Embedded Processor for Remote Laboratory Development

Abstract: This paper describes the design, development and implementation of a remote laboratory facility utilizing an embedded processor system to reduce the development cost as well as minimize the implementation time and maintenance overhead. Two experimental processes are reported within this paper. One is the remote programming of a Raspberry Pi to control and monitor a number of sensors and actuators, and the other is to control a coupled tank system to control the fluid levels in the tanks. Both experimental processes are supported by a user friendly graphical user interface (GUI) as well as real-time video feed. Students can develop their own controller designs and upload them to the remote server to monitor performance.

1. Introduction

The Internet of Things (IoT) has added a new element to the world of engineering and technology. With the advent of IoT, a large number of devices are now being connected to the web for data collection, management, and control [1, 2, 3, 4]. As a subset of IoT, remote laboratories allow to access laboratory equipment over the web to perform experiments. A traditional remote laboratory system involves a full-scale computer system along with associated interfacing and web hosting technologies, but sometimes there is significant overhead for the initial commission and subsequent maintenance of a remote laboratory system [5, 6]. To address this issue, this paper reports the design, development, and implementation of a remote laboratory facility using an embedded processor. The embedded processor performs the function of the computer and server used for traditional remote laboratories [7].

In terms of embedded processors, this project utilized a Raspberry Pi that the system controller as well as the web server. Raspberry Pi, a small-scale computing system with its own operating system, replaced a full-scale computer/server, thereby reducing the cost and complexity of the remote laboratory design [8, 9]. The developed experiments can be accessed remotely over the web using a suitable graphical user interface (GUI). The GUI is accompanied by a live video feed, using a Raspberry Pi Camera, so the user can have a real time video of the experimental facility. The system includes a provision so users can download data to a local system for offline analysis.

This paper presents the two remote laboratory experimental systems designed and developed using the described strategy. Section 2 presents the design strategies for the proposed embedded process based remote laboratory facility. Section 3 describes the development of a sensors and actuators experiment process in which students used Python to program a Raspberry Pi over the web to work with the sensors and actuators. Section 4 illustrates the development of a coupled tank system to perform process control experiments. The experiments involved the design and development of a feedback control system utilizing the pump and fluid level sensor outputs. The last section provides concluding statements.

2. Remote Laboratory with Embedded Systems

In this section we present the general layout of the remote laboratory system. Figure 1 illustrates this layout with Raspberry Pi at its core. We have found Raspberry Pi a cheap yet a very efficient solution to set up a remote laboratory system. As a single board computer, it

functions as a web server through which remote users can access the laboratory through the internet. It also functions as an embedded processor to interact with the hardware of the experiment modules residing in the laboratory.

It should be noted that though we have used Raspberry Pi in this project, any other single board computer, such as BeagleBone Black, can be used equally to develop such a system. Such computers, also referred to as microprocessor breakouts, at times may even be preferred over Raspberry Pi if higher capacity or performance is sought.

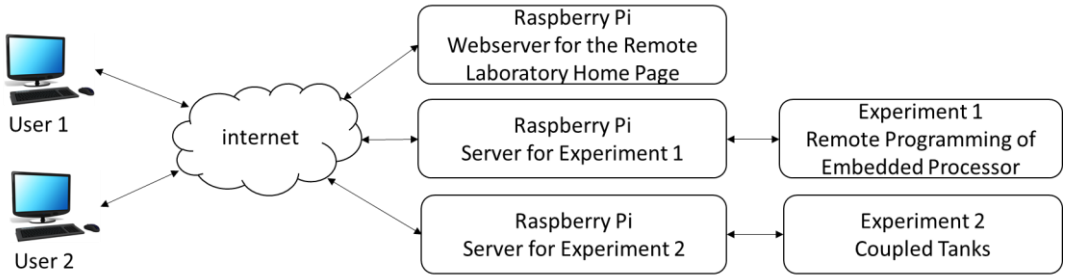

Figure 1: General layout of the remote laboratory.

Figure 1 shows two Raspberry Pis, each of which runs its own circuit and controls an experiment. A third Raspberry Pi was used in this configuration to run a webserver that posts an entry webpage to the remote laboratory. This was an initial stop point or welcome page for remote users. Then the webpage served as a gateway to direct a remote user to the webserver of the experiment he wants to access.

In this project we have used Python as the main programming language to develop the required software on Raspberry Pi due to its advantages: As one of the most widely-used high-level programming languages, a large knowledge base is available for novice programmers to start developing their programs. Additionally, Python has a simple and easy to understand syntax that emphasizes readability and thereby reduces the cost of program maintenance. As will be presented next, it is easy to develop internet applications with the aid of open source modules and packages developed for Python.

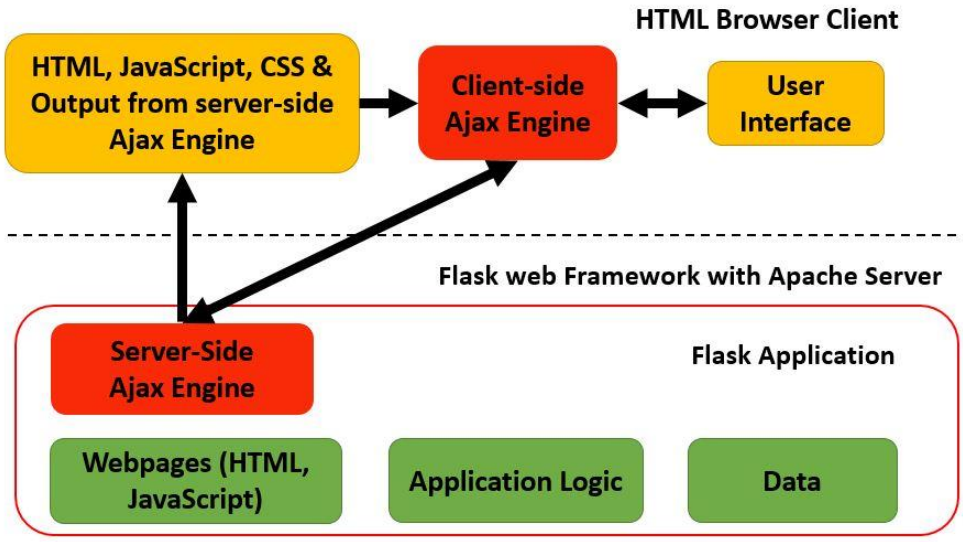

Figure 2: Software block diagram and information flow.

The operating system being used on Raspberry Pi is Raspbian, a Debian Linux based operating system customized for Raspberry Pi. Each Raspberry Pi runs its own web server, a built-in Apache webserver, that allows Raspberry Pi to render webpages for the client-side graphical user interfaces (GUIs) and facilitate remote interaction with hardware attached to the Raspberry Pi. Python is used to program the web server through Flask to facilitate this communication and execution procedure. Flask is one of the popular web frameworks, such as Django, that expedites development of web applications using Python by providing a solid core with basic services. Moreover, Flask supports databases, web forms, authentication, and other high-level tasks by the application of its extensions. Flask extensions are readily available to integrate with the core packages, and not all extensions need to be installed when developing an application. The web technologies used in this project are illustrated in Figure 2.

Bootstrap, a front-end component library, was used to develop the GUIs on the client side. Bootstrap is an open source toolkit provided by Twitter for developing with HTML, cascading style sheets (CSS), and JavaScript. When a remote client requests a specific function using a GUI on his browser, JavaScript sends the request to the server through JSON (JavaScript Object Notation) data. The request is redirected to the Flask application, which executes the Python function mapped to return the request from the given URL.

3. Experiment 1: Remote Programming of Embedded Processor

In this section we present the Remote Programming of the Embedded Processor module, one of the experiments we developed for the remote laboratory. In this module, multiple input and output devices are connected to a Raspberry Pi, which serves as an embedded processor to manipulate these devices. The goal of the experiment was to provide students with a platform to learn how to develop an embedded system using Raspberry Pi as the embedded processor and Python as the programming language.

A student can remotely access this module through the webpages broadcast from the webserver on the Raspberry Pi. The student can watch a live video stream that shows the entire module as well as observe the state of the devices attached to the module through custom displays on the GUIs of these webpages. The student can manipulate the state of these devices through the controls provided in the GUIs and immediately observe the changes made.

The Python codes that perform the manipulation of the devices are downloadable from the webpages together with relevant background information needed to program the Raspberry Pi for these manipulations. The student can download these code files and study them to learn how they work. Then the student can write his own Python program to manipulate these devices, upload his code file through a GUI on the webpage, execute his own code remotely on Raspberry Pi, and immediately observe the results. Thus, the main design aspect of this system is that it can be programmed from anywhere at any time.

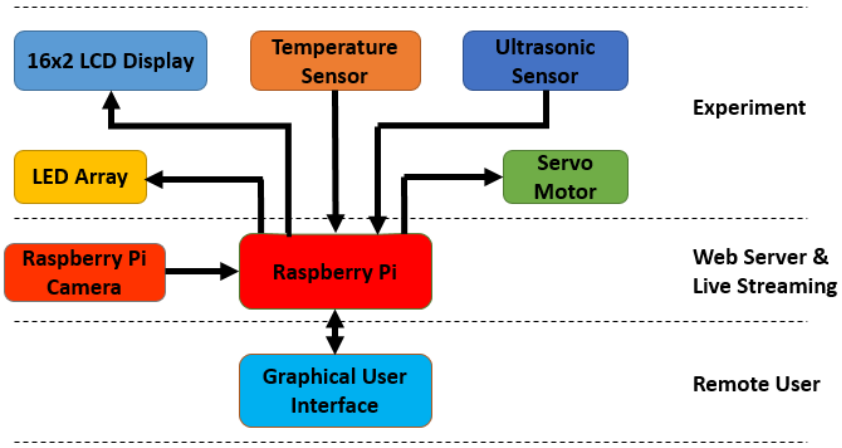

Figure 3: Block diagram of remotely programmable Raspberry Pi.

Figure 3 shows the general layout of this experiment. In addition to a live video camera attached to the Raspberry Pi, the setup includes two input devices: an ultrasonic sensor and a temperature sensor. There are three output devices: LEDs, a 16x2 LCD, and a servo motor.

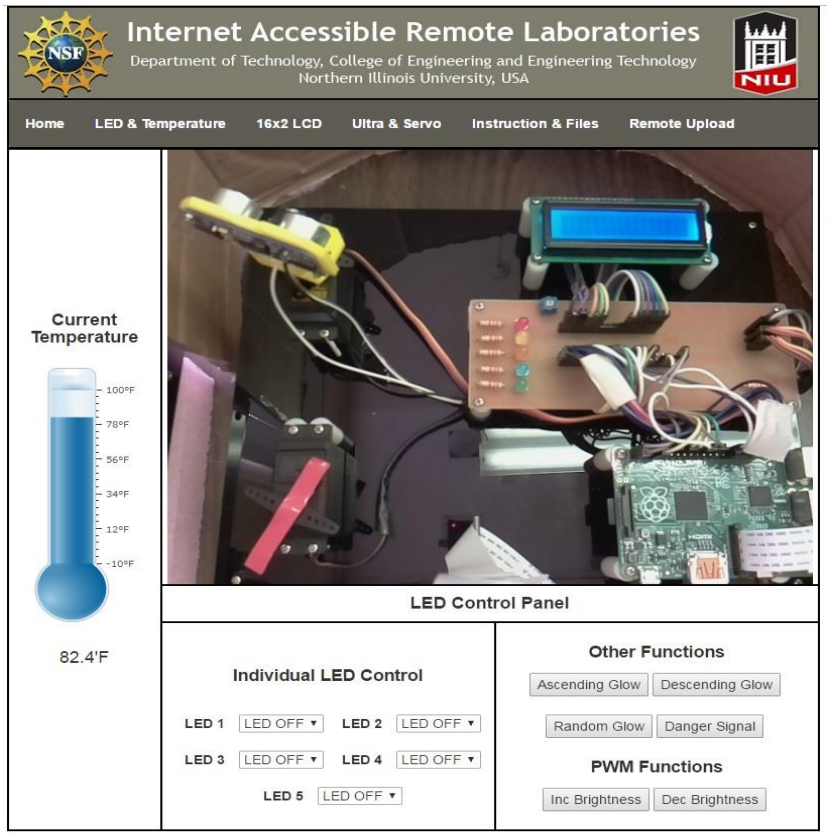

Figure 4: GUI for LEDs and temperature sensor.

Figure 4 shows an image of the webpage for LED control and temperature monitoring. The horizontal menu on the page has links to custom pages for other devices in the module. The picture on the image is the live video of the module from the camera, which shows the ultrasonic sensor at the top left, servo motor at the bottom left, LEDs at the center of the circuit board, and the LCD display above and Raspberry Pi below the circuit board. The GUI

on the left of the page shows the temperature in the laboratory room using a graphical display and in text. The GUI at the bottom shows the state of the LEDs and provides controls to manipulate their states. Five LEDs are connected to the system, and LEDs can be controlled individually as well as together using specific functions, such as making LEDs glow in ascending or descending order or increasing and decreasing brightness levels by implementing pulse width modulation (PWM) through the controls on the GUI.

Another page, accessible through the “12x2 LCD” link on the horizontal menu, displays the text on the LCD and lets the user change it remotely by entering his own text in a text field on the GUI and sending it to the display by clicking on a button. Users can see the change on the LCD live on the video stream. The LCD module is integrated with the Raspberry Pi using the Adafruit LCD library.

The “Ultra & Servo” link on the horizontal menu leads the user to another page, shown in Figure 5, in which the servo motor and the ultrasonic sensor are controlled. Some obstacles are placed around the ultrasonic sensor at various distances. The ultrasonic sensor is mounted on a servo motor to measure the distance to the obstacles at various directions. The servo motor is controlled either by angle or duty cycle that changes the angle as shown in Figure 5. Some default options for duty cycle and angle are provided for the servo motor to be controlled. The maximum angle of mobility of the servo motor is 180 degrees. The movement can be observed in the live streaming window.

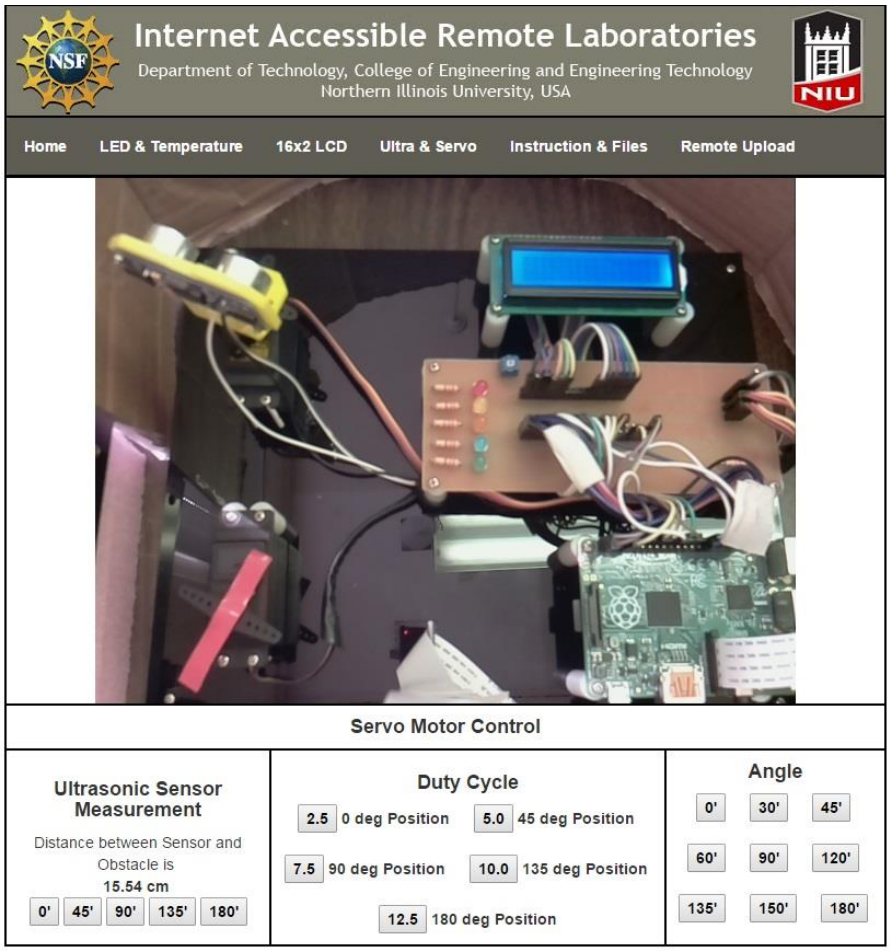

Figure 5: GUI for ultrasonic sensor and servo motor.

The “Instructions & Files” page provides the user with help about the module so they can start to develop their own programs. The page shows basic instructions as to how to operate the entire system and provides a document that illustrates how each GUI functions. The page provides a file that shows how each input/output device is connected to the Raspberry Pi’s General-Purpose Input/Output (GPIO). A separate file is provided for each device to clearly show its operation. Apart from example files for connected devices, other files include basics of Python, connection diagrams, and the code for each of the input and the output devices.

Another important feature of this system is remote upload and execution. The user can upload his own program to Raspberry Pi using the GUI shown in Figure 6. The upload form accepts only *.py format for it to execute the program file once it is uploaded. Once the “Execute” button is clicked, the uploaded file is executed and either the error or the program output is displayed in the “Execution Output” tab in the GUI. Response to the execution can be observed live in the video stream as well. The two graphs in Figure 6 show data from the temperature and the ultrasonic sensor captured over time. These data are logged into a file and are available for download by remote user to perform offline analysis of his experiment.

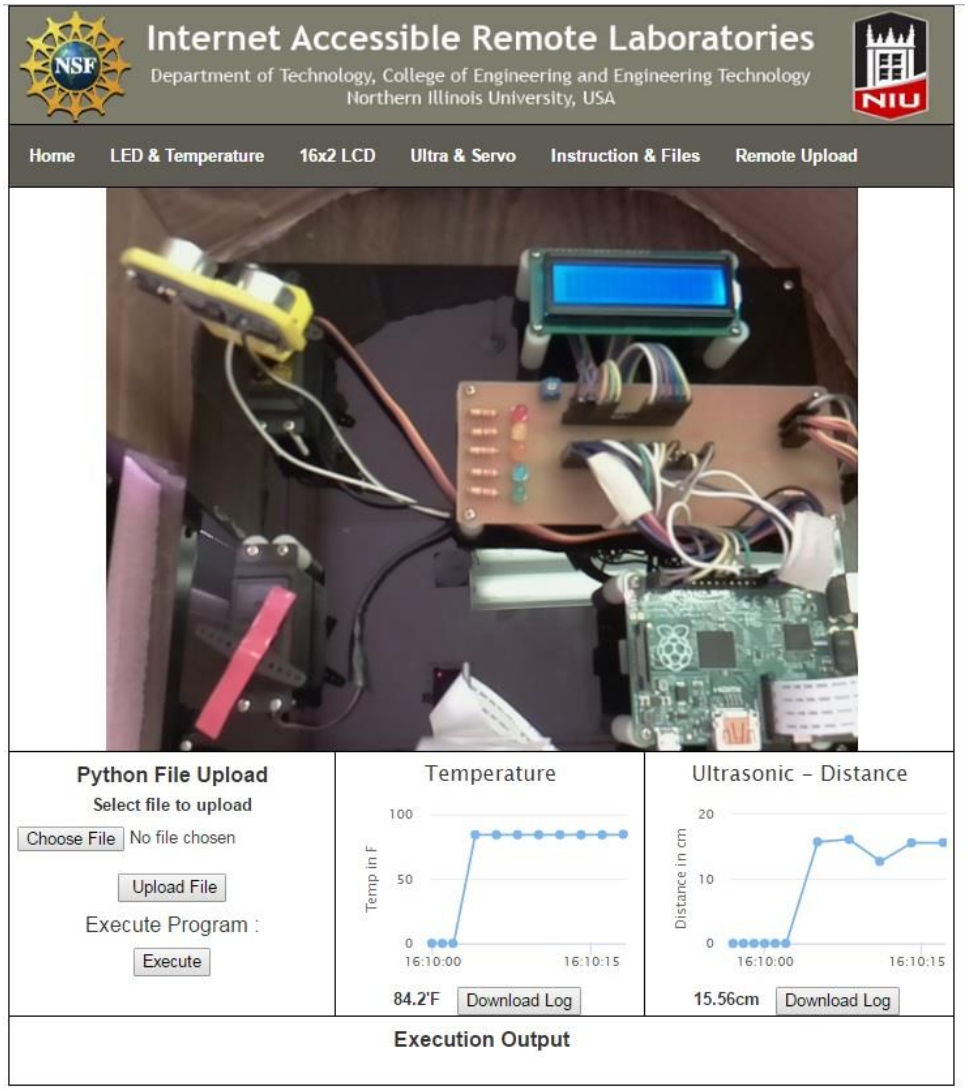

Figure 6: GUI for remote upload and data log.

4. Experiment 2: Controlling a Coupled Tank System

The second module in the remote laboratory system project is a Raspberry Pi integrated with a coupled tank system, shown in Figure 7, to control the water level in the tanks. The coupled tanks system serves as a process control experiment that provides students exercise various concepts they have learned about control theory, including transfer function representation, linearization, level control, PID, feed-forward, and control parameter tuning, etc.

The coupled tank system, as shown in Figure 7, consists of a water pump with two tanks. A pressure sensor is mounted at the bottom of each tank to measure the water level. The pump is used to drive the water from the basin at the bottom. The two tanks are mounted one below the other in such a way that the outlet of Tank 1 (top) flows into Tank 2 (bottom). The outlet from Tank 2 flows directly into the basin at the bottom of the system. The outlet of each tank is configurable and can be replaced by changing the insert that screws to the bottom of the tank to adjust the rate of flow as desired. The inserts are available in three sizes: small, medium, and large. The inlets for the tanks come from the pump and are split equally through Out 1 and Out 2 orifices. Tank 1 is also provided with a drain tap so water can flow directly into the basin. Each tank is additionally equipped with a vertical scale (in centimeters) alongside the tank to visualize the water level in each tank.

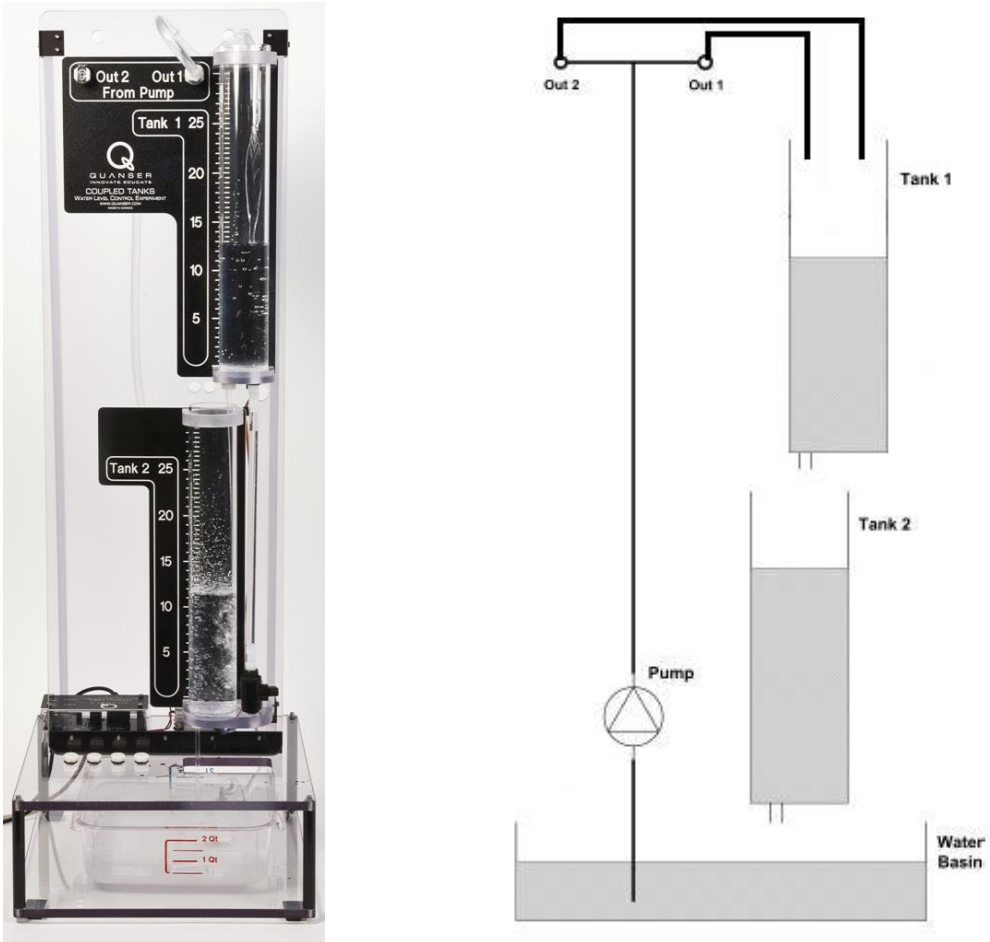

Figure 7: Images of the coupled tank system.

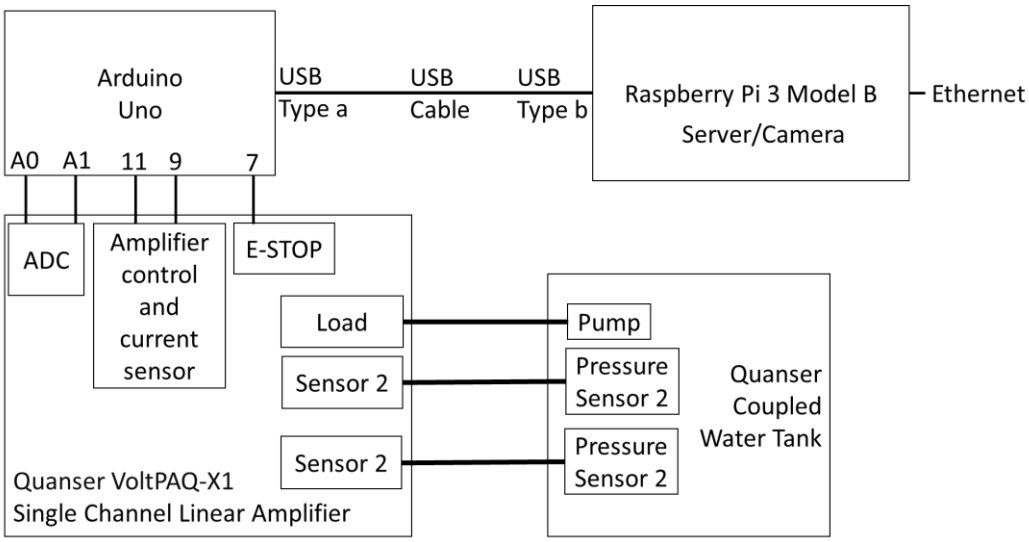

Figure 8: Schematic diagram of the designed system.

The schematic diagram, shown in Figure 8, represents how the coupled tanks are connected to the embedded system. In this configuration, an Arduino board was used as an intermediate stage between the Raspberry Pi and the control module of the coupled tanks. The pressure sensors mounted at the bottom of each tank produce analog readings that are converted into digital data by Arduino and sent to Raspberry Pi. A linear amplifier, controlled by Arduino, supplies power to the pump and activates it. As an additional feature, the amount of electrical current used by the pump is sensed and reported to Raspberry Pi through the Arduino.

It should be noted that in the experiment configurations presented below, the pump is operated in on/off mode. The pump can be operated with variable speed by controlling the power supplied by the amplifier, which will provide experiments to exercise concepts such as PID control.

Developed Experiments

The coupled tanks module is configured into 3 types of experiments, where each experiment is controlled with its own GUI on the module's webpage. The minimum and maximum values for the water level that can be maintained in the tanks are 1cm and 20cm. The water level is regulated to avoid overflow of the tank: A safety measure is incorporated in the system so, if by any chance the water level exceeds 20cm in the tank, the system automatically turns the pump off. Another safety measure is that if the experiment runs in an infinite loop, it can be stopped remotely by clicking an emergency stop button on the GUI.

Configuration 1

Configuration 1 is a Single Input Single Output System (SISO) in which the pump feeds into Tank 1. This configuration is designed to regulate the water level in Tank 1 and monitor it while Tank 2 is not controlled. The webpage shown in Figure 9 is used to control and monitor this experiment. The right column in the GUI allows the user to set the water level to be maintained in Tank 1 and the duration for which the experiment should run. It then shows the water level in each tank and the current drawn by the load (water pump) as graphical data while the experiment is running. The image on the bottom right side is the live video stream

that shows the progress of the experiment. Once the experiment is finalized, the water level and the load current data can be downloaded by the remote users for offline processing.

Configurations 2 and 3

Configuration 2 is a State-Coupled Single Input Single Output System in which the pump feeds into Tank 1, which in turn feeds Tank 2. This configuration is designed to regulate the water level in Tank 2. Configuration 3 is also a State-Coupled Single Input Single Output System in which the pump feeds into Tank 1 and Tank 2 using a split flow. In this configuration the water level in Tank 2 is regulated. The webpages for Configurations 2 and 3 are similar to that of Configuration 1.

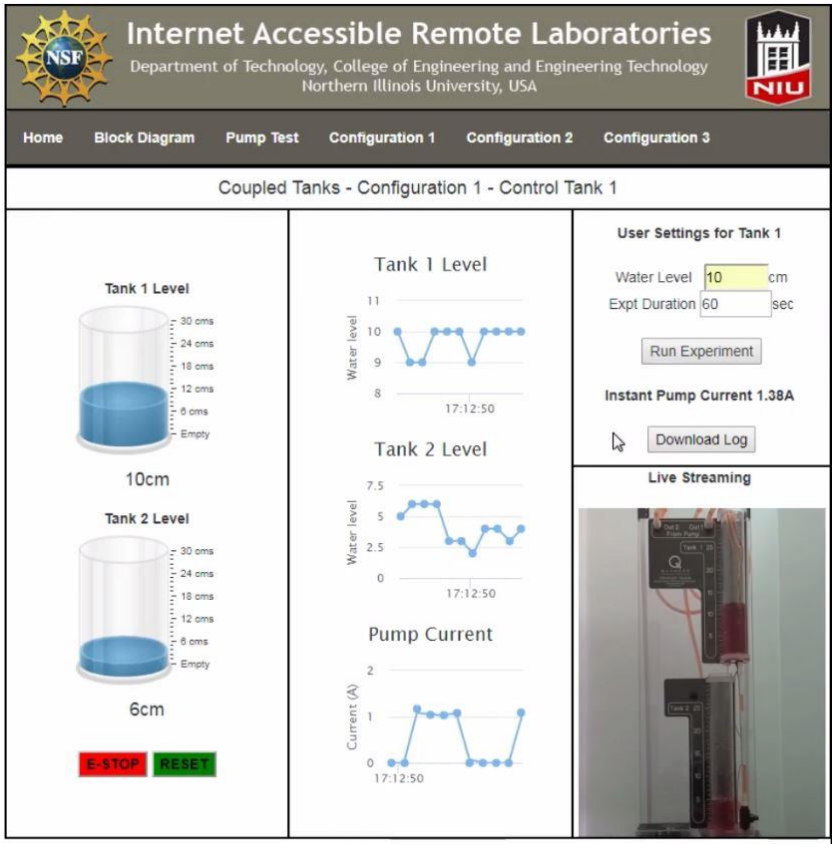

Figure 9: Configuration 1 interface

5. Conclusions

This paper presents two remote laboratory experimental systems designed and developed using the described strategy. One is a sensor actuator system the user can manipulate to perform certain laboratory experiments. The system also complements teaching Python programming for a Raspberry Pi over the web. Students can write their programs and upload them on the remote experimental system. The sensors and actuators are an array of light emitting diodes, a temperature sensor, a liquid crystal display, a servo motor, and an ultrasonic sensor. A suitable GUI was also developed so remote users can manipulate the controlled entities with little difficulty. The second experiment is a coupled tank system for performing process control experiments. The experimental system consists of two tanks on top of each other in such a way that the outlet of one tank flows into the other. The outlet from the bottom tank flows to a basin at the bottom. A motor is used to pump fluid to the top

tank. By controlling the motor one can control the fluid level in the tanks. The experiment involves the design and development of a feedback control system utilizing the pump and fluid level sensor outputs. The GUI includes the graphical representation of the fluid levels of the tanks, pump current, trigger buttons, and a video of the experimental system. Students can develop their own controller designs and upload them to the system to monitor performance.

6. References

[1] G. Kortuem, F. Kawsar, V. Sundramoorthy, and D. Fitton, "Smart Objects as building blocks for the Internet of Things," *Internet Computing, IEEE,* vol. 14, no. 1, pp. 44-51, 2010.

[2] R. Piyare, "Internet of Things: Ubiquitous Home Control and Monitoring System using Android based Smart Phone," *International Journal of Internet of Things,* vol. 2, no. 1, pp. 5-11, 2013.

[3] Custom Solutions Inc., "Custom Solutions Inc.," 2012. [Online]. Available: http://www.csi3.com/app_22.pdf.

[4] V. Lasky, "What are Remote Laboratories?," 2008-2016. [Online]. Available: https://remotelaboratory.com/remote-laboratories/what-are-remote-laboratories/.

[5] D. D. Magdum and R. D. Patane, "Raspberry Pi Based Remote Lab Implementation," *International Journal of Innovative Research in Computer,* vol. 4, no. 8, pp. 15181-15185, 2016.

[6] M. Kalúz, L. Cirka, R. Valo and M. Fikar, "ArPi Lab: A Low-cost Remote Laboratory for Control Education," *IFAC Proceedings Volumes (IFAC-PapersOnline),* vol. 47, no. 3, 2014, pp. 9057-9062.

[7] D. Fallon, "Survey of Existing Remote Laboratories used to Conduct Laboratory Exercises for Distance Learning Courses," in *2013 ASEE Annual Conference & Exposition,* Atlanta, 2013.

[8] Y. Limpraptono, A. A. P. Ratna and S. Harry, "New Architecture of Remote Laboratories Multiuser based on Embedded Web Server," in *Remote Engineering and Virtual Instrumentation (REV), 2012 9th International Conference,* Bilbao, Spain, 2013.

[9] I. Angulo, J. García-Zubia, P. Orduña and O. Dziabenko, "Addressing low cost remote laboratories through federation protocols: Fish tank remote laboratory," in *Global Engineering Education Conference (EDUCON), 2013 IEEE,* Berlin, Germany, 2013.