AC 2009-745: IMPLEMENTATION AND EVALUATION OF A LINEAR AXIS RAPID DEVELOPMENT SYSTEM

Mike Fleming, Missouri University of Science and Technology

Vedant Jain, Missouri University of Science and Technology

Robert Landers, Missouri University of Science and Technology

Hong Sheng, Missouri University of Science and Technology

Richard Hall, Missouri University of Science and Technology

Implementation and Evaluation of a Linear Axis Rapid Development System

Abstract

In most control courses the topic of feedback control is introduced at a theoretical level. A typical assignment, for example, is to design a controller (i.e., compute controller gains) to regulate the position of a linear axis given a very simple model of the linear axis. The student may conduct a simple simulation; however, they will probably not be able to implement the controller on physical hardware. In this situation the student misses opportunities to 1) explore the effects saturation, quantization, nonlinear friction, and sample period have on their controller and 2) investigate real physical results. These opportunities are lost due to the fact that the time required for the student to interact with the hardware is prohibitive in a traditional semester course.

This paper presents a Linear Axis Rapid Development System (RDS), based on Simulink, which provides the student with a tool to explore all phases of controller development after the theoretical work is complete. When the student has an algorithm that is ready to implement, they encode the algorithm as a subsystem in Simulink. The controller inputs and outputs, along with their engineering units, are carefully specified. The student then utilizes the Linear Axis RDS to analyze the controller. The Linear Axis RDS has three modes: simulation, emulation, and implementation. In the simulation mode the student simulates a linear axis system that includes their controller and detailed models of the interface hardware and linear axis. In the emulation mode, the simulation is performed on the computer hardware that will implement the controller. In this mode the student can ensure their algorithm will run in real time (i.e., the algorithm's execution time is less than the sample period). In the implementation mode, the controller is deployed on the hardware system and experimental data is gathered. This paper presents the results of the implementation of the linear axis RDS in a manufacturing automation course and initial usability studies, aimed at assessing its effectiveness as a learning tool.

Introduction

The process of designing a controller to govern the actions of a piece of equipment requires careful controller algorithm design and precise tuning of the controller parameters. For a student, the implementation of a controller on a physical system can be complicated and time consuming, and can distract from controller design and analysis. A software program, which minimizes the amount of effort required by the student to implement a controller, is proposed in this paper. This software program is called a Rapid Development System (RDS). The goal of the RDS is to allow the student to remain focused on controller design and analysis. This is done by creating an environment where the software can be rapidly reconfigured to integrate the student's controller. There are three modes of controller operation used to analyze the controller performance: simulation, emulation, and implementation. In the simulation mode the student simulates a linear axis system that includes their controller and detailed models of the interface hardware and linear axis. In the emulation mode, the simulation is performed on the computer hardware that will

implement the controller. In this mode the student can ensure their algorithm will run in real time (i.e., the algorithm's execution time is less than the sample period). In the implementation mode, the controller is deployed on the hardware system and experimental data is gathered. The RDS provides the interface needed to operate the controller in these three modes.

There has been an abundance of work in developing hardware control laboratories including, to name only a few, double tank system [1], inverted pendulum [2], inverted double pendulum [3,4], triple inverted pendulum [5], and ball and beam [6], ball and beam on a roller [7]. However, these laboratories did not provide the ability for students to conduct emulation studies, nor did they provide a software interface that allowed the students to quickly transition between simulation, emulation, and implementation studies.

This paper develops a RDS for a linear axis. The Linear Axis RDS was implemented in a manufacturing automation course and initial usability studies, aimed at assessing its effectiveness as a learning tool, were conducted.

Linear Axis Rapid Development System

A Linear Axis RDS is developed and applied to the x–axis of the mini–CNC machine (Figure 1). A host computer, which has a non real–time operating system, runs the Linear Axis RDS Graphical User Interface (GUI), as well as the controller and models in simulation mode. A target computer, which has a real–time operating system, runs the controller and models in emulation and implementation modes. The target computer has a National Instruments (NI) 6711 digital to analog (D/A) output board and a NI 6602 counter–time (C/T) board. The target computer, which utilizes the xPC real–time operating system, sends voltage commands through the output board and receives encoder measurements through the C/T board, both at a sampling rate of 1 kHz. The output voltage range is ±10 V and is amplified before reaching the DC motor by a factor of 2.4. The motor has a gear ratio of 20/1 from the motor shaft to the output shaft. The output shaft is connected to a lead screw with a pitch of 5.9 mm/rev, which translates the linear axis. The encoder, which has a resolution of 500 counts/rev, measures the motor shaft displacement. The C/T board is run in quadrature mode for an effective encoder resolution of 2000 counts/rev.

The student must create the controller in the Simulink syntax, which is a visual syntax consisting of blocks and connectors that can be used to create the student's controller and operate the linear axis. Three Simulink models allow the student to operate the controller in three different modes: simulation, emulation, and implementation. These modes are described below.

Simulation mode allows the student to run the Simulink model on the host computer with a model simulating the linear axis dynamics. This is beneficial because the student does not need to be connected to the physical linear axis to perform simulations. The Simulink model used for simulation mode, shown in Figure 2, contains four subsystems: Reference Generator, Controller, Computer–System Interface, and Linear Axis Model. The Reference Generator subsystem contains code that generates the desired linear axis position, called the reference position, which is sent to the controller. The Controller subsystem contains the student's controller or a controller

selected from a menu. The Controller subsystem receives the reference and measured positions from the Reference Generator and Linear Axis Model subsystems, respectively, and sends the command voltage to the Computer–System Interface subsystem. The Computer–System Interface subsystem, shown in Figure 3, contains the simulated quantization and saturation affects of the D/A output board and the simulated quantization affect of the encoder. The Linear Axis Model subsystem, shown in Figure 4, contains the linear axis dynamic model. The model calculates the linear axis position given the command voltage from the Computer–System Interface subsystem. The reference position, measured position, and commanded voltage time histories are saved and used for controller performance analysis.

Emulation mode allows the student to run their controller on the target computer in real–time without moving the linear axis. The linear axis is simulated on the target computer. Emulation allows the student to determine if the target computer is able to perform all computations and perform send and receive tasks within the specified sample period. The Simulink model used for the Emulation mode contains the same four blocks as the Simulink model used for Simulation mode. The difference between the Simulink models lies in the Computer–System Interface subsystem. The output voltage sent to the physical linear axis is zero and the position measurements from the encoder are disregarded as shown in Figure 5. These send and receive blocks in the Computer–System Interface subsystem are used to determine the time required to perform the communication tasks on the target computer. Since the linear axis is only one component of the mini–CNC, zero voltage signals are also sent to the other components to ensure they do not move. The same controller used in Simulation mode is also used in Emulation mode.

Implementation mode allows the student to operate the hardware system using their controller. The Simulink model used for Implementation mode, shown in Figure 6, contains three subsystems: Reference Generator, Controller, and Computer–System Interface. The difference between the implementation and emulation modes is 1) there are no simulated affects in the Computer–System Interface subsystem and 2) there is no model to simulate the linear axis. This is because these affects are present in the computer interface hardware and physical linear axis and, thus, do not need to be simulated. Within the Computer–System Interface subsystem the command voltage from the controller is sent to the D/A output board and the encoder measurements are received from the C/T board as shown in Figure 7. The encoder measurements are used to calculate the measured linear axis position, which is sent to the controller. The same controller used in Simulation and Emulation modes is also used in the Implementation mode.

The Simulink model for each mode, referred to as a Simulink mode model, correlates with a text based code (i.e., source code) read by Matlab 7.1 to generate the blocks and connections of the Simulink mode model. It is possible to alter the contents of a Simulink mode model by changing its source code. A program, referred to as the model–altering program, is developed to reconfigure the Simulink mode model source code to contain the controller selected by the student. The model–altering program reads the selected controller, finds the location of the corresponding subsystem in the Simulink mode model, and inserts the student's controller into the Simulink mode model at that location. Therefore, the student only has to create a Simulink program with one subsystem; namely, the controller. This allows the student to concentrate on controller design and tuning instead of writing and debugging Simulink code. A specific

Simulink format for the student's controller is used. The student programs his/her controller within a subsystem named "Controller" containing two inputs, reference and measured positions (in meters), and one output, the commanded voltage (in volts). This is the same format as the Controller subsystem shown in the Figure 4.

A Graphical User Interface (GUI) was developed to allow the student to select the controller and mode, and view the results. The GUI visually displays desired subsystem content options the student can select using the host computer. The order in which the options appear, the location of the options on the host computer screen, and the content of the options must be designed in a way to allow the student to easily select the content of each subsystem and easily analyze controller performance results. The GUI was created using the Matlab program GUIDE. The GUI consists of two pieces: the visual display and the callback code. The program GUIDE provides buttons and drop down lists that are positioned on the visual display according to the designer's preference. The buttons and drop down lists must be linked to the callback code to perform the automated controller implementation. The program GUIDE generates callback code that is modified by the designer to create the GUI.

Phase 1 Linear Axis RDS

The RDS is designed in two phases. This paper describes the design and implementation of the first phase, as well as initial work for the second phase. The goal of the first phase is for a student to be able to easily simulate, emulate, and implement a controller for a linear axis. This is done by creating a Simulink model for each mode that acts as the interface between the student's controller and the linear axis, and by creating a GUI that allows the student to select the mode and controller using the host computer.

In Phase 1 the reference is a sinusoid with an amplitude of 20 mm and a frequency of 0.125 Hz. The controller used in the Controller subsystem is selected by the student. The simulated D/A quantization is 4.88 mV, and the simulated encoder quantization is 0.1475 μm. The model used to describe the linear axis is a second order differential equation

$$\tau_x \ddot{x}(t) + \dot{x}(t) = K_x e_x(t) - K_x f_C \operatorname{sgn}\left[\dot{x}(t)\right] \tag{1}$$

where $\tau_x = 1.48 \cdot 10^{-2}$ s, $K_x = 0.840$ (mm/s)/V, and $f_C = 0.980$ V. The model is determined by subjecting the linear axis to a series of step input voltages. The linear axis response to these inputs closely resembles the second order model given in equation (1) whose parameters are determined using recursive least squares estimation. The linear axis model is used in the Simulation and Emulation modes and is shown in Figure 4 in Simulink syntax.

The Phase 1 GUI is shown in Figure 8 and includes four options: Mode, Controller, Build, and Help. The Mode option is a drop down list containing the three modes: Simulation, Emulation, and Implementation, which are described above. The modes appear in the order in which they should be used. The Controller option is the second drop down list containing three controller options: Proportional, Proportional plus Integral plus Derivative (PID), and Student Controller. The student can use the first two controllers, Proportional and PID, to operate the RDS without

having to create their own controller. The controller gains for the Proportional and PID controllers have been tuned using trial and error methods. These controllers provide a benchmark with which the students can compare the performance of their own controller. Each controller is a subsystem in a separate Simulink file. The Student controller option is a controller the student creates in a separate Simulink subsystem file according to the previously specified format. This separate subsystem is automatically inserted in the Simulink mode model using the model–altering program. The student can use this option to insert their controller. When the student makes a selection from either list, the selection is stored as a variable to be used by the model–altering code.

The Build option is a push button that runs the model–altering program to create the desired Simulink model. The model–altering program first gathers the block parameters from the selected controller subsystem and appends them to the end of the block parameters of the Simulink mode model. The block parameters describe the contents and connections of the blocks contained in the controller subsystem. The model–altering program then reads the block parameters of the desired controller subsystem and inserts them into the Simulink mode model Controller subsystem. When this is complete, a new Simulink mode model file is saved and then opened in Matlab 7.1.

The Help option is a push button that opens a document in Microsoft Word. This file contains directions for each step the student must take to select the mode and controller, and build the new Simulink mode model. The student may view the help document at anytime while using the GUI. The help file provides step–by–step directions to make mode and controller selections and to build the new Simulink mode model. The student can return to the GUI to create another Simulink mode model any number of times; however, the previous Simulink mode model must be closed before creating a new Simulink mode model since the new Simulink mode model overwrites the previous Simulink mode model. The student must operate the Simulink mode model outside of the GUI using the build and run buttons provided in Simulink. The build button provided in Simulink converts the Simulink mode model into C code and then loads that C code onto the target computer. The run button provided in Simulink initiates the Simulink mode model on the target computer. After running the Simulink simulation mode model the student types the command “plotsimulation” in the Matlab 7.1 command window to run a plotting program that reads the simulation results on the host computer and generates the controller performance plots. The command “plotresults” is typed in the Matlab 7.1 command window after running the Simulink emulation or implementation mode models. This command runs a plotting program that reads the results on the target computer and generates the controller performance plots. Different plotting programs are required since the format of the data saved after a run in Simulation mode is different that the format of the data saved after a run in Emulation or Implementation modes.

Assessment Model

The model depicted in Figure 9 serves as a guide for the evaluation of the Linear Axis RDS. The model is based on the Technology Acceptance Model (TAM) and examines the connection between the user, the possibility of the Linear Axis RDS as a learning tool, and the learning

outcomes. The model is an extension of assessment models conducted by the evaluation team in previous projects [8].

Technology Acceptance Model (TAM) is an extension to Fishbein and Ajzen's [9] Theory of Reasoned Action that explains the relationship between attitude towards the technology and intention to use it. Theory of Reasoned action suggests that if a person is to perform a certain action, it would depend upon his/her attitude towards that action, and how others would see it, as to whether or not he/she performs it [9,10].

Technology Acceptance Model (TAM) suggests that perceived usefulness of the technology, as well as users' perceived ease of use of the technology, determine the users' attitude towards the technology, which in turn influences the users' intention to use the technology. Technology Acceptance Model has been widely applied to study user acceptance of new technologies and has been found to be effective in predicting users' intention to adopt or use a particular technology. For example, Davis [10] showed that TAM can explain the use of information technology; while Lederer and his colleagues successfully applied it with the World Wide Web [11].

Learning styles are examined using the abbreviated version of the Index of Learning Styles for engineering students suggested by Felder and Spurlin [12]. In the questionnaire, they established various dimensions that define how students receive and process information and the manner in which they learn.

Learning outcome are comprised of questions about perceived knowledge gained by the participant with respect to various methods of studying/teaching, such as text, lab, and computer software [8]. Learning outcomes also include motivation to learn, real–world applicability and Metacognition, which represents the learners knowledge with regard to what they know and do not know about.

Methodology

Three participants were recruited for this study. They were taking the course "Modeling and Control of Manufacturing Process," which is an advanced graduate course, and had never used the Linear Axis RDS. In order to measure the effectiveness of the Linear Axis RDS and learning outcomes of using the Linear Axis RDS, it was required that the participants had prior knowledge, at least theoretical, about control design, were currently working in or studying controls, and would be representative of those who would utilize the learning system. Both qualitative and quantitative data were collected consistent with the assessment model. A principal goal of the assessment was to make constructive suggestions for developing the Phase II Linear Axis RDS based on the results of this study.

Eye Tracking

Eye–tracking is a method of recording the users' eye–movements during their interaction on the screen when performing the tasks. Gaze Plot was the main method used for analyzing the eye tracking data in this study. The Gaze Plot displays a static view of the gaze data for each image

of the stimuli and is a useful tool when visualizing scan paths. Each fixation is illustrated with a blue dot where the radius represents the length of the fixation (i.e., the larger the dot, the longer the user fixated on that particular point).

Experimental Procedure

The experiment was conducted one participant at a time. After completing a consent form, the participant filled out a pre–experiment questionnaire, which included questions on learning style, adapted from [13], and general and career interest in science and enjoyment in science, adapted from Fraser [14]. Next, the participant was given a very brief introduction of the Linear Axis RDS and instructed in how to "think aloud" [15] as they performed the tasks. The participant was then given a list of tasks to perform. The list consisted of three tasks that covered most of the functionality of the Phase I Linear Axis RDS:

Task 1: Select and simulate the controller. Generate model.
Task 2: Select and emulate the controller. Generate model.
Task 3: Go back and implement a pre–existing controller.

The participants were occasionally prompted during the process with questions like "what are you thinking right now?" and "what is going on in your mind?" After a participant finished working on the first two tasks, eye–tracking equipment was used to track the user's gaze on the third task.

When the participants completed all three tasks, they were given the post–questionnaire. In the post–experiment questionnaire, questions about interest and enjoyment in science were repeated to see if working with Linear Axis RDS for control design/implementation had an impact on the user's feelings about science. The post questionnaire also included questions from the TAM, including perceived usefulness and perceived ease of use, adapted from [10], and attitude towards the technology and intention to use the system in the future, adapted from [16,17]. Also, learning outcome was measured with questions adapted from [8]. After completing the post–questionnaire, each participant was interviewed with open–ended questions about his/her experience using the Linear Axis RDS, application of the Linear Axis RDS, strengths and weaknesses of the Linear Axis RDS, and suggestions for improvement.

Results

The Linear Axis RDS was used by three students in a manufacturing automation course. As an example, one students' implementation is described. Following this, the results of all three students are presented.

A student learning the fundamentals of controller design and implementation used the Phase 1 Linear Axis RDS to implement their controller and their results are now presented. The student designed a modified PID controller in Simulink, shown in Figure 10, based on the linear axis model in equation (1). The equation used to calculate the command control voltage is

$$V_c(z) = \left[K_I \frac{z}{z-1} \right] E(z) - \left[K_P + K_D \frac{z-1}{z} \right] X(z) \quad (2)$$

where $V_c(z)$ is the command voltage, $E(z) = X_r(z) - X(z)$ is the position error, $X_r(z)$ is the reference position, $X(z)$ is the measured position, K_I is the integral gain, K_P is the proportional gain, and K_D is the derivative gain. The Phase 1 Linear Axis RDS was used to simulate, emulate, and implement the student's controller using a sinusoidal reference with magnitude 20 mm and frequency of 0.125 Hz. The controller gains were tuned based on the performance results for each test. Using the Phase 1 Linear Axis RDS the student controller was simulated and tuned using trial and error methods until the following controller gains are reached: $K_P = 985.9$, $K_I = 44{,}810$, and $K_D = 6.032$. The results for these controller gains using simulation are shown in Figure 11. The student then emulated the controller and found the results were similar to the simulation results and that the controller could be run in real time using a sample period of 1 ms. The emulation results are shown in Figure 12. The student then implemented his controller on the physical system. The limitations of the physical model are evident in the implementation results; therefore, the gains were modified to account for effects that are not represented in the model. The final controller gains were $K_P = 943.9$, $K_I = 41{,}950$, and $K_D = 5.878$, corresponding to the results shown in Figure 13. This shows that the primary mode used for controller tuning is the simulation mode.

Only one out of three participants completed all the tasks. One of the participants did not complete any of the three tasks. Table 1 summarizes the task performance of each participant. Since the study only included three participants, only descriptive statistics were considered. Every question in the post–questionnaire and science–related questions in the pre–questionnaire had a 7–point Likert scale response from "Strongly Disagree" to "Strongly Agree." Once the data was collected, the average for each item was calculated. Results of the learning style questionnaire verified that the engineering students are strong visual learners. Two of the participants are shown to have preference on visual learning. Participant 3's response showed he was both a visual and auditory learner.

Figure 14 summarizes the results on the technology acceptance model (TAM). The results of the TAM model questionnaire are presented, in terms of each of its components, in the following list:

1. Perceived Usefulness: Unlike participants 1 and 3, participant 2 did not have any previous practical experience with control design/implementation, and very little theoretical exposure in courses taken in the previous semester (Spring 2008). It was seen that both experienced participants perceived the Linear Axis RDS would be useful with the average response tending towards positive.
2. Perceived Ease of Use: Results showed the experienced participants felt the Linear Axis RDS was quite easy to use. The inexperienced participant disagreed with the ease of use of the Linear Axis RDS. The results can also be validated from the task participant performance where the participant who completed all three tasks had the highest response to the perceived ease of use and the participant who could not complete any of the three tasks had the lowest response.

3. Attitude Towards Linear Axis RDS: The findings show that regardless of experience and task performance all participants had a positive attitude towards the Linear Axis RDS.
4. Intentions to Use Linear Axis RDS: The results indicate an overall positive intention for future use of the Linear Axis RDS.

Using the questions about the perceived information learned about control design/implementation, motivation to learn, and real–world applicability of the Linear Axis RDS, lectures and class text were measured and compared. Results indicated the class lecture was perceived to be the most effective overall (Mean = 5.67), while the class text (Mean = 4.33) and the Linear Axis RDS (Mean = 4.0) were largely equivalent. Students found the Linear Axis RDS to be particularly effective with respect to "real world applications" (see Figure 15).

In addition, students were asked to rate their knowledge about control design before and after interacting with the Linear RDS. As indicated in Figure 16, two participants knew quite a bit before, indicating no change, while one participant reported increased knowledge after the experience with the Linear Axis RDS.

Results from pre versus post questions about career, general interest in science, and enjoyment with respect to science were compared. Findings reveal no change in these constructs (see Figure 17).

Data about the users' facial expressions and voice and screen recording were captured and analyzed. Several key themes emerged from observation and analysis of the data recovered from the Think Aloud protocol. Some of the major themes are as follows:

- Since the participants were using the Linear Axis RDS for the first time, it would presumably be necessary for them to use the HELP file. However, two of the three participants did not notice the HELP button on the Linear Axis RDS interface until the experimenter prompted them.
- All participants had confusion about the function of Student Controller and which controller to choose in the first task.
- All participants were reluctant to use the HELP file. Even when they started using the HELP file, all the participants found it extremely difficult to find the right information from the HELP file.

An eye–tracker was used to record the participants' eye–movements and activity on the screen during Task 3. After the experiment, Gaze Plot data was exported in the form of image files. The results of the Gaze Plots for each participant are discussed below.

Participant 1: The Gaze Plot for participant 1 is shown in Figure 18. The window in the center is the Linear Axis RDS interface. We found that participant 1 focused on the Linear Axis RDS as well as at the window in the background.
Participant 2: The Gaze Plot for participant 2 is shown in Figure 19. Participant 2 did not complete any of the tasks. Analysis of the Gaze Plot indicates participant 2 was not able to find the relevant information on the Linear Axis RDS interface to complete the tasks. For example, although there is just the desktop in the background and no other windows, the

participant's eyes were all over the screen, showing signs of confusion and nervousness in using the Linear Axis RDS interface.

Participant 3: The Gaze Plot for participant 3 is shown in Figure 20. Participant 3 was successful in completing all the tasks. The Gaze Plot indicates the participant knew what to do and where to go as most of the dots were concentrated on the Linear Axis RDS interface.

After the participants completed the tasks, interviews were conducted to understand the participants' previous experience with the subject area, experience using the Linear Axis RDS, application of the Linear Axis RDS, their learning experience, strengths and weaknesses of the Linear Axis RDS, and suggestions for improvement. Three themes emerged, based on the participants' responses:

1. Past Experience
   - Participants 1 and 3 were very experienced with control design/implementation, while participant 2 did not have previous practical experience with control design/implementation, and very little theoretical exposure in courses taken in the previous semester (Spring 2008).
2. Strengths of the Linear Axis RDS
   - Participant 1 felt the Linear Axis RDS was very convenient as it kept users from rewriting a lot of code every time. He thought it would be useful in the academic environment.
   - Participant 2 said the HELP file was self–explanatory and felt the Linear Axis RDS would be useful as a learning tool.
   - Participant 3 felt the Linear Axis RDS would be a good tool for first time exposure to control design/implementation, stating, "It would be nice if I first got exposed to it." He also said the Linear Axis RDS is a good prototyping tool in the "real–world" and it is good that there is a template which one knows that works; all he/she has to do is make slight modifications to the template as opposed to starting from scratch. He thought the Linear Axis RDS would be useful in industry and introductory controls courses. Overall, he enjoyed the experience with the Linear Axis RDS and said "Nice development system for very complex programs." Like participant 2, participant 3 also felt the HELP file was self–explanatory.
3. Weaknesses of the Linear Axis RDS
   - Participant 1 did not like Simulink as a programming language. He said, "RDS limits the controllers that you are able to make with it because one might want to differentiate past values of the control signal."
   - Participant 2 felt that working on the Linear Axis RDS was very confusing. Also, the HELP file was not at all useful, because it was not arranged in order of the tasks.
   - Participant 3 stated that the Linear Axis RDS helps to see the overall picture but one does not learn as much as he/she is not able see the full aspects of how it is working. Also, he felt that more information about the Linear Axis RDS could be incorporated in the HELP file.
   - All participants indicated the computer that they were working on was very slow.

Phase 2 Linear Axis RDS

The goals of the Phase 2 Linear Axis RDS are to enable the student to build and operate the Simulink model using the GUI, to visually analyze the controller performance using the GUI, to expand the selectable options for the contents of the Reference Generator and Controller subsystems, and to include visible tips and an interlinking help file. The method for building the Simulink model used in the Phase 1 GUI is also used in the Phase 2 GUI. This includes the three Simulink mode models and the model–altering code. Callback code linked to selectable features in the GUI is used to allow the student to operate the Simulink model while using the GUI. An animation of the linear axis is used to allow the student to view the motion of the linear axis during the operation of any of the three modes. A modified PID controller and a general tracking controller are added to the list of selectable controllers. A box containing tips to help the student operate the GUI is added along with a help file containing links for specific actions.

The Phase 2 GUI consists of two displays: the model builder and the model operator. The model builder is shown in Figure 21 and contains 7 options. A brief introduction to the model builder is given above the **help** push button. The **help** push button is currently incomplete but will open a help display, which contains selectable directions and detailed instructions for configuring a Simulink mode model. The mode selection option is a drop down box containing the three operation modes: simulation, emulation, and implementation. The controller selectable option is a drop down box containing Proportional, PID, Student, Modified PID, and General Tracking controllers. When the **Student Control** option is selected a display appears with the current Matlab directory contents, i.e., the file location which Matlab currently uses to open or save programs. The student can select the controller they previously created. The reference selection option is a drop down box containing three reference position generators: square, triangular, and sinusoidal. When the mode, controller, and reference have been selected, the student selects the **build** push button. This will activate the model–altering program that will read the desired content of the selected controller and reference and insert it into the Simulink mode model. When the new Simulink mode model is complete the **build** push button will display "Complete." The student will have the option to open the new Simulink mode model by selecting the **view model** push button or to continue to the model operator by selecting the **continue** push button.

The model operator is shown in Figure 22 and contains ten options. A brief introduction to the model operator is given above the **help** push button, which has the same function as the **help** push button in the model builder. The new Simulink mode model created using the model builder is opened by pushing the **view model** push button. The new Simulink mode model is operated by pushing the **run** push button, which is labeled according to the simulation, emulation, or implementation mode. For example, simulation is displayed on the **run** push button in Figure 22. The length of time in seconds that the system is in operation is entered by the student in the **run time** edit textbox. To stop the operation of the Simulink model, the **stop** push button is pushed at anytime during the operation. An animation of the linear axis moves according to the measured position while the system is in operation in any mode. This is done by first creating a CAD model of the linear axis and then importing it into the GUI. The model image is displayed and updated using animation callback codes. The physical linear axis position can be adjusted by pushing the **jog** push button. This opens the jogger display, shown in Figure 23, which allows the student to move the linear axis in the positive or negative direction in three incremental

distances. When the linear axis is located in the desired position the **return to model operator** push button is selected and the model operator display is shown. When the system operation is complete the student is given two options to analyze the controller performance results. The controller results are given in a display containing three graphs when the **plot** push button is pushed (see Figures 11–13). The reference and measured positions are shown in the first graph, the command voltage is shown in the second graph, and the measured position error is shown in the third graph. The controller results are saved in a text file by pushing the **save** push button. The student must type the name of the data file in an edit text box which appears when the **save** push button is selected. This text file is saved in the current Matlab directory. The student can navigate back to the model builder using the **back to model builder** push button. This closes the model operator and opens the model builder with the last Simulink mode model saved as the current Simulink mode model.

The Phase 2 Linear Axis RDS has not yet been tested. Future versions of the Linear Axis RDS will include the following: linear axis model selection to allow the student to insert a linear axis model, real–time plotting of the controller results during operation, and remote simulation of the Simulink model via the internet.

Summary and Conclusions

The first phase of a Linear Axis Rapid Development System (RDS) was developed and utilized by students learning controller design and implementation concepts. The Linear Axis RDS has two interface components: the GUI interface with the student and the computer–system interface with the linear axis. Usability results were obtained using the Think Aloud protocol and the eye tracking tool. Based on the results, the second phase of the Linear Axis RDS is currently being developed.

The following suggestions for a Phase 2 Linear Axis RDS were made based on the results that were gathered from both qualitative and quantitative data:

1. Since all the participants had confusion about the function of each controller, it would useful if a tip/status box was incorporated into the Linear Axis RDS. When the user selects a particular controller from the drop down menu, the tip/status box would brief the user with the function of the controller.
2. Since it did not register in the minds of the users about the importance of the HELP file and the fact that all the participants were reluctant to use the HELP file, it would be highly effective to prompt the users for using the HELP file. For example:
   [To insert your controller, first update your filename to Studentcontroller and go to Student controller]
   For more directions, click on the button below. [HELP button below]
3. Reorganize the HELP file in order of the tasks that the users would perform on the Linear Axis RDS. Highlight important points, headings and appropriate use of diagrams of the Linear Axis RDS.
4. Design Implication I: Make the HELP button bigger and bolder.

5. Design Implication II: Have vertical arrangement of the drop down menus (Mode, Controller) in order to accommodate the status/ tip box.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant Number 0736731.

References

[1] Malmborg, J. and Eker, J., 1997, "Hybrid Control of a Double Tank System," *IEEE Conference on Control Applications*, Hartford, Connecticut, October 5–7, pp. 137–138.
[2] Mori, S., Nishinara, H., and Furuta, K., 1976, "Control of Unstable Mechanical System, Control of Pendulum," *International Journal of Control*, Vol. 23, pp. 673–692.
[3] Yamakita, M., Iwashiro, M., Sugahara, Y., and Furuta, K., 1995, "Robust Swing Up Control of a Double Pendulum," *American Control Conference*, Seattle, Washington, June 21–23, pp. 290–295.
[4] Sasaki, N., Ohyama, Y., and Ikebe, J., 1997, "Design Exercises for Robust Controller Using a Double Inverted Pendulum," *4th IFAC Symposium on Control Education*, Istanbul, Turkey, July 14–16, pp. 301–305.
[5] Meier, H., Farwig, Z., and Unbehauen, H., 1990, "Discrete Computer Control of a Triple Inverted Pendulum," *Optimal Control Applications and Methods*, Vol. 11, pp. 157–171.
[6] Whelan, J. and Ringwood, J.V., 1994, "A Demonstration Rig For Control Systems Based On the Ball and Beam with Vision Feedback," *3rd IFAC Symposium on Control Education*, Tokyo, Japan, August 1–2, pp. 9–15.
[7] Sridharan, S. and Sridharan, G., 2002, "Ball on Beam on Roller: a New Control Laboratory Device," *Industrial Electronics*, Vol. 4, pp. 1318–1321.
[8] Hall, R.H., Philpot, T., and Hubing, N., 2006, "Comprehensive Assessment of a Software Development Project for Engineering Instruction," *Journal of Learning, Technology, and Assessment*, Vol. 15, No. 5, pp. 4–42.
[9] Fishbein, M. and Ajzen, I., 1975, *Belief, Attitude Intention, and Behavior: An Introduction to Theory and Research*, Reading, Massachusetts: Addison–Wesley.
[10] Davis, F., 1989, "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology," Vol. 13, No. 3, *MIS Quarterly*, pp. 318–340.
[11] Lederer, A.L., Maupin, D.J., Sena, M.P., and Zhuang, Y, 2000, "The Technology Acceptance Model and the World Wide Web," *Decision Support Systems*, Vol. 29, No. 3, pp. 269–282.
[12] Felder, R.M. and Spurlin, J., 2005, "Applications, Reliability, and Validity of the Index of Learning Styles," *International Journal of Engineering Education*, Vol. 21, No. 1, pp. 103–112.
[13] Felder, R.M. and Soloman, B.A., 2001, Index of Learning Styles Questionnaire, North Carolina State University, Online at: http://www2.ncsu.edu/unity/lockers/users/f/felder/public/ILSdir/ILS–a.htm.
[14] Fraser, B.J., 1981, "Test of Science Related Skills," Australian Council for Educational Research, The Australian Council for Educational Research Limited: Hawthorn, Victoria.
[15] van Someren, M.W., Barnard, Y.F., and Sandberg, J.A.C., 1994, *The Think Aloud Method: A Practical Guide to Modeling Cognitive Processes*, San Diego, California: Academic Press.
[16] Shirley, T. and Todd, P.A., 1995, "Understanding Information Technology Usage: A Test of Competing Models," *Information Systems Research*, Vol. 6, No. 2, pp. 144–176.
[17] Gefen, D., Karahanna, E., and Straub, D.W., 2003, "Trust and TAM in Online Shopping: An Integrated Model, *MIS Quarterly*, Vol. 27, No. 1, pp. 51–90.

Table 1: Task Performance.

Participant	Task 1	Task 2	Task 3
1	Completed	Completed	Not completed
2	Not completed	Not completed	Not completed
3	Completed	Completed	Completed

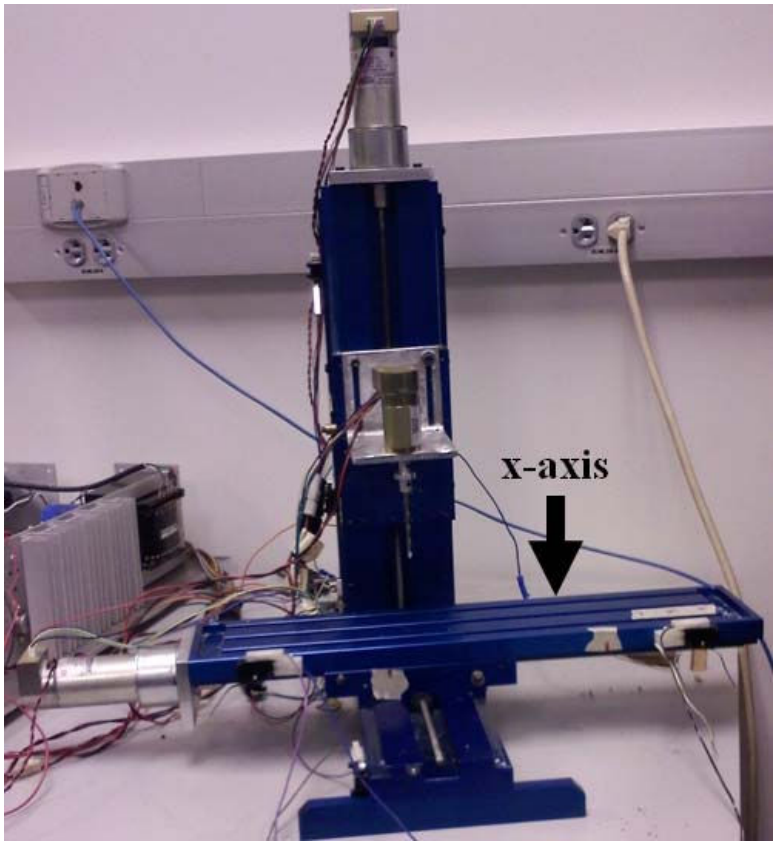

Figure 1: Mini–CNC System Containing Linear Axis Used in RDS.

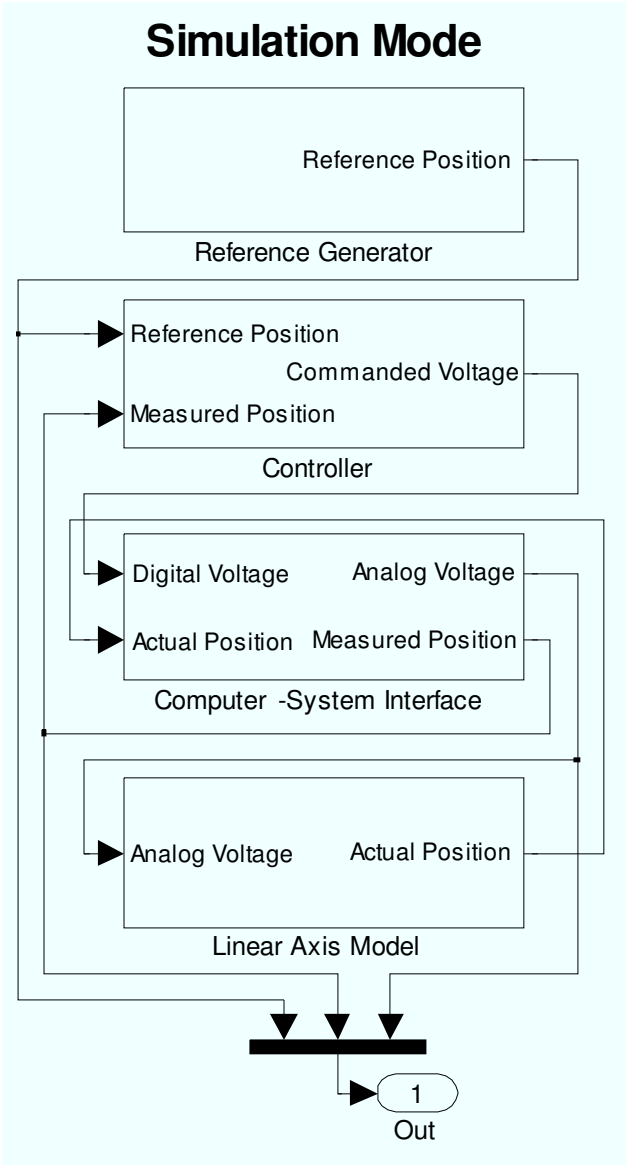

Figure 2: Simulink Model for Simulation Mode.

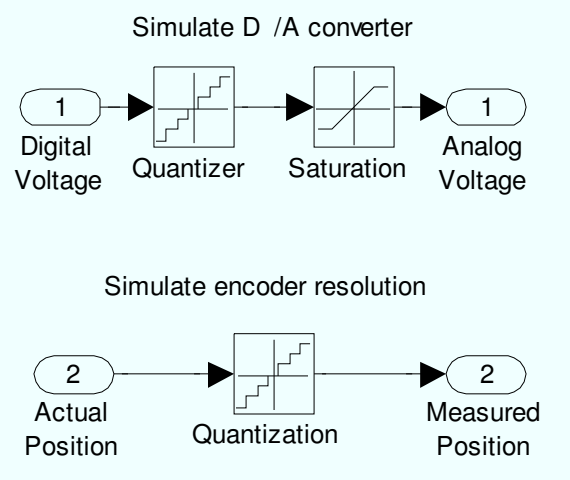

Figure 3: Contents of Computer–System Interface Subsystem for Simulation Mode.

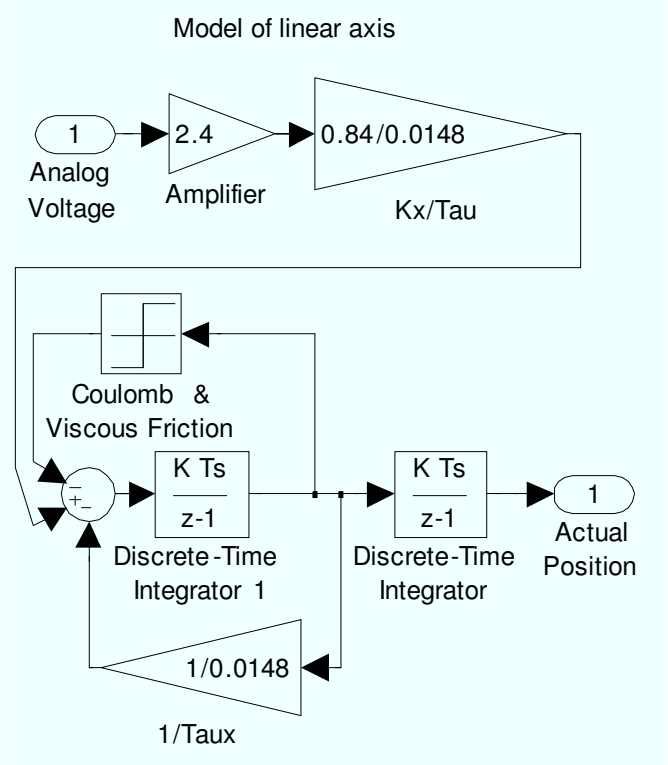

Figure 4: Linear Axis Model Contained in Linear Axis Model Subsystem.

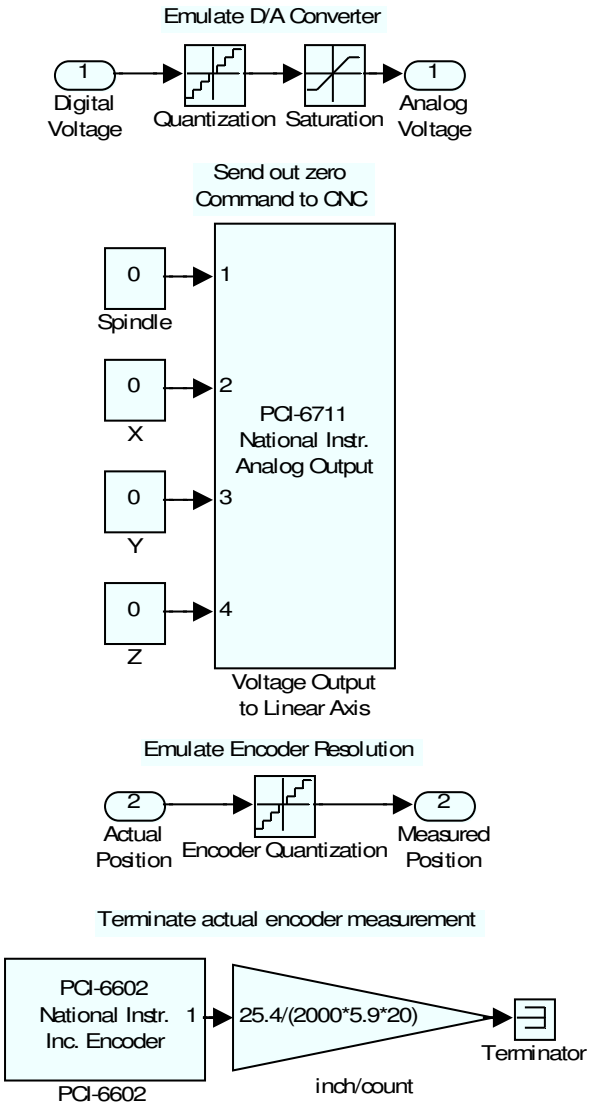

Figure 5: Contents of Computer–System Interface Subsystem for Emulation Mode.

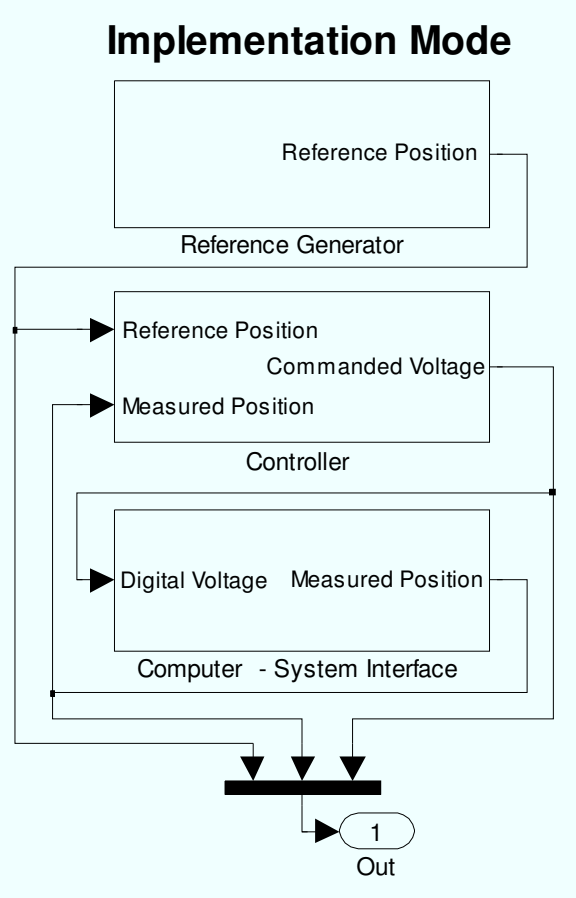

Figure 6: Simulink Model for Implementation Mode.

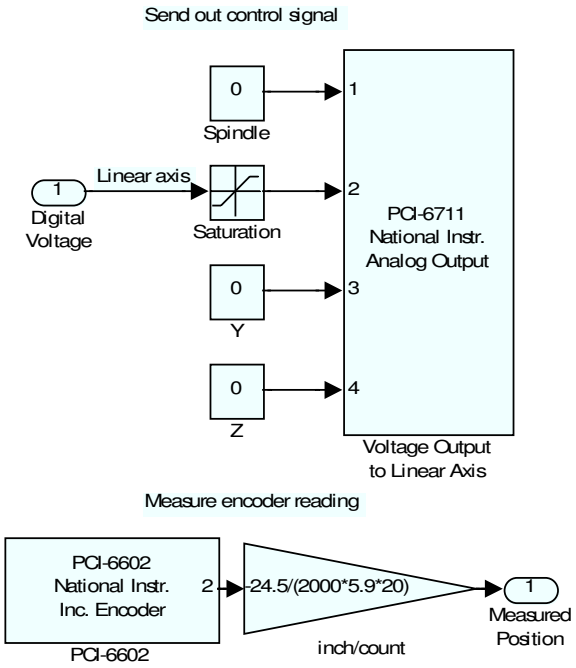

Figure 7: Contents of Computer–System Interface for Implementation Mode.

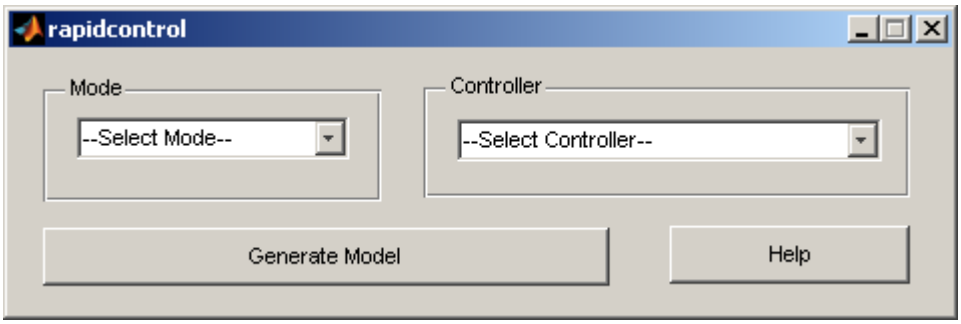

Figure 8: Phase 1 Linear Axis RDS GUI.

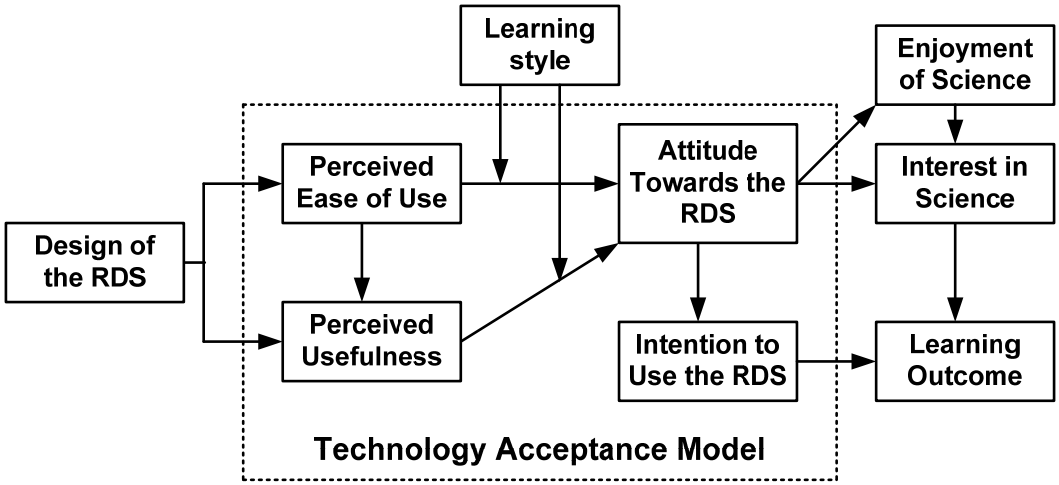

Figure 9: Assessment Model.

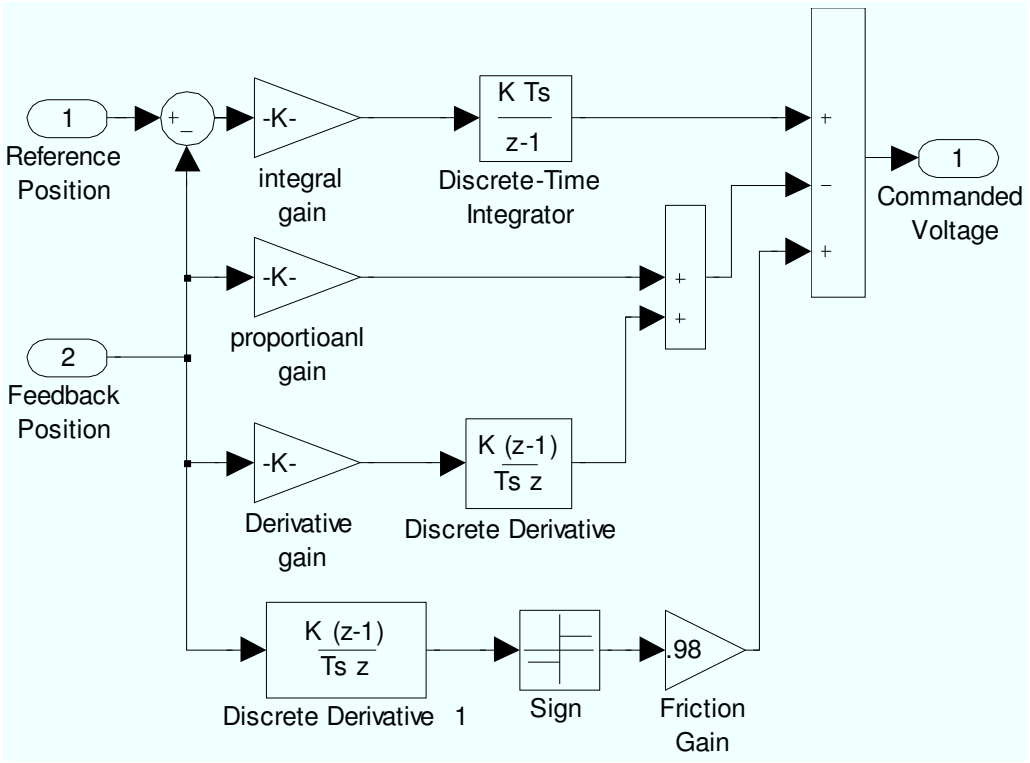

Figure 10: Modified PID Controller Subsystem.

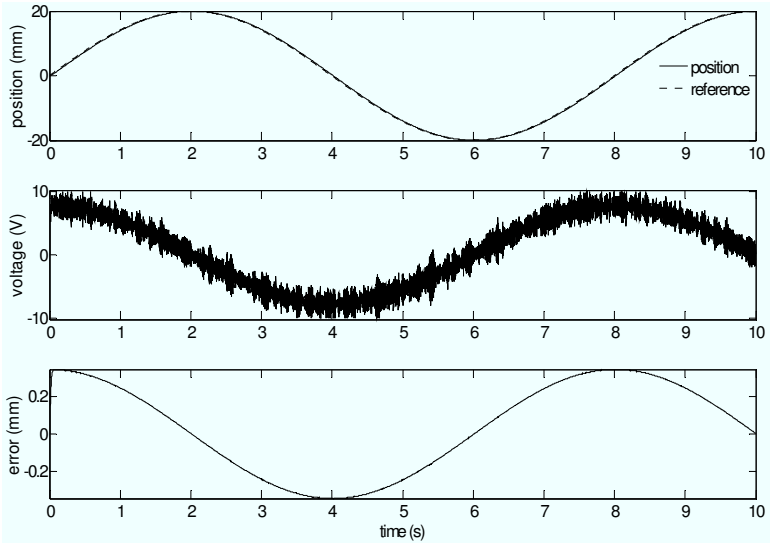

Figure 11: Results for Modified PID using Simulation Mode.

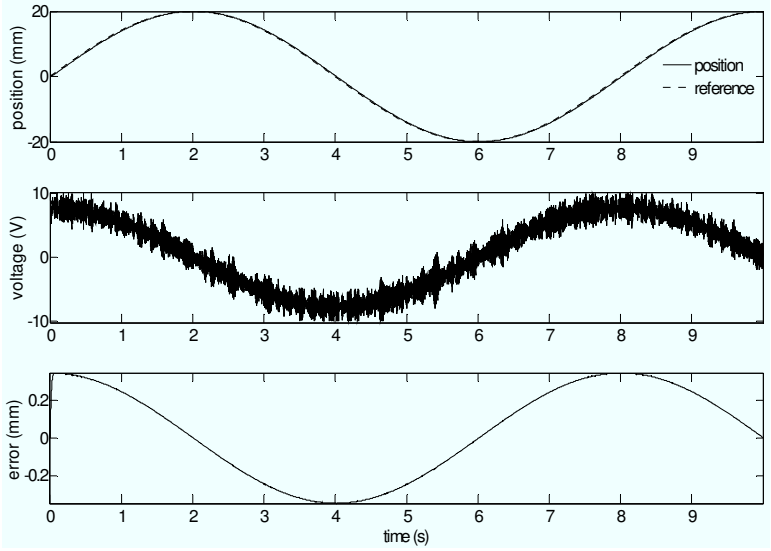

Figure 12: Results for Modified PID Controller using Emulation Mode.

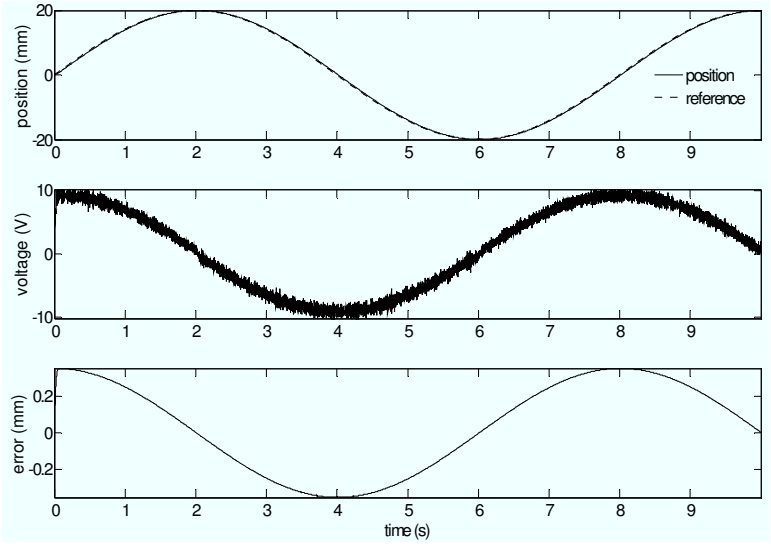

Figure 13: Results for Modified PID Controller using Implementation Mode.

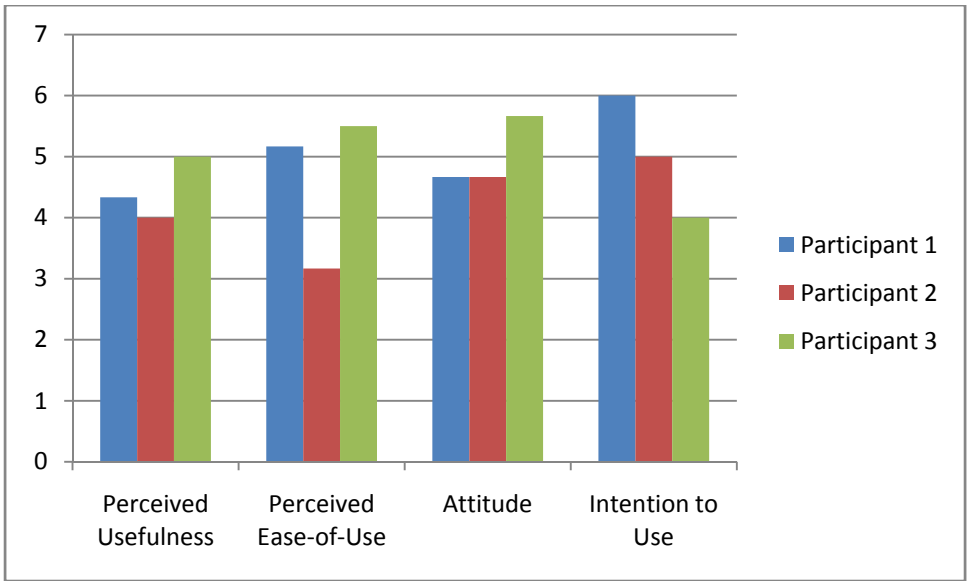

Figure 14: Technology Acceptance Model.

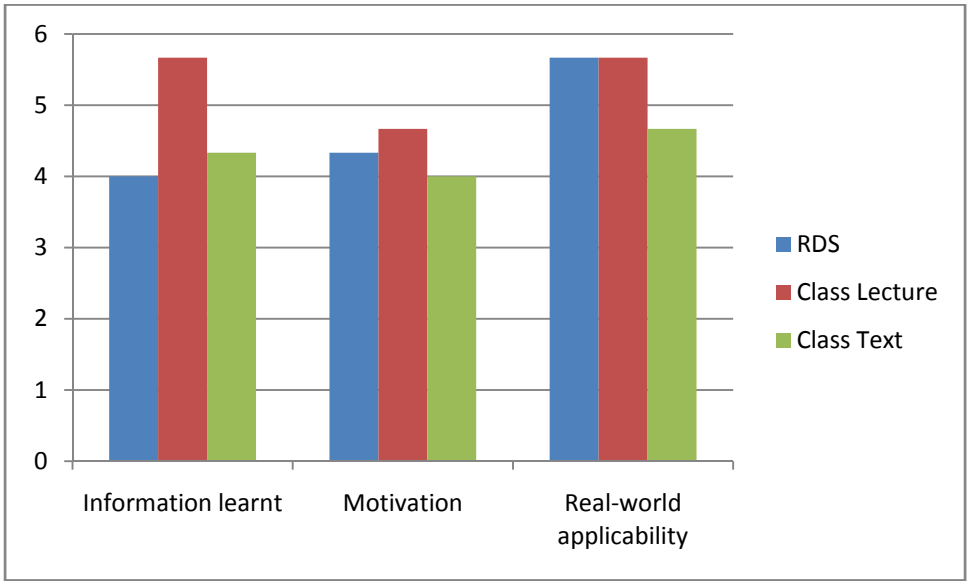

Figure 15: Learning Outcomes I.

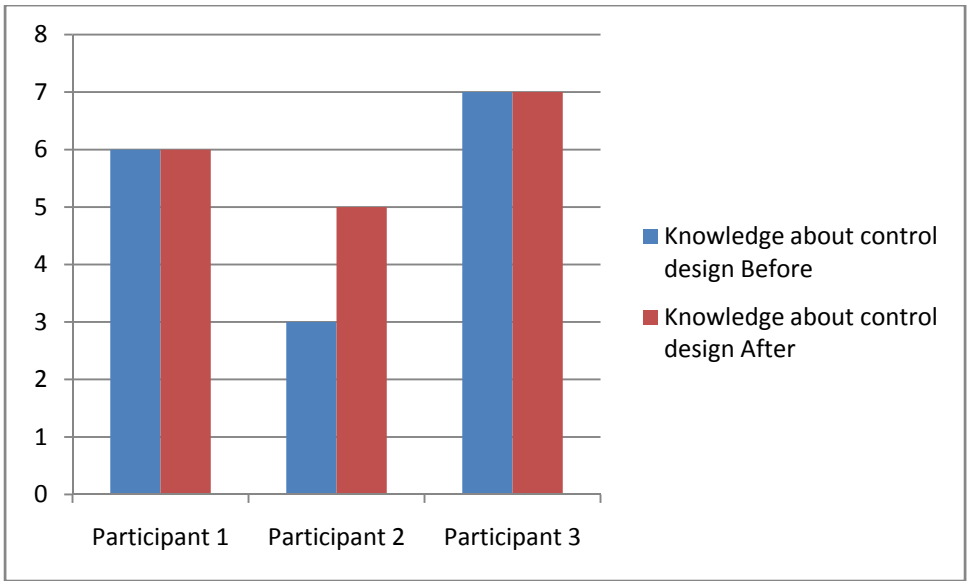

Figure 16: Learning Outcomes II.

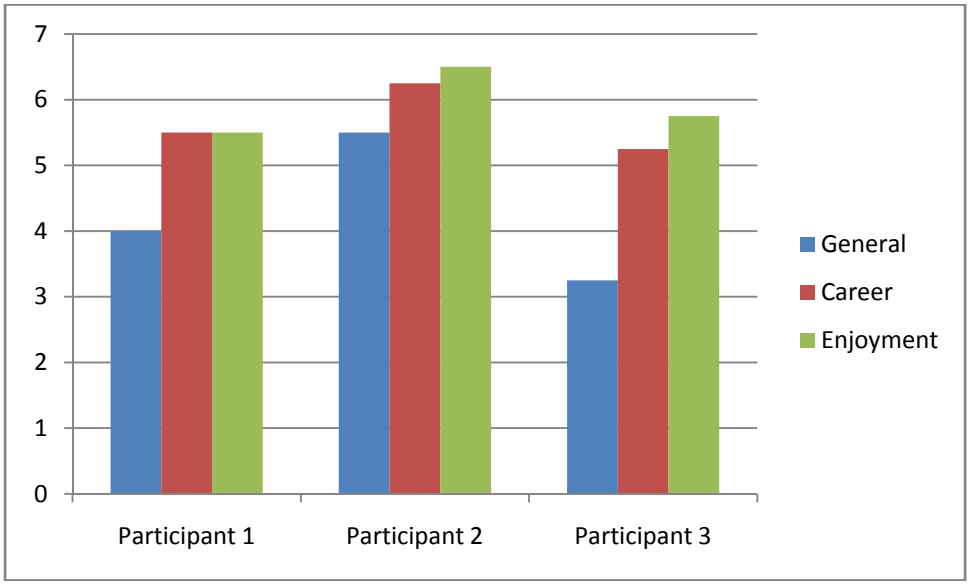

Figure 17: Interest and Enjoyment in Science.

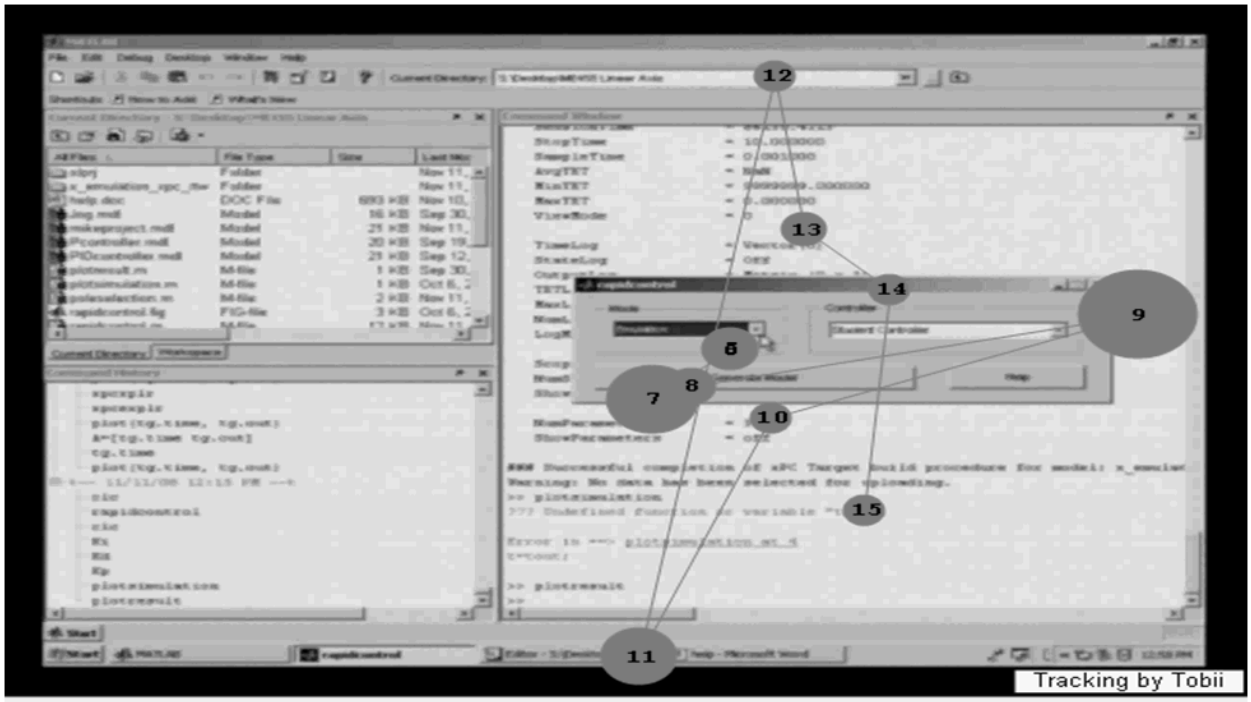

Figure 18: Gaze Plot – Participant 1.

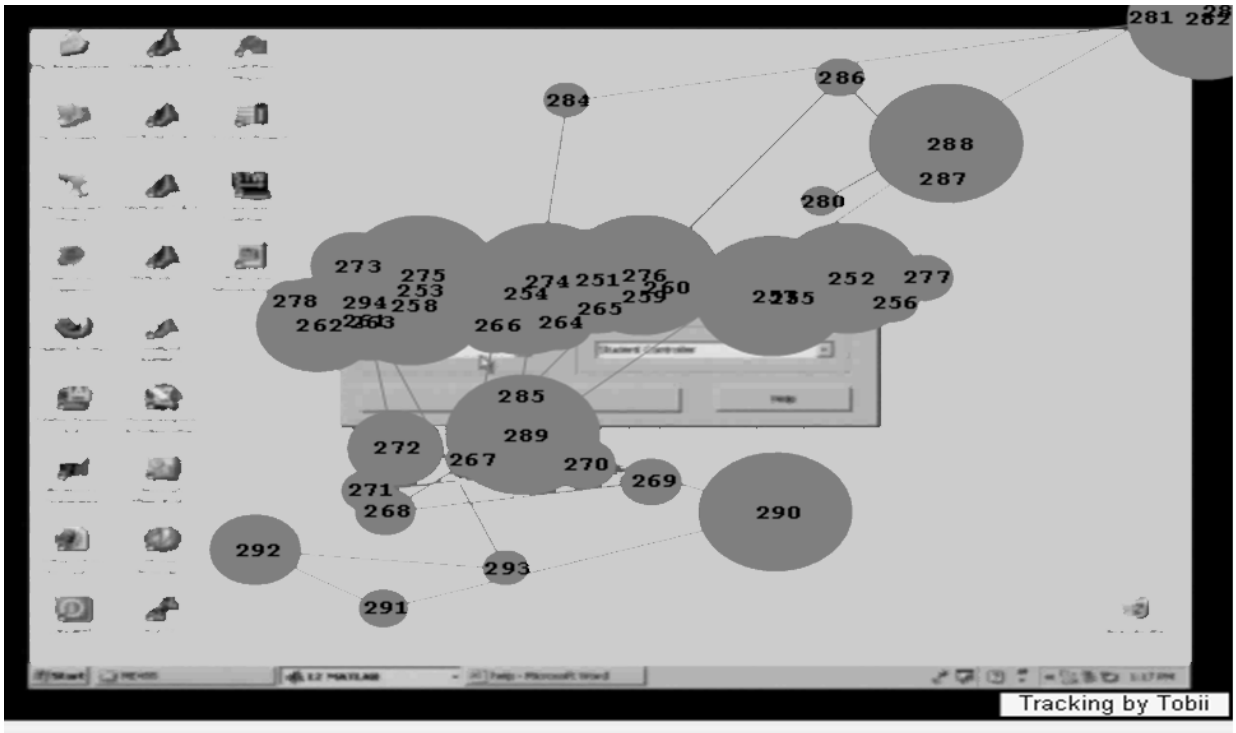

Figure 19: Gaze Plot – Participant 2.

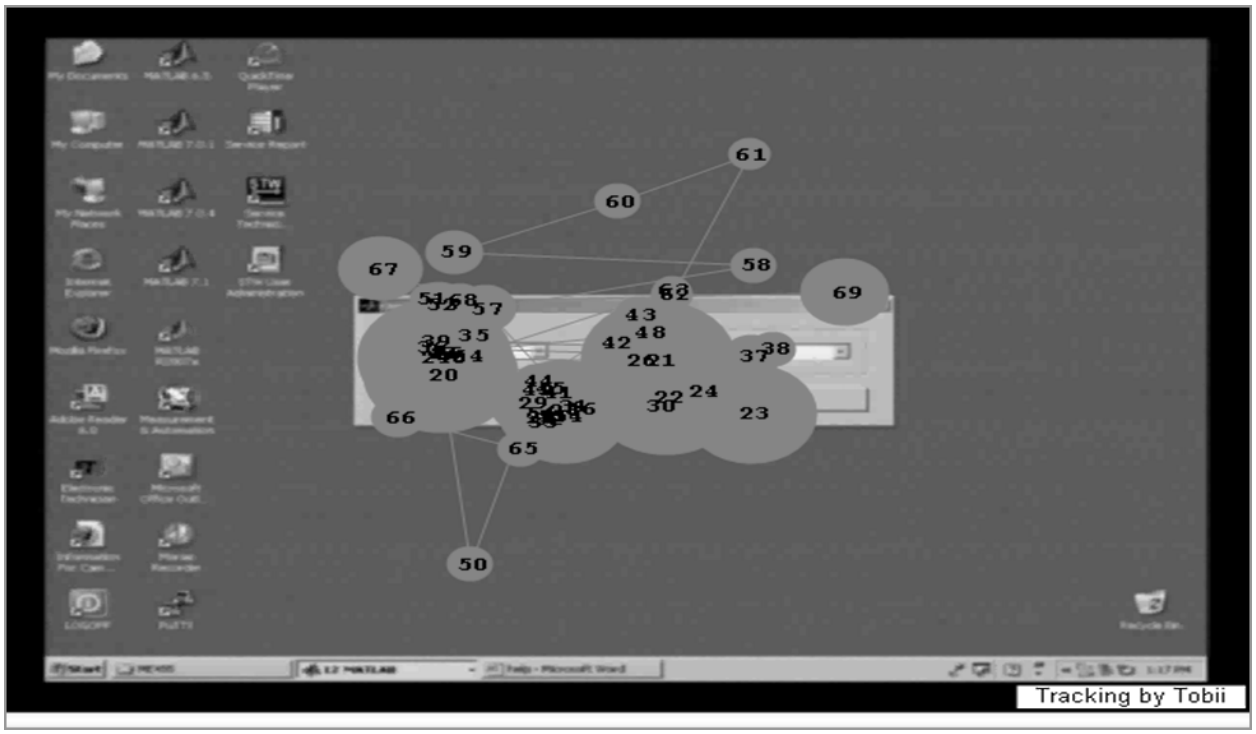

Figure 20: Gaze Plot – Participant 3.

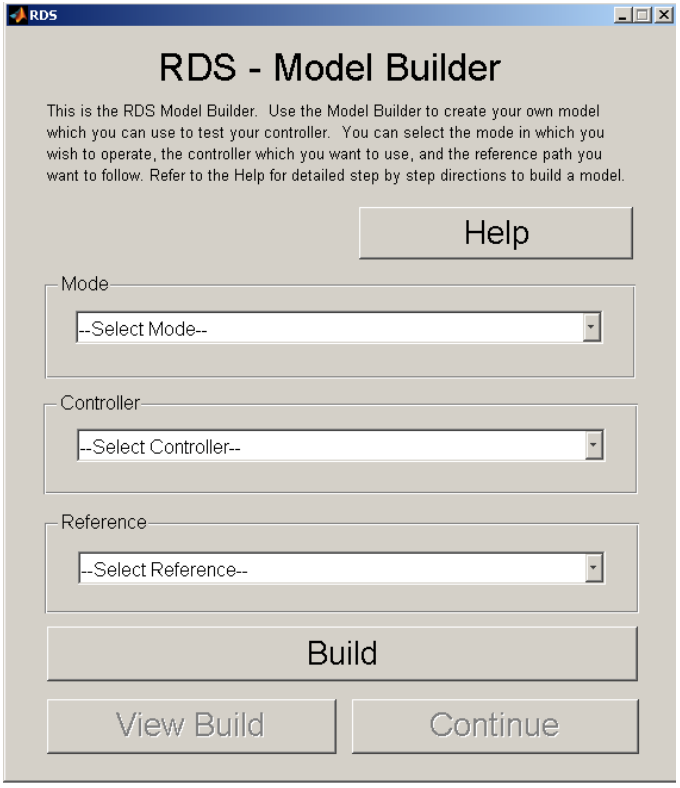

Figure 21: Linear Axis RDS Phase 2 GUI Model Builder.

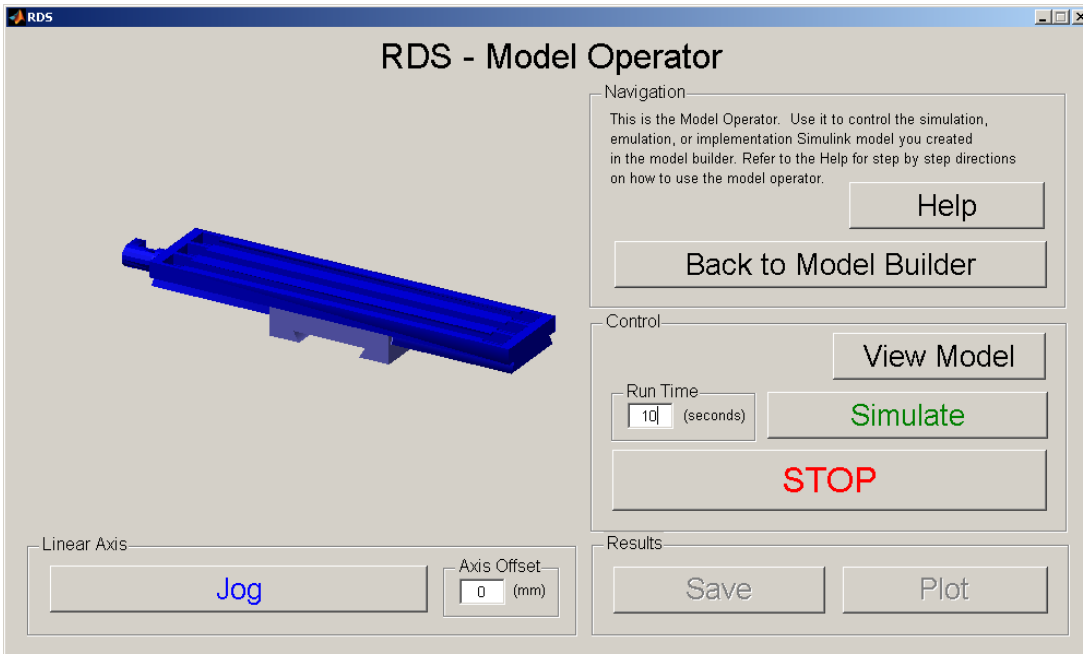

Figure 22: Linear Axis RDS Phase 2 GUI Model Operator in Simulation Mode.

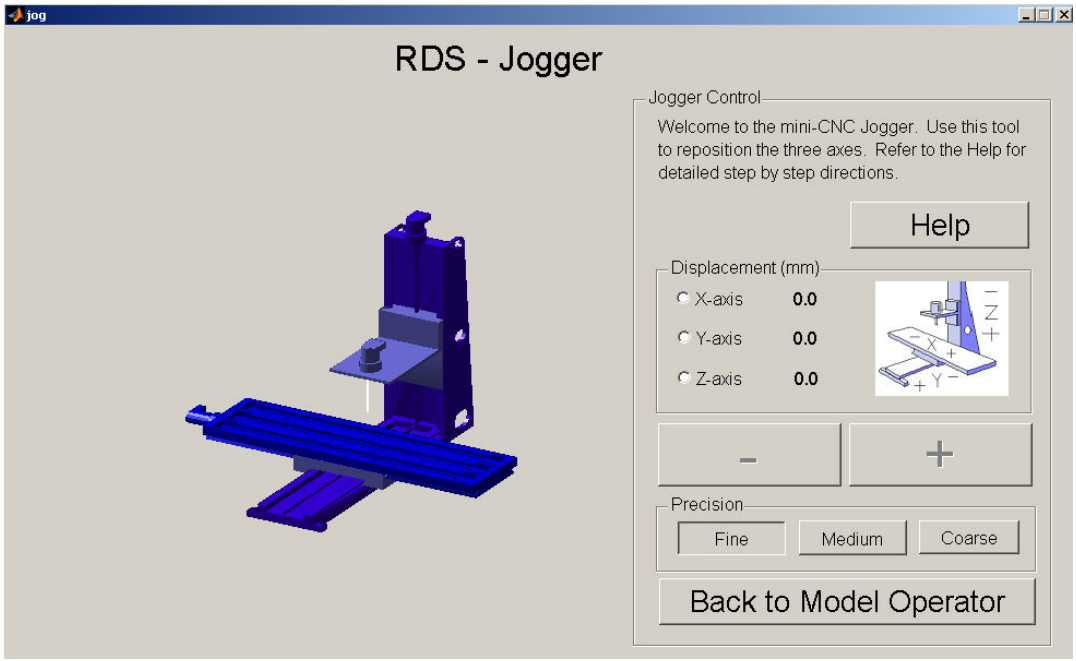

Figure 23: Linear Axis RDS Phase 2 GUI Jogger.