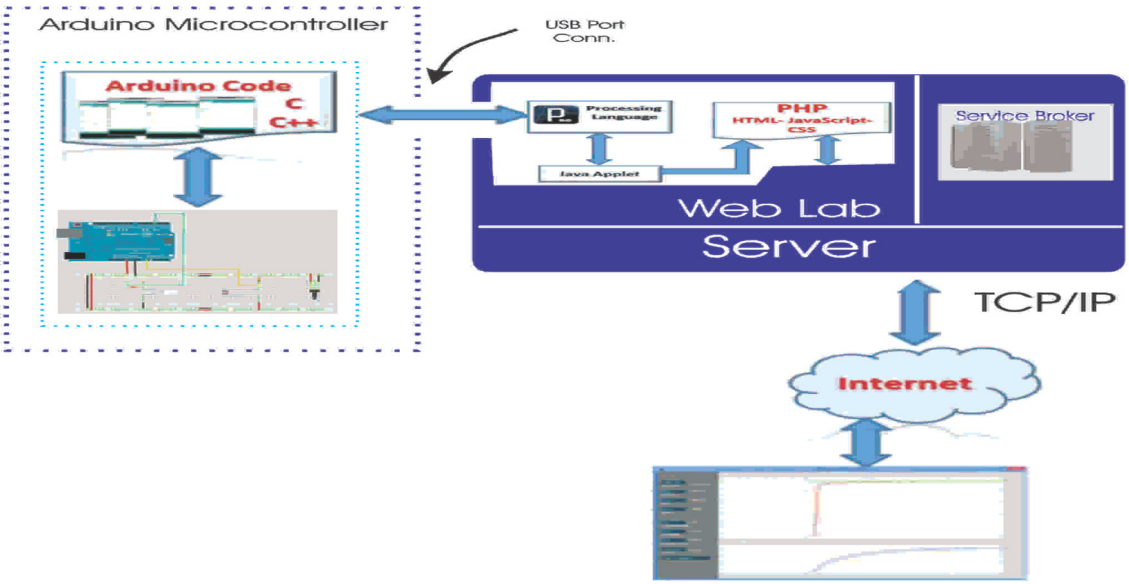

Figure 12. Architecture of PID Brightness Controller at UoB-iLab.

VI. CONCLUSIONS

The Arduino® Microcontroller was used as a reliable controlling platform for UoB-iLab, as it is compatible with different types of sensors and actuators. The designed web interface using HTML-JAVA- PHP-JavaScript code can be successfully used with an Arduino® Microcontroller to develop several remote experiments to achieve different learning objectives for the engineering college.

The developed application for UoB-iLab has several advantages:

- Can work on any web interface and does not need any runtime code.
- Can work on any operating system platform (Windows, MacOs, Linux, Unix, BeOS) on PCs and mobile devices.
- Can be easily encrypted to protect the information from spies and hackers.
- Can develop a new design of UoB-iLab using an available server like (Apache or IIS).

Python programming language has been successfully used to efficiently read data from a USB interface with a microcontroller and has the ability to convert data to different forms (txt, csv, xcl. etc.) to make it ready for use with web development programs and compatible with major operating systems (Windows, MacOS, Linux, Unix, BeOS). By using a relay Arduino® Microcontroller and its family can deal with a device that operates using more than 5V.

REFERENCES

[1] V. J. Harward, J. A. del Alamo, S. R. Lerman, P. H. Bailey, J. Carpenter, K. DeLong, C. Felknor, J. Hardison, B. Harrison, I. Jabbour, P. D. Long, L. Naamani, J. Northridge, M. Schulz, D. Talavera, C. Varadharajan, K. Yehia, R. Zbib, and D. Zych, "The iLab Shared Architecture: A Web Services Infrastructure to Build Communities of Internet Accessible Laboratories," *Proc. IEEE*, vol. 96, no. 6, pp. 931–950, Jun. 2008. http://dx.doi.org/10.1109/JPROC.2008.921607

[2] J. A. Del Alamo, "about iLabs - iLabs Dev - MIT Wiki Service," 1998. [Online]. Available: https://wikis.mit.edu/confluence/display/ILAB2/about+iLabs. [Accessed: 06-Aug-2014].

[3] I. Ahmed, H. W. H. Wong, and V. Kapila, "Internet-based remote control using a microcontroller and an embedded Ethernet," *Proc. 2004 Am. Control Conf.*, vol. 2, 2004.

[4] H. W. and V. Kapila, "Internet-Based Remote Control of a DC Motor using an Embedded Ethernet Microcontroller," in *Proceedings of the 2004 American Society for Engineering Education Annual Conference & Expositi*, 2004.

[5] G. Viedma, I. J. Dancy, and K. H. Lundberg, "A web-based linear-systems iLab," in *Proceedings of the 2005, American Control Conference, 2005.*, 2005, pp. 5139–5144. http://dx.doi.org/10.1109/ACC.2005.1470837

[6] J. Garcia-Zubia, D. L. deIpina, and P. Orduna, "Accessing WebLabs from cellular phones," in *IECON 2006 - 32nd Annual Conference on IEEE Industrial Electronics*, 2006, pp. 3779–3781.

[7] A. Agrawal and S. Srivastava, "WebLab: A Generic Architecture for Remote Laboratories," in *15th International Conference on Advanced Computing and Communications (ADCOM 2007)*, 2007, pp. 301–306. http://dx.doi.org/10.1109/ADCOM.2007.71

[8] M. Niederstaetter, T. Klinger, and D. G. Zutin, "An Image Processing Online Laboratory within the iLab Shared Architecture," *International Journal of Online Engineering (iJOE)*, vol. 6, no. 2. pp. pp. 37–40, 29-Apr-2010.

[9] D. Ursutiu, D. T. Cotfas, M. Ghercioiu, C. Samoila, P. A. Cotfas, and M. Auer, "WEB Instruments," in *IEEE EDUCON 2010 Conference*, 2010, pp. 585–590.

[10] M. E. Auer, D. G. Zutin, and C. Rajyaguru, "A LabVIEW toolkit for the development of iLab batched lab servers," in *2011 IEEE Global Engineering Education Conference (EDUCON)*, 2011, pp. 26–29. http://dx.doi.org/10.1109/EDUCON.2011.5773107

[11] K. DeLong, J. Harward, P. Bailey, and G. Kohse, "Online Spectrometer Experiments Using the iLab Shared Architecture," *Using remote labs in education: two little ducks in remote experimentation*. Ediciones Deusto, pp. 121–134, 2011.

[12] I. Gustavsson, "On Remote Electronics Experiments (Bookchapter by Ingvar Gustavsson) - Electronic Research Archive @ Blekinge Institute of Technology (BTH)." Blekinge Institute of Technology, pp. 157–176, 2011.

[13] S. Marchisio, S. Concari, H. Kofman, and F. Lerro, "Real Experiments by Remote Laboratories for Physics Teaching at Argentina," in *World Conference on Educational Multimedia, Hypermedia and Telecommunications*, 2011, vol. 2011, no. 1, pp. 1950–1955.

[14] A. Nafalski, J. Machotka, and Z. Nedic, "Collaborative Remote Laboratory Netlab for Experiments in Electrical Engineering," *Syst. Cybern. INFORMATICS*, vol. 6, no. 3, pp. 22–27, 2011.

[15] A. Selmer, M. Goodson, R. Watson, A. Braumann, M. Kraft, S. Baguley, M. Abbott, and C. Callegari, "An Undergraduate Weblab Using the SIMATIC PCS7 Process Control System," *Using remote labs in education: two little ducks in remote experimentation*. Ediciones Deusto, pp. 375–386, 2011.

[16] S. S. Tickodri-Togboa, C. Mwikirize, A. A. Tumusiime, and P. I. Musasizi, "iLabs: Revolutionizing Teaching and Learning at Makerere University," *Using remote labs in education: two little ducks in remote experimentation*. Ediciones Deusto, pp. 113–120, 2011.

[17] D. Ursutiu, C. Samoila, P. Cotfas, D. T. Cotfas, D. V. Pop, M. E. Auer, and D. G. Zutin, "Multifunction iLab implemented laboratory," in *2011 IEEE Global Engineering Education Conference (EDUCON)*, 2011, pp. 185–190. http://dx.doi.org/10.1109/EDUCON.2011.5773135

[18] D. G. Zutin and M. E. Auer, "Work in progress — Integrating educational online lab platforms around the iLab Shared Architecture," in *2011 Frontiers in Education Conference (FIE)*, 2011, pp. F1G–1–F1G–3.

[19] D. G. Zutin, M. E. Auer, and I. Gustavsson, "A VISIR lab server for the iLab Shared Architecture," in *2011 IEEE Global Engineering Education Conference (EDUCON)*, 2011, pp. 30–33. http://dx.doi.org/10.1109/EDUCON.2011.5773108

[20] C. Mwikirize, "A Sequential Logic iLab Utilizing NI ELVIS II+ and the Interactive iLab Architecture," *International Journal of Online Engineering (iJOE)*, vol. 8, no. 3. pp. pp. 34–40, 23-Jul-2012.

[21] K. P. A. O.B.Akinwale, "Implementing Remote Laboratories with the iLab Architecture: Three Case Studies from Obafemi Awolowo University, Nigeria," *Comput. Educ. J.*, vol. 3, no. 1, pp. 86 – 98, 2012.

[22] D. O. X. C. O. A. D. S. and D. Olowokere, "Virtual and Remote Laboratory Framework Development for Engineering Technology Education—A Case Study (ASCE)," in *Earth and Space 2012*, 2012, pp. 1211–1217.

[23] O. B. A. O.S. Oyebisi, "Development of a Remote Operational Amplifier iLab Using Android-Based Mobile Platform," *Comput. Educ. J.*, vol. Volume 4, no. 4, pp. 100 – 110, 2013.

[24] C. Mwikirize, A. T. Asiimwe, L. Musasizi, V. Namuswa, M. D. Nakasozi, C. Mugga, A. Katumba, S. S. Tickodri-Togboa, J. Butime, and P. I. Musasizi, "Development of Online Laboratories for Modulation and Combinational Logic Circuit Analysis Using NI ELVIS II™ Platform," *Inf. Technol. New Gener. (ITNG), 2010 Seventh Int. Conf.*, 2010.

[25] K. DeLong, V. J. Harward, P. Bailey, J. Hardison, G. Kohse, and Y. Ostrocsky, "Three Online Neutron Beam Experiments Based on the iLab Shared Architecture," *International Journal of Online Engineering (iJOE)*, vol. 7, no. 1. pp. pp. 4–9, 30-Jan-2011.

[26] J. García-Zubía, D. López-de-Ipiña, P. Orduña, and U. Hernández-Jayo, "Experience with WebLab-Deusto," in *IEEE International Symposium on Industrial Electronics*, 2006, vol. 4, pp. 3190–3195.

[27] X. Chen, G. Song, and Y. Zhang, "Virtual and Remote Laboratory Development : A Review," in *Earth and Space 2010: Engineering; Science; and Operations in Challenging Environments*, 2010, pp. 3843–3852.

[28] F. Lerro, S. Marchisio, S. Martini, H. Massacessi, E. Perretta, A. Gimenez, N. Aimetti, and J. I. Oshiro, "Performing Real Experiments From a Remote Learning Management System," *IEEE Rev. Iberoam. Tecnol. del Aprendiz.*, vol. 9, no. 1, pp. 23–27, Feb. 2014.

[29] J. L. Hardison, K. DeLong, V. J. Harward, J. A. del Alamo, R. Shroff, and O. Oyabode, "Enabling Remote Design and Troubleshooting Experiments Using the iLab Shared Architecture," in *Earth and Space 2010*, 2010, pp. 3721–3733.

[30] B.-A. Deaky, D. G. Zutin, and P. Bailey, "The First Android Client Application for the iLab Shared Architecture," *International Journal of Online Engineering (iJOE)*, vol. 8, no. 1. pp. pp. 4–7, 16-Feb-2012.

[31] T. L. L. Ingvar Gustavsson, Johan Zackrisson, Kristian Nilsson, Javier Garcia-Zubia, Lars Håkansson, Ingvar Claesson, "A Flexible Instructional Electronics Laboratory with Local and Remote Lab Workbenches in a Grid," in *2nd International Workshop on e-learning and Virtual and Remote Laboratories*, 2008, pp. 45 – 50.

[32] O. H. Graven and D. A. H. Samuelsen, "Using Remote Laboratories in Combination with Hands-on Laboratories in an Analogue Electronics Module," *Using remote labs in education: two little ducks in remote experimentation*. Ediciones Deusto, pp. 231–252, 2011.

[33] B. Aktan, C. A. Bohus, L. A. Crowl, and M. H. Shor, "Distance learning applied to control engineering laboratories," *IEEE Trans. Educ.*, vol. 39, no. 3, pp. 320–326, 1996. http://dx.doi.org/10.1109/13.538754

[34] [34 S. Monk, *30 Arduino Projects for the Evil Genius, Second Edition [Kindle Edition]*. McGraw-Hill/TAB Electronics; 2 edition, 2013.

[35] M. Banzi, *Getting Started with Arduino*. O'Reilly Media; 1 edition, 2009, p. 128.

[36] "Arduino Playground - Python." [Online]. Available: http://playground.arduino.cc/interfacing/python.

[37] B. Comnes and A. La Rosa, "Arduino PID Example Lab," 2013.

[38] https://github.com/br3ttb/Arduino-PID-Librar

AUTHORS

Bahaa I. Kazem is with Mechatronics Engineering Department, Al-Khwarizmi Engineering College, University of Baghdad, Iraq and currently Visiting Professor, MAE-college of Engineering, University of Missouri-Columbia, USA.

Mazin Rahman Ali is with Mechatronics Engineering Department, Al-Khwarizmi Engineering College, University of Baghdad, Iraq.

Submitted 22 August 2014. Published as resubmitted by the authors 10 March 2015.

APPENDIX A

Script adopted Arduino C for Controlling a DC motor

```
1   int motorPinPlus = 4;
2   int motorPinMinus = 5;
3   int motorPinEnable = 6;
4   // Setup
5   void setup()
6   {
7     pinMode(motorPinPlus, OUTPUT);
8     pinMode(motorPinMinus, OUTPUT);
9     pinMode(motorPinEnable, OUTPUT);
10    Serial.begin(9600);
11  }
12  // Loop
13  void loop()
14  {
15    // Accelerate forward
16    for (int i = 100; i < 250; i++)
17    {
18      setMotor(true, i);
19      delay(10);
20    }
21    // Decelerate forward for (int i = 255; i > 100; i--) {
22      setMotor(true, i);
23      delay(10);
24    }
25    // Accelerate reverse
26    for (int i = 100; i < 250; i++){
27      setMotor(false, i);
28      delay(10);
29    }
30    // Decelerate reverse
31    for (int i = 255; i > 100; i--){
32      setMotor(false, i);
33      delay(10);
34    }
35  }
36  // Function to control the dc motor
37  void setMotor(Boolean forward, int speed){
38    digitalWrite(motorPinPlus, forward);
39    digitalWrite(motorPinMinus, !forward);
40    analogWrite(motorPinEnable, speed);
41  }
```

The core of this script is the function setMotor, and it will also be used in the rest of this project:

```
// Function to control the motor
void setMotor(Boolean forward, int speed){
  digitalWrite(motorPinPlus, forward);
  digitalWrite(motorPinMinus, !forward);
  analogWrite(motorPinEnable, speed);
}
```

APPENDIX B.

Remote Laboratories architecture (1996-2014)

No.	(Reference number, Year)	Microcontroller Type	Web based Platform	Country
1	([33],1996)	Control Interface (MCI) and UNIX work-station	shell scripts on the workstation and corresponding batch files on the PC.	USA
2	([1],2008; [2], 1998)	Agilent 4155B	Java Applet	USA
3	([3],[4], 2004)	DSTINIM400	Java Applet	USA
4	([25],2008)	ELVIS /DAC	VS.NET C#	Nigeria,
5	([5]., 2005)	LabJack DAQ	Java Applet	USA
6	([7], 2006)	WebLab PLD	AJAX	Spain
7	([13],2011)	WebLab-PLD hardware	AJAX	Spain
8	([15], 2011)	PLC S7-400	PCS7	UK
9	([8],2010)	WebLab-PLD	Ajax- java	Spain
10	([16],2011; [26],2010)	ELVIS	LabView - Matlab	Uganda
11	([11],[27], 2011)	4DH1 beam port (Nuclear Reactor)	LabView	USA
12	([6],[28],2006)	XR4000(autonomous mobile Robot)	Ajax – Servlet- JSP	UK
13	([29],2010)	NI-ELIVIS	Java Applet	USA
14	([30] 2014)	DAQ-1200	Ajax- Visual Studio.net	Australia
15	([31],, 2010)	ELMO (HV-550XG)	LabView	Australia,
16	([10], 2011)	NI-ELVIS	LabView	USA
17	([9],2010; [17], 2011)	PXI-8360	LabView	USA
18	([19],2011)	Agilent USB device U2351A	LabView	Australia
19	([18],2011)	VISIR workbench	FLASH	USA
20	([33]. 2012)	ELVIS	LabView	USA
21	((12],[34],2011)	VISIR Platform workbench	LabView	Sweden
22	([14],2011)	E8408A- VXI	CAD (Matlab- multism)	Australia
23	([20],2012)	ELVIS -DAQ	Java Applet	USA
24	([22], 2012)	ELVIS-II	LabView	Uganda
25	([36],2011)	PCI-6221	LabView	Norway