DEVELOPMENT OF A REMOTE OPERATIONAL AMPLIFIER iLAB USING ANDROID-BASED MOBILE PLATFORM

O. S. Oyebisi, O. B. Akinwale, K. P. Ayodele and L. O. Kehinde
Department of Electronic and Electrical Engineering
Obafemi Awolowo University
Nigeria

Abstract

iLabs are experimental setups that can be remotely accessed through the Internet using a web browser. They allow students and educators to carry out experiments from remote locations anywhere and at any time. Globally, more than a dozen iLabs have been developed and deployed. However they all collectively suffer from an increasingly important oversight: their interfaces are developed for desktops and laptop computers. This is considered an oversight because there is substantial evidence that mobile devices are emerging as the form factor of choice in the near future.

This paper describes a completely functional Android-based mobile Operational Amplifier iLab that will enable students all over the world perform experiments remotely from a mobile device using the Android platform. There are already other mobile platforms on the market today. However, Android is an environment that combines an open, free development platform based on Linux and gives good access to hardware. Android allows users to explore the mobile Internet afresh with its new features, easy access to the Internet, ease of development, new services and application. This makes it easy for a user/client to interact with the iLabs service broker and perform experiments. Android's openness and flexibility makes it a deciding factor over the closed iPhone framework that provides a similar set of features. This work serves as an improvement to the earlier research and work done in the area of mobile Laboratories under iLab.

Keywords: Android, iLabs, mobile Service Broker

Introduction

Online laboratories are experimental setups that can be accessed and performed over the Internet. With online labs, anyone can perform experiments from anywhere in the world at any time.

Online laboratories have several benefits. By making labs sharable online, the number of users of online labs scale up dramatically, particularly with the fact that online labs can be performed round the clock with no need for a physical lab attendant present at the lab for each lab session. Hence with the rising cost of undergraduate laboratory equipment and increasing undergraduate enrollment, online labs are a solution, a solution of particular import in the developing world [1].

iLabs are online laboratories which make use of the iLab Shared Architecture [2, 3]. The iLab project started at MIT and its aim was to create a movement to develop and disseminate technology for sustainable and scalable iLabs so that they could be shared worldwide. Globally, more than a dozen iLabs have been developed and deployed. However, all iLabs to date use clients that were developed for desktop PCs. With the recent surge in the use of tablets and mobile phones, and in particular, those running the Android and iOS operating systems, it has become necessary to develop clients that run on these devices.

This paper reports a completely functional mobile client application for iLab that will enable students to perform experiments remotely by consuming the iLab service broker web service from a mobile device. The client allows students to perform authentication, submit experiment specifications and retrieve experiment results. The client will communicate with the service broker web service which will in turn communicate with the Lab Server (consisting of Lab Server, web service and experiment engine) and the required experiment will be performed.

Previous Work

The iLab developed in this work is based on the original Op Amp iLab earlier reported by the authors [4], which was itself based on the "dozen impedance operational amplifier circuit" [5]. The original Op Amp iLab allowed students to remotely interact with, and configure simple operational amplifier circuits. The lab had six different experiment configurations namely: Non-Inverting Amplifier, Inverting Amplifier, Unity Gain Amplifier, Summer, Integrator and Differentiator. The user interface was developed using the C# programming language (Figure 1). The client allowed the user to select a configuration, connect wires from one node to another, and submit the experiment specification to the Lab Server for execution via the Service Broker. The Lab Server performed the experiment and sent the result back to the client program. The result was then displayed as a waveform. A more recent iteration used realistic interfaces developed on the Adobe Flex platform [6], but was also targeted at desktop computers.

While this work appears to be the first report of an Android-based client on mobile devices for the iLab architecture, there are a number of previous works that combine the first two elements, if not the third. Guerra et.al [7] have reported the development of PortableLab, a mobile remote laboratory for the Android platform. The lab was developed for the measurement of power supply quality. It makes use of a two-tiered architecture (a Lab Server and a Client), whereas this paper reports the development of an Android laboratory using the MIT iLab Shared Architecture, a three-tiered architecture.

Rationale for Employing Android

There are many mobile platforms, with the most popular being Android, iOS, Windows Mobile (in its various variants), Blackberry OS, and Symbian. Of these, the fastest growing is Android [8]. The operating system has a number

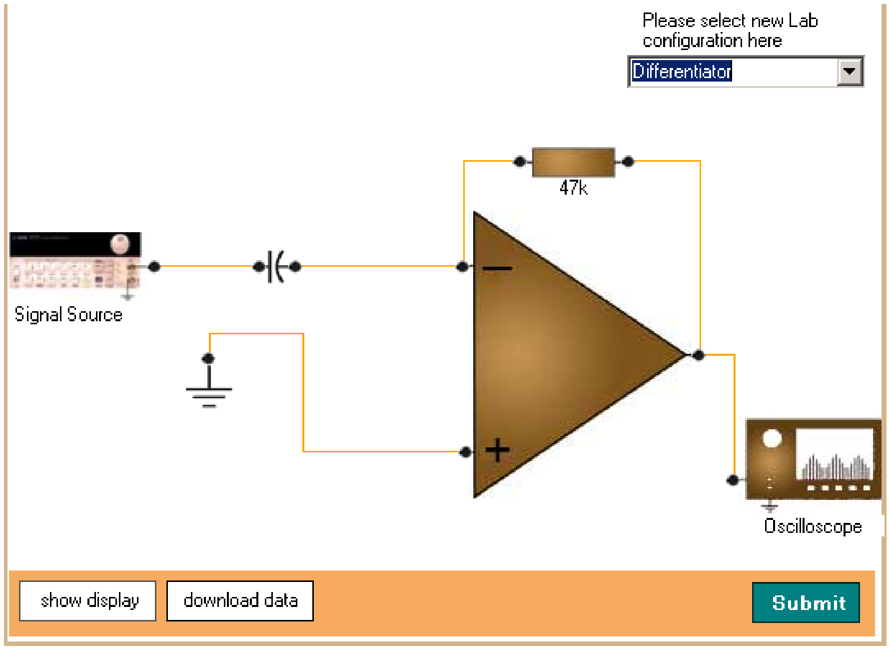


Figure 1: First Op-Amp iLab user interface [4].

of benefits that make it attractive to developers. It is reputed to be the first environment that combines:

1. An open, free development platform based on Linux and gives direct access to hardware, meaning the only "limitation is imagination" in designing Android software.

2. A component based architecture, which means parts of one application can be used in ways not originally envisioned by the developer, which makes it a very good platform for designing mobile clients for online labs, particularly as this is a fluid field with the technology not being completely mature yet.

3. Because of Android's openness and flexibility it is easy to produce a budget Android-based device to be sold in targeted areas – the deciding factor of choosing Android over the closed iPhone framework that provides a similar set of features. Hence, Android phones and tablets are likely to more quickly pervade the developing world than iOS devices. It is still too early to call on Windows 8 RT devices.

Other advantages of the Android system are:

1. Android allows access to core mobile device functionality through a well-developed set of standard API calls.

2. Android is unbiased between the phone's core applications and third party applications. Hence, a developer can write an application to replace the phone's home screen or even its dialler.

3. Android seamlessly integrates data from the web with data from the phone to give a rich user experience. Hence, applications can be written to use the phone's current location to pull up restaurants close to it.

4. Android applications can be developed reasonably quickly and easily. A Software Development Kit (SDK) is provided. The SDK includes a true device emulator and advanced debugging tools.

In addition, Android features the following characteristics that make it an ideal platform for remote experimentation [9, 10]: the Davlik virtual machine, a low-memory virtual machine designed to work well in low power situations; adaptable graphics, connectivity, hardware support, and a very healthy developer ecosystem with mature development environments.

Android is not without its disadvantages. Two of them are fragmentation and security. Due to the fact that mobile device manufacturers are free to modify Android as they see fit, there is a real danger that fragmentation may result, with the attendant inability of apps written for one dialect of Android to run on other dialects. This fear has not materialized, and Google appears to be taking steps to prevent it. Another drawback to the open nature of the platform is the danger that stealthy Trojans, spyware, and other forms of intrusive, disruptive or destructive software may be much easier to develop. An example could be using the GPS feature of the mobile device to track a person's location without their knowledge.

THE MIT ILAB BATCHED ARCHITECTURE

The iLab shared architecture (ISA) was developed by MIT to "facilitate the rapid development and effective management of iLabs" [11]. To this effect, several toolkits and reusable modules were developed as well as a set of standardized protocols and web services.

The iLab batched architecture is a three-tiered architecture consisting of the Client, Service Broker and the Lab Server [2, 3]. Figure 2 shows the iLabs batched architecture. These three tiers are connected together using web services. iLabs' design separates online labs into

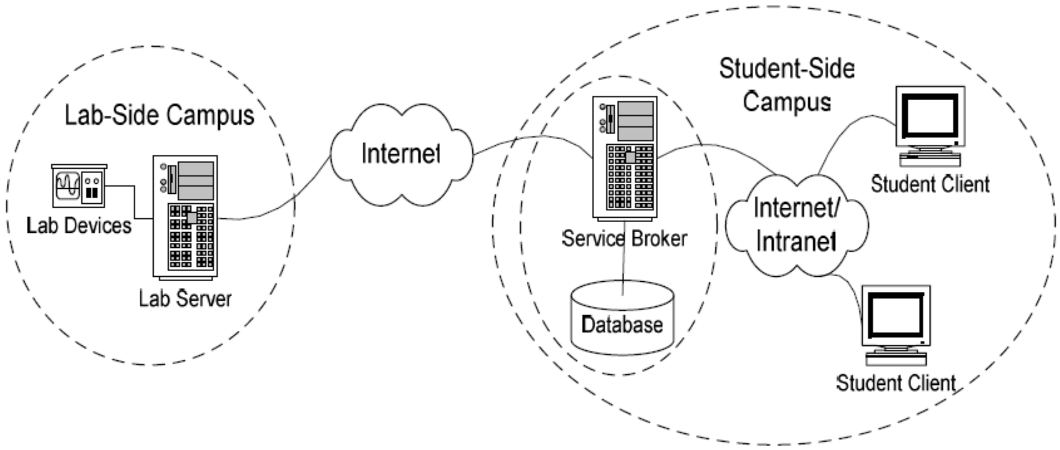


Figure 2: iLabs Architecture Overview [12].

three distinct modules connected by a web service architecture.

The Client

The Client is usually an application that runs in a browser, though it can also be a standalone application, which the user interacts with to configure experiments, send experiment specifications and retrieve results from the Lab Server via the Service broker. The client is the front end of the system. The client program interacts directly with the Service broker through web services.

The design and development of a functional Mobile Client for iLab is the aim of this project and it is an improvement over the originally used clients for desktop computers.

Lab Server

The Lab Server is the backend, a server connected to the remotely located lab equipment (or laboratory model in the case of a virtual lab). It executes the experiment based on the experiment specification and notifies the Service Broker when the result is ready for download. In remote labs, the hardware may be an electronics lab built on the National Instruments ELVIS platform. A data acquisition system is used to get the result hardware setup.

The Service Broker

The Service Broker is the middleman that links the client with the Lab Server. The Service Broker performs administrative functions; it takes care of authentication and the authorization of users, user sessions, and experiment data storage, forwarding experiment specifications to the Lab Server and retrieving the result. The Service Broker website is where the user registers, logs-in, and stores experiment data (experiment specifications and results) to be used by the lab client.

A Service Broker was developed by the iLab team at MIT and this ships with the ISA, but for this project, a mobile Service Broker was designed which is similar in functionality but yet differs (by written code) from the originally developed service broker.

THE MOBILE LABORATORY ARCHITECTURE AND DESIGN

The interface of the Android Op Amp iLab was designed to have a power supply unit, a breadboard and an oscilloscope. The former two were to be used by the user for inputting his experiment specifications. The last was for viewing the results gotten from the experiment. The client was designed to consume the service broker web service which involved allowing a user to authenticate, submit experiment specifications, and retrieve experiment results.

Two factors informed the redesign of a new mobile Service Broker for this project instead of using the downloadable Service Broker which comes with the ISA. Firstly, many parts of the iLab ISA's Service broker were not needed for this project. Secondly, the iLab ISA's Service Broker was not optimized for consumption by an Android application. Web applications designed for consumption by touchscreen mobile devices ought to be designed with minimal text and optimal sizes of text and textboxes.

For remote labs deployed on computers, the client is often a Java applet or a C# application or a Flash application or a LabVIEW application embedded in a webpage creating what is called a thin client with the client residing on the Service Broker or Lab Server and not on the client's computer. In the case of mobile applications, the mobile application can be designed to take advantage of the mobile user's hardware such as touchscreen, accelerometers, gyroscopes and GPS. To do this, the client must reside on the user's mobile device.

The iLab Mobile Architecture for Android

The architecture of the Android Op Amp iLab is shown in Figure 3. This architecture, though similar to the original iLab architecture is subtly different: instead of the client (the mobile application) residing on the Service Broker website and being deployed from there, it resides permanently on the lab user's mobile phone. Whenever the user wants to run an experiment he launches the client on his device and logs in to the Service Broker with it, after which, he is able to run the experiment on his device.

Mobile Service Broker

The mobile Service Broker, just like the normal Service Broker, is for authentication, experiment submission and result retrieval. The mobile Service Broker is a web service created using the Microsoft .NET framework and was created with four web methods: *AuthenticateUser*, *Submit*, *GetExperimentStatus* and *RetrieveExperimentResult*.

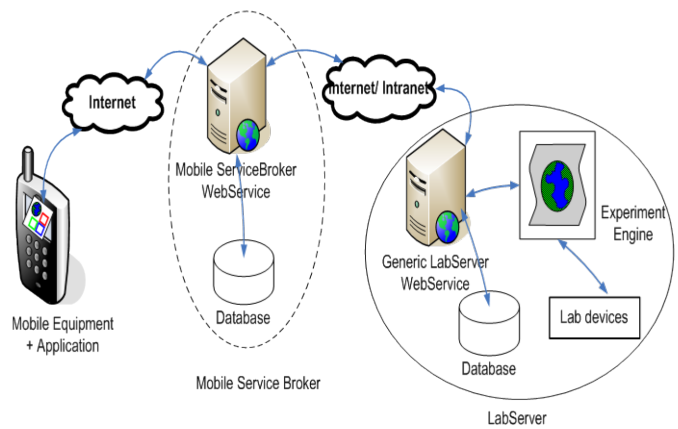


Figure 3: The modified iLab architecture.

The *AuthenticateUser* method is used to verify if the person attempting to run the experiment is a registered user. It takes in two string parameters: Username and Password. The method checks the mobile Service Broker database for the supplied username; if found, it retrieves its corresponding password and compares with the Password string entered by the potential user.

The *Submit* web method allows the user to submit an already prepared experiment specification to the Lab Server for execution. The method takes in two parameters, the experiment specification string to be submitted and the identifying string for the Lab Server to which the experiment is to be submitted. When the *Submit* method is invoked with valid arguments, a copy of the experiment specification is saved in the mobile Service Broker database and the mobile Service Broker submits the experiment specification to the Lab Server by invoking a corresponding *Submit* method on the Lab Server.

GetExperimentStatus is used to monitor the state of the experiment being performed. The experiment takes in two parameters, the experiment identification integer which is a positive integer unique to every experiment and the Lab Server identification string of the Lab Server to which the experiment has been submitted. An integer, from 1 to 7, representing the status of the experiment is assigned to each experiment on the Lab Server so that when this method is invoked it checks the Lab Server database for the experiment using the experiment identification integer and returns the status which is represented by the integer assigned to it. The possible states for an experiment are shown below with the integers representing them.

1- Experiment waiting in queue.
2- Experiment currently running.
3- Experiment finished running and terminated normally.
4- Experiment terminated with errors.
5- Experiment cancelled by user.
6- Experiment not found/ unknown experiment identification.
7- Invalid experiment.

After retrieving the status, it is returned to the mobile client which proceeds execution according to the experiment status returned.

The *RetrieveExperimentResult* web method is used to obtain the results of experiments that have finished execution. When the status of the experiment being performed is either 3 or 4 (execution of experiment has been completed) the *RetrieveExperimentResult* can be invoked in order to download the experiment result from the lab server to the client. This result also takes in two parameters, the unique experiment identifier integer and the identification string of the lab server to which the experiment was submitted. When this method is invoked it checks the lab server database for the experiment and when found it takes the corresponding experiment result and saves a copy on the mobile Service Broker database before returning it to the client performing the experiment.

Algorithm and Flow Chart

Figure 4 is the flowchart for the *Authenticate User* method that was implemented. The approach to make the call for the method is as follows:

- Input parameters were set (client side) and parsed into the experiment specification.
- The Experiment specification and Lab Server ID were wrapped into objects by the kSOAP class; PropertyInfo, the objects were then placed in the soap envelope.
- The kSOAP class; HTTPTransportSE was used to open a connection to the web service and set the soap headers and envelope into the soap message after which they were sent.
- Another kSOAP class: SoapPrimitive, was used to retrieve the response to the SOAP request from the mobile Service broker web service.
- The received response was parsed using SAXParser (DOMParser or any other parsing mechanism could be used).

The steps above also constitute the basic method approach that is used for each call to the web service.

Mobile Op-Amp Client Application Layout

The Op Amp IC is mounted on a soft realistic interface which is similar to the traditional experiments. Figure 5 shows a blank soft breadboard with Op Amp IC being used.

Figure 6a shows the login activity screen for the mobile client application. On launch the user inputs username and password for authentication purposes. On successful login, the user is confronted with three menu options: start menu, about menu and exit menu as shown

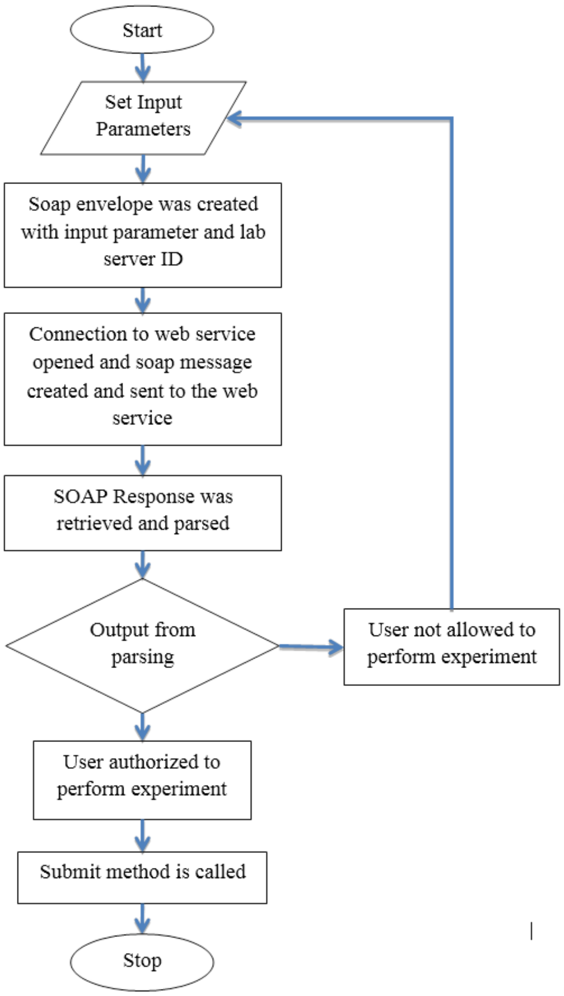


Figure 4: Flow chart for AuthenticateUser method call.

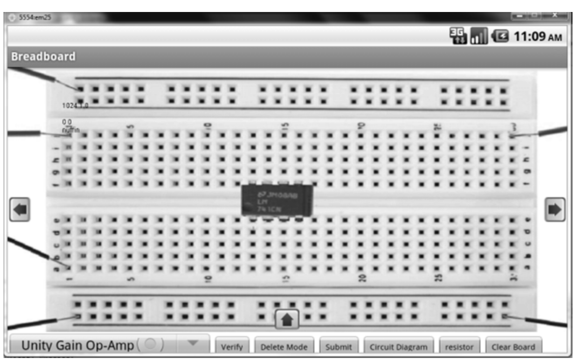
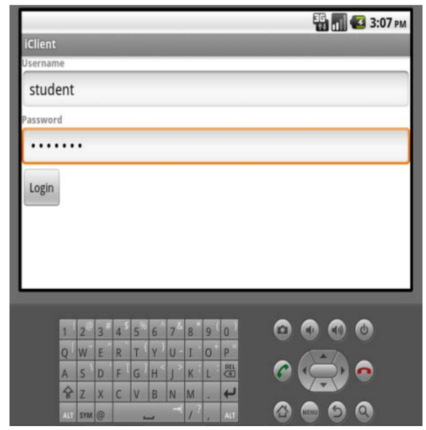


Figure 5: The Bread Board Soft Interface.

in Figure 6b. The main features were brought into a single application, launched on clicking start, and they were placed under three corresponding flip views:

- The voltage supply - this allows the user to set input voltage parameters;
- The breadboard – for selecting circuit configuration and making connections; and
- The oscilloscope – for displaying the result of the experiment.



(a)

(b)

Figure 6. (a) Login Activity Screen (b) Client Menu Options.

The three features are connected as shown in Figure 7.

Performing Experiments

Four Op Amp configurations were implemented in the application design, they are: Unity-Gain Amplifier, Inverting Amplifier, Non-Inverting Amplifier, and Difference Amplifier. Their corresponding schematics are shown in Figure 8.

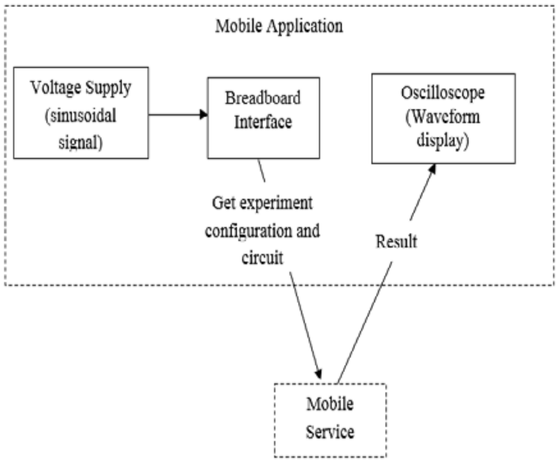
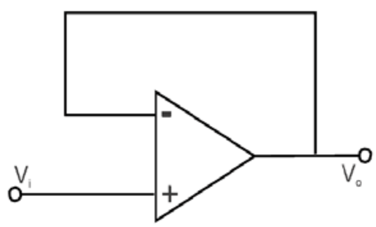
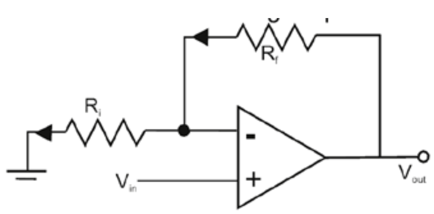


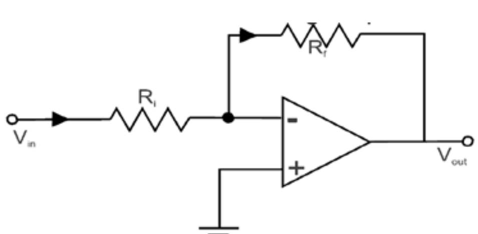
Figure 7: Op-Amp client: Interconnection of main features.



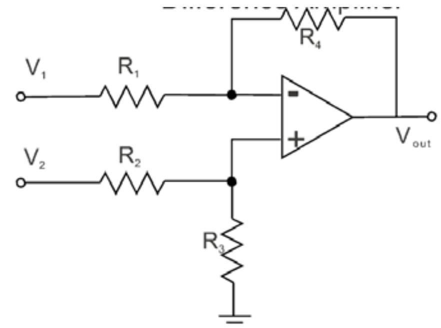
(a)



(b)



(c)



(d)

Figure 8: Op Amp configurations supported by the mobile Op Amp iLab Client (a) Unity-Gain Amplifier (b) Non-Inverting Amplifier (c) Inverting Amplifier (d) Difference Amplifier.

In order to perform an experiment, the user inputs the appropriate DC voltage on the power supply interface. On the breadboard interface, the user selects one of the four Op-Amp configurations he wishes to experiment with and makes the appropriate circuit connection, and then he submits the experiment. If an internet connection is available on the android device, the experiment specification is prepared using the input parameters and an attempt to submit is made. The breadboard interface for the unity gain amplifier is shown in Figure 9.

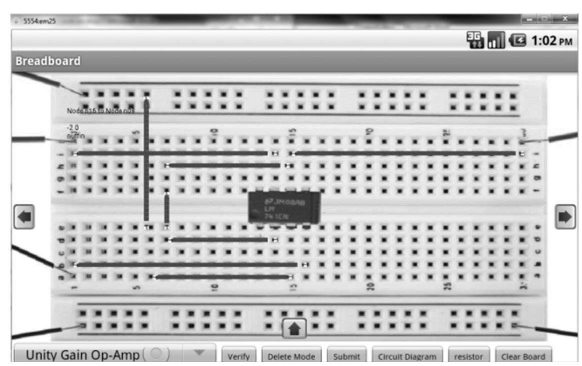


Figure 9: Breadboard in Unity-Gain Amplifier Configuration.

Connections are made on the breadboard by making use of the touch screen capability to draw cables from one point on the breadboard to another. It was designed to have

- A drop down menu for selecting one of the 4 op-amp configurations.

- A verify button – To check if the circuit has been connected properly and is in line with the Op-Amp Configuration chosen.
- A delete button – To enable the user delete unwanted connections and resistors
- A submit button – After proper connection, the submit button is clicked and the experiment is sent to the service broker.
- Circuit Diagram button- It allows the user to view the circuit connection of the Op-amp configuration chosen.
- Resistor button: To enable the user to select resistors to use for connections.
- A Clear Board button – This takes the breadboard interface to the initial/default state.

Displaying results

The *Submit* method (for submitting an already prepared experiment specification to the lab server for execution) is called on clicking the *Submit* button on the breadboard interface. After execution, a copy of the result is placed on the service broker and sent to the client. The result when returned to the client is displayed on the oscilloscope interface as a graph. The oscilloscope interface is as shown in Figure 10.

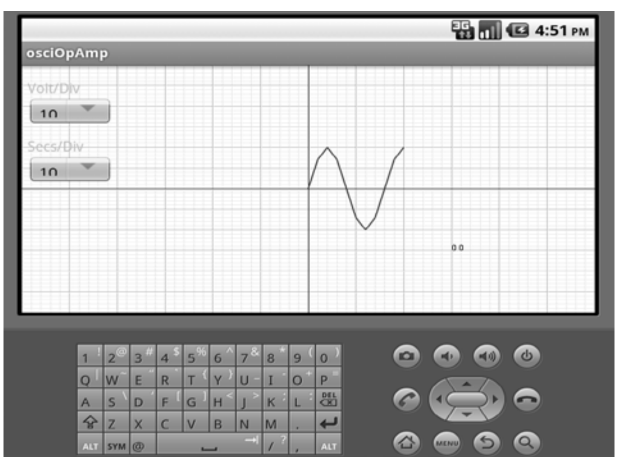


Figure 10: The Oscilloscope Interface.

Implementation And Testing

The Android Client for the Op Amp iLab was developed and tested in the Eclipse IDE. Some tests have also been carried out on a 10-inch Samsung Galaxy Tab.

Testing the Main Features.

For its Android platform, Google provides a well-put together software development kit (SDK), a plug-in for Eclipse IDE, and a number of tools to aid application development. The tool mostly used in this project was, naturally, the emulator. Some parts, such as the oscilloscope were tested on the HTC Desire – An android mobile device which at the point of writing was running Android version 2.2 (Froyo) - and a Samsung Galaxy 10-inch tablet . A screen shot of the emulator is as shown in Figure 11.

A thorough evaluation of how the application would behave under possible circumstances was performed. The circumstances were divided into two main areas: Performing the experiment and obtaining results.

Figure 11: Screenshot of the Emulator.

Performing Experiments

The following circumstances would result in premature termination of the experiment:

- When the circuit connection does not match the Op-Amp configuration chosen; or
- When the circuit is wrongly connected.

If the circuit is properly connected, on clicking the verify button, the application is designed to show “Connection Ok”, if not, that is, if there is an error in connection or a wrong configuration is selected, the verify button helps to localise the fault on the breadboard. The exact location of the error on the breadboard will be indicated.

Obtaining Results

Error pertaining to this area is seen when the device is offline, one of the assumptions made is that the device is connected to the internet, as it needs to access the service broker via the internet. Hence, if there is no connection or the data source is unavailable, a time out occurs.

Future Development

Other Op Amp configurations such as the Summer, Integrator and Differentiator can be added to existing configurations though the problem of hardware restriction has to be overcome before this can be possible. Another development that could be achieved on solving the hardware limitation issue is designing two or more voltage input channels on both the breadboard and the voltage supply interface.

On the breadboard, pinch and zoom are being implemented so that the application can be used conveniently on mobile phones and not just tablets. The challenge in implementing pinch-to-zoom stems from the fact that a drag gesture is used to connect wires across points on the breadboard and if combined with pinch and zoom which uses the drag feature also, it would interfere and cause the application to crash. Hence a new method has to be used for connecting wires across points on the breadboard.

Limitation of Mobile Laboratories

One limitation to using a mobile client to access iLab is network availability. Network downtime may render a laboratory unavailable to students who urgently need to work on it. Also low bandwidth may lead to slow response from the remote server and an unsatisfactory student experience. Although the bandwidth situation in Nigeria is improving, it still remains an area of concern

Another limitation is the security of the mobile platform in use. As Android is open source i.e. source code is available to everyone, which includes black hat hackers who develop stealthy Trojans hidden in animated images and particular viruses although mobile protection in the form of anti-virus, anti-spam, firewall software is up and running on the Android Operating System.

References

1. Kehinde, L. O., Chen, X., Ayodele, K. P., & Akinwale, O. B. (2012). Developing Remote Labs for Challenged Educational Environments. In A. Azad, M. Auer, & V. Harward (Eds.), *Internet Accessible Remote Laboratories: Scalable E-Learning Tools for Engineering and Science Disciplines* (pp. 432-452). doi:doi:10.4018/978-1-61350-186-3.ch022

2. Harward, V., Del Alamo, J., Choudhary, V., DeLong, K., Hardison, J., Lerman, S., Zych, D. (2004). iLab: A Scalable Architecture for Sharing Online Experiments. *International Conference on Engineering Education, ICEE2004.* Gainesville, Florida.

3. Harward, V. J., Alamo, J. A., Lerman, S. R., Bailey, P. H., Carpenter, J., Delong, K., Zych, D. (2008). The iLab shared architecture: A web services infrastructure to build communities of internet accessible laboratories. *Proceedings of the IEEE.* IEEE.

4. Ayodele, K. P., Kehinde, L. O., Jonah, O., Ilori, O., Ajayi, E. O., & Osasona, O. O. (2008). Development of an Operational Amplifier Virtual Laboratory Based on iLab Architecture and NI ELVIS. *ASEE Annual Conference and Exposition* (pp. AC 2008-1098). Pittsburgh, PA: American Society for Engineering Education.

5. Kehinde, L. O. (1989). The “Dozen-Impedance” Operational Amplifier Module for Experimentation. *International Journal of Electrical Engineering Education, 26*(3), 224-232.

6. Ishola, B. I., Ayodele, K. P., Kehinde, L. O., Akinwale, O. B., & Aboluwarin, O. O. (2012). An Improved Operational Amplifier iLab with a Realistic Looking Interface. *ASEE Annual Conference & Exposition* (pp. Paper AC 2012-4608). San Antonio: ASEE.

7. Guerra, M. A., Francisco, C. M., & Madeira, R. N. (2011). PortableLab: Implementation of a Mobile Remote Laboratory for the Android Platform. *Global Engineering Education Conference (EDUCON), IEEE* (pp. 983-989). Amman, Jordan: IEEE. doi:10.1109/EDUCON.2011.5773266

8. Sharma, K. (2011). Android in opposition to iPhone. International Journal on Computer Science and Engineering (IJCSE), 3(5), 1965-1969.

9. Google Inc. (2012). *Gingerbread*. Retrieved November 18, 2012, from http://developer.android.com/about/versions/android-2.3-highlights.html

10. Google Inc. (2012). *Honeycomb*. Retrieved November 18, 2012, from Android Developers: http://developer.android.com/about/versions/android-3.0-highlights.html

11. MIT. (2011, January 24). *iLab: Remote Online Laboratories*. Retrieved November 18, 2012, from iCampus Project: http://i-campus.net/projects/iLabs.shtml

12. Ayodele, K. P., Akinwale, O., Kehinde, L. O., Osasona, O., Ajayi, E. O., & Akinwunmi, O. O. (2009). Advanced Digital Laboratory: An FPGA-Based Remote Laboratory for Teaching Digital Electronics. Proc., ASEE Annual Conference & Exposition. Paper AC 2009-1206. Austin , Tx: ASEE.

Biographical Information

Oyediran Samuel Oyebisi is a graduate of the Obafemi Awolow University, Ile Ife. He received his Bachelor of Science Degree from the department of Electronic and Electrical Engineering in 2012. He was a part of the OAU iLAB team for over three years while he was an undergraduate. His passion is mobile software development especially for the Java ME and Android platforms. He can be contacted at samueloyebisi108@gmail.com.

Olawale Akinwale earned his first degree at the Department of Electronic and Electrical Engineering, Obafemi Awolowo University, Ile-Ife, finishing with first class honors in 2004. He obtained his second degree from the same department in 2011. He currently lectures in the same department. He is also a lab developer in the OAU iLab Research Group. He developed the first reported robotic arm remote lab in Africa making use of the MIT iLab shared architecture and National Instruments LabVIEW. His interests include online experimentation, methods in enhancing pedagogy, machine learning and artificial intelligence, and home automation.

Kayode P. Ayodele obtained his Ph.D. and MSc degrees in Electronic and Electrical Engineering from Obafemi Awolowo University, Nigeria, in 2012 and 2005 respectively and is currently a member of the faculty at the same university. As a member of the OAU iLab research group, he has been working on developing and studying remote laboratories since 2005 and is particularly interested in usability and reliability issues. Other research interests of his include acquisition and processing of neurophysiological signals.

Professor Lawrence Kehinde, a Professional Engineer, is a Professor of Electronic and Electrical Engineering and the Coordinator of the Remote Lab Development Group of the Obafemi Awolowo University (OAU), Ile-Ife, Nigeria. He had worked in a Techno-Managerial position as the Director of ICT at OAU for years. His major field is Instrumentation Designs and he has designed various equipment. He currently designs experiments for remote and virtual experimentation in Science and Engineering and was the founding Principal Investigator of the University’s iLab research. He is at present the Coordinator of a State Research and Educational Network in Nigeria.