The Use of Real Time Operating Systems and Development Tools as a Mean to Revitalize Computer Engineering Programming Courses

Halima M. El Naga
California State Polytechnic University, Pomona

Samuel J. Stokes
Academic Developer Evangelist
MicroSoft

Nagi M. El Naga,
California State University, Northridge

Abstract

In programming courses for electrical and computer engineering, it is difficult for the students to cognitively connect that abstract languages will eventually lead to the ability to build real hardware. The problem is that the students are used to fully functional software, and are not fascinated with the implementation of very simple programs. Often, it takes them over a period of 18 months, before they are able to write useful programs that solve real world problems. Now, if a simple iconic language, such as the one included in the Legos™ Creative Robotics system, is first introduced to them that will be the first step in teaching them the use of software to create a simple machine controllers and robots. With the release of Visual Studio 2003, and the Softwire™ Technology, it is possible for the students to experience the same creation of controllers and robots, within the Legos set of hardware. Because Visual Studio 2003 is tightly integrated with the Windows CE™ hardware, systems developed and tested using the Legos set of hardware, can be transferred over to more advanced controllers using the Windows CE based microcontrollers. In this manner, the beginning Electrical/Computer Engineering student is able to begin programming with tools that are iconic in nature, review how the iconic language creates useful code, observe it's use in inexpensive hardware components, and then move to building actual engineering grade products using Windows CE type of single board computers.

In this paper, the use of the Softwire Technology Iconic language and the ability for beginning Computer Engineering students to make the cognitive connection between abstract languages and actual building of software, and deploying the software to an embedded hardware controller with actuation of servomechanisms and reading sensors will be discussed.

1. Introduction

The use of the Legos Mindstorm™ allows the students to utilize an iconic based language for sensors and servomechanism control; however, the engineering students resist the movement to languages such as C after this experience. Now, is it possible to build a curriculum that starts with the concepts of iconic programming, move to system architecture, operating system design/ maintenance, and finally to a realistic senior project that the students need to design? If the concepts covered in the total curriculum are first briefly introduced to the students in introductory course, then this course could mimic what the students will experience during the total curriculum.

Since much of the Computer Engineering curriculum introduces the idea of machine control, hardware architecture, and low level design. The goal of this introductory course is to start with the knowledge which the students may have gained at home or school during their K-12 education then extend it by using the iconic based language of Softwire™ Technology to control test equipment which is similar in many ways to Labview™. Softwire™ is a Microsoft Partner product (made available free through the Microsoft Developer Network Academic Alliance, in partnership with Softwire™). Then by using the Legos Mindstorm™ the students will be able to gain an introductory look at event driven programming at a very basic level. Finally, Visual Studio 2003 is tightly integrated with the Windows CE ™ hardware, a final extension, via a simple lab, the students would be able to create a simple program based on the one that they used with the Legos Mindstorm, Softwire working with the Legos Robotics system, and then use this program within Windows CE, an embedded and real-time operating system. Over the course of the student's later classes, they will be able to expand on the work they did in this introductory course. The systems developed and tested using the Legos set of hardware, can be transferred over to more advanced controllers using the Windows CE based microcontrollers. In this manner, the beginning Electrical/Computer Engineering student is able to begin programming with tools that are iconic in nature, review how the iconic language creates useful code, observe it's use in inexpensive hardware components, and then move to building actual engineering grade products using Windows CE type of single board computers. The components of this introductory programming course are given in Figure 1.

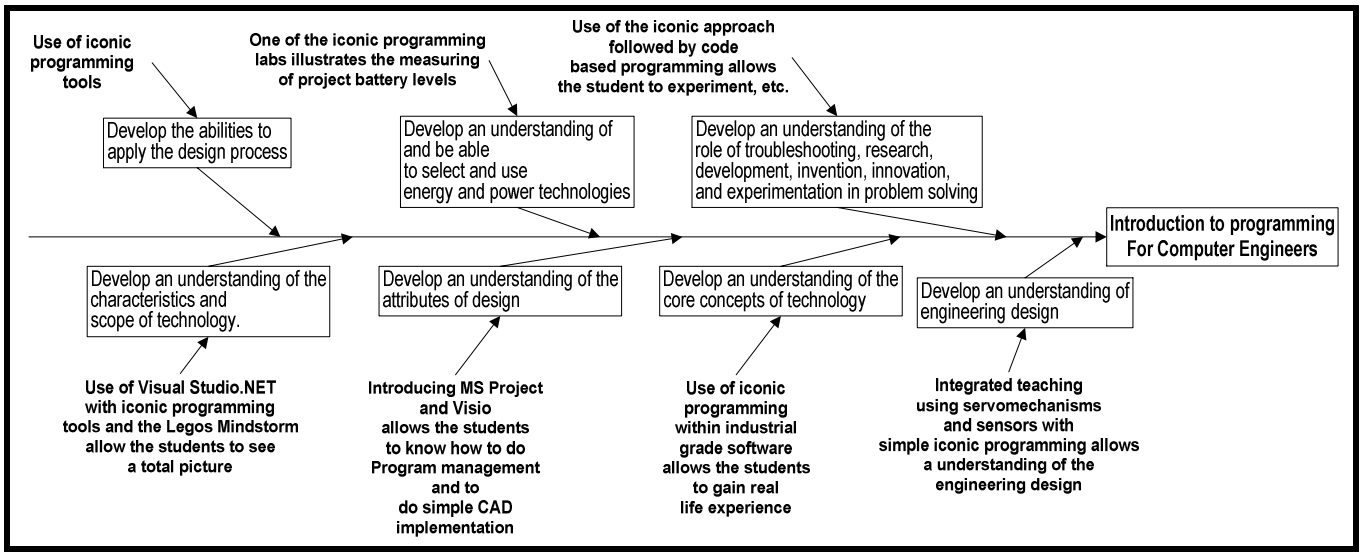

Figure 1 Components of the Introductory Programming Course

Later classes in the second and third years would introduce CPU design at the field programmable gate array (FPGA) level, using the same servomechanisms and sensors used in the introductory classes, thereby the students experience timeframe continue to grow rapidly. Classes in parallel to the computer engineering curriculum such as Physics

can also be implemented in such a manner as to make use of and extend the experience gained in the introductory course. For example, a physic component on dynamics would use the engineering characteristics of the servomechanisms and sensors to examine ideas such as inertia and friction. Later classes in the junior and senior level engineering would integrate the earlier work with 8 bit processors into a higher level embedded processors system using a 32-bit system that has a well defined architecture and design systems using tools similar to that used by the industry.

II. Electrical and Computer Engineering Introductory Programming Course.

This course is an introduction to Iconic Programming using servomechanisms and sensors. In this course, the Legos Mindstorm and Softwire™ iconic languages will be used to build simple robotic systems, to gain familiarity with simple programming techniques and to set up simple experiments. Using this language, the students will learn programming through a number of modules. Examples of these modules and their goals are explained in the following sections. In the first three modules only Legos Mindstorm iconic language is used while in the rest of the module Softwire™ iconic language is also used.

II.1 Module 1 Learning goals: Implement a simple programming statement that responds to external input and generates an visual output

In the first module, the student would implement a simple robot that would respond uniquely to touch sensor inputs. A program would be written in the Legos Mindstorm Iconic Language and an output such as a light or motor would be used to give feedback.

Lab: Create a simple controller that implements the ability to read input from sensors and to output a human readable response to the input from the sensors. Students will discuss how humans interact with sensors; define latency, and the use of Iconic Programming systems.

Building the Robot:

At the end of this module the student would be able to explain the purpose of an "If" branch in programming, how to deploy a program and implement a simple input and output hardware design. In this figure, the Legos Robot was built using a touch sensor and two motors, most students can complete the construction in less than 20 minutes, or the robot could be built ahead of time. A simpler project would be to use the touch sensor with the brick, and not use the motors, and the time involved in building the simpler robot is very short.

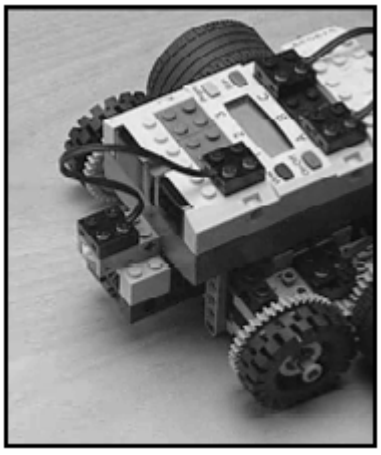

Figure 2: Simple Lego Robot

Programming the Robot:

A partial screen shot of the Legos Mindstorm™ 2.0 programming environment is shown in Figure 3. In this case, the simple robot beeps when the touch sensor is pressed. Therefore, the students get audio and tactical feedback on what an "If" statement does. To manage the students' experience, the professor would use a series of questions that would ask the student to explore the programming environment.

Modifications to the robots programming is very fast and the download to the robot is clean, with positive tactile and audio feedback. Within this early exercise it is possible to implement a simple form of Macros; the student would use the "Small Blocks" to build larger, reusable code blocks that would be stored inside of the "My Blocks" package.

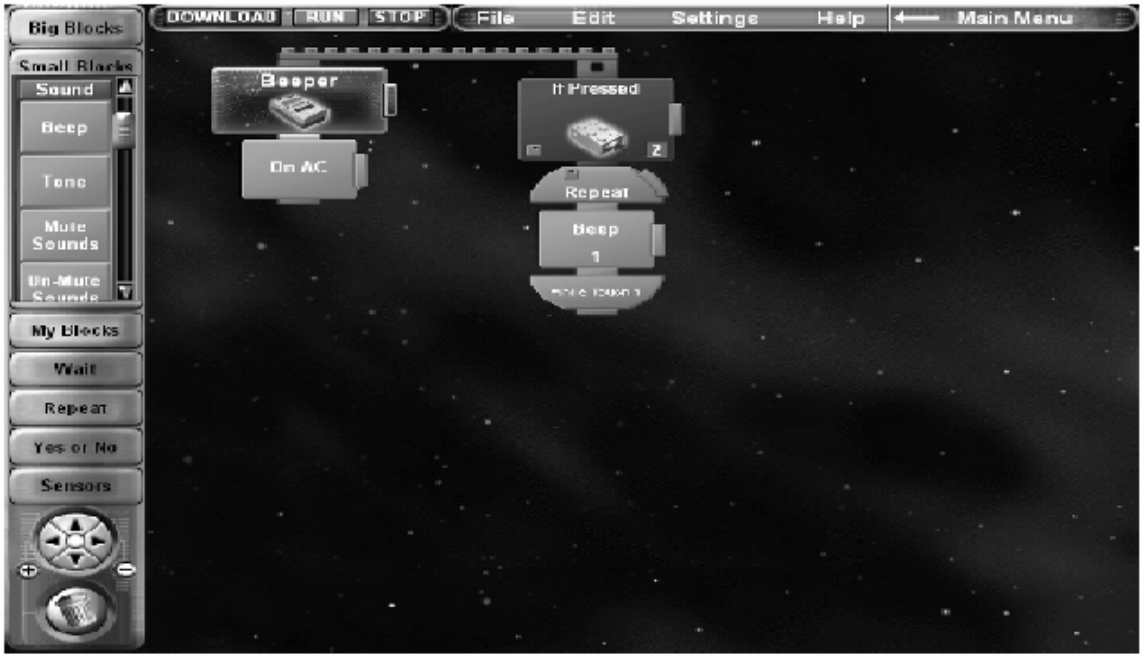

Figure 3: Legos Programming Environment

Expected outcome:

The student will have a visual and tactile interaction with an "If" statement. The Professor would be able to challenge the students to create more complex structures using the "If" statement at this point.

II.2 Module 2 Learning Goals: Adding the ability to respond to a varying environment, using a loop

In this module, the student will implement a simple robot that will use a touch to control audio output. In this case, the program will use a looping process to monitor the sensor input.

Lab: Create a simple controller that implements the ability to loop until an input from a touch sensor is received.

Programming the robot:

In this program, the student uses a repeat block to experiment with looping. As an extension, the student would be challenged to use a "Wait until" statement. In this case the student gains experience with the concepts of the "While loop" programming type of statement.

Figure 4: Legos Programming Environment

Expected outcomes

At the end of this module the student will be able to explain the concept of looping within the context of Input/Output from sensors. A challenge exercise could be given to build a simple tone generator that changes frequency when the touch sensors are depressed and released, using the "Repeat" block.

II.3 Module 3 Learning Goals: Using motors and optical inputs to control a mobile project

Building the Robot:
In this module the student's will build a mobile robot, with two motors and an optical sensor. There a number of robots that can be built, the robot in the picture is very simple, there are other configurations that would illustrate other types of engineering technologies such as mechanical engineering using gears, pulleys, etc.

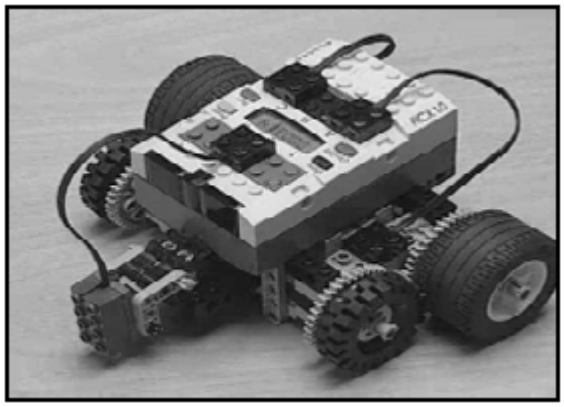

Figure 5: Robot set up for optical input

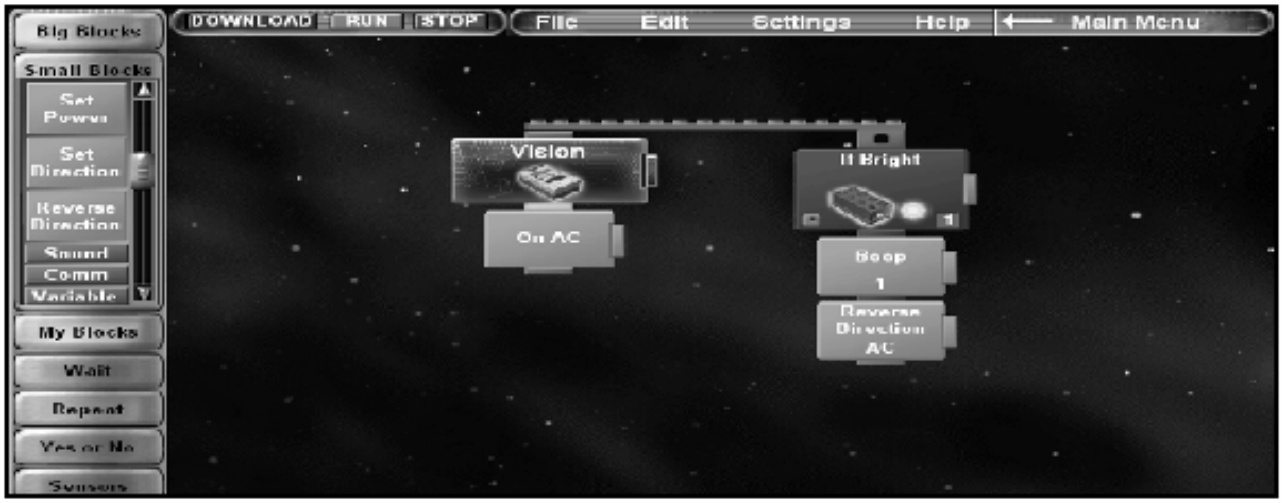

Figure 6: Programming for an optical sensor

Programming the Robot:

In this case, the student will set the brightness level for the optical sensor to sense as "dark" and then the sensor will respond to the contrast by beeping, and if connected to a motor reverse the motor voltage.

Expected Outcomes:

The student in this case would learn how to work with optical sensors, and gain an understanding of adjusting threshold levels. A challenge exercise would be for the students to use an "If" statement to find a block of a certain color or gray scale.

II.4 Module 4 Create and use a human interface with the Lego Mindstorm:

In this module, the student will implement a graphical user interface that is capable of controlling a Legos Mindstorm Robot [4]

Building the robot:

In Module 3, a mobile robot was created; the student will reuse this robot in this lab. Creating the Graphical User Interface: In this case, the students will require access to the Visual Studio.NET and Softwire™ tool set, both of which are made available through the MSDNAA program.

Building the robot:

The mobile robots used in the earlier labs are used in this lab.

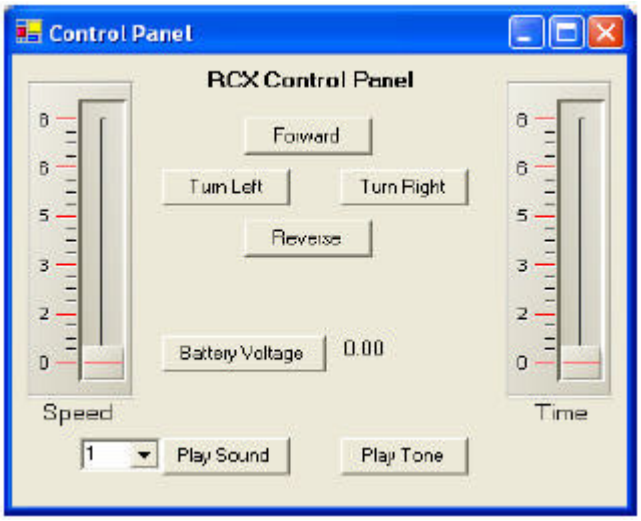

Figure 7: Graphical User Interface

Programming the robot:

The student will create a machine control panel that will control a robot in performing certain visual and aural tasks. In this lab, expect that the student will require extra help to be able to implement the user interface, and the "Softwire" virtual programming wires.

A sub set of the instructions are shown, the complete set is available in reference [3]:

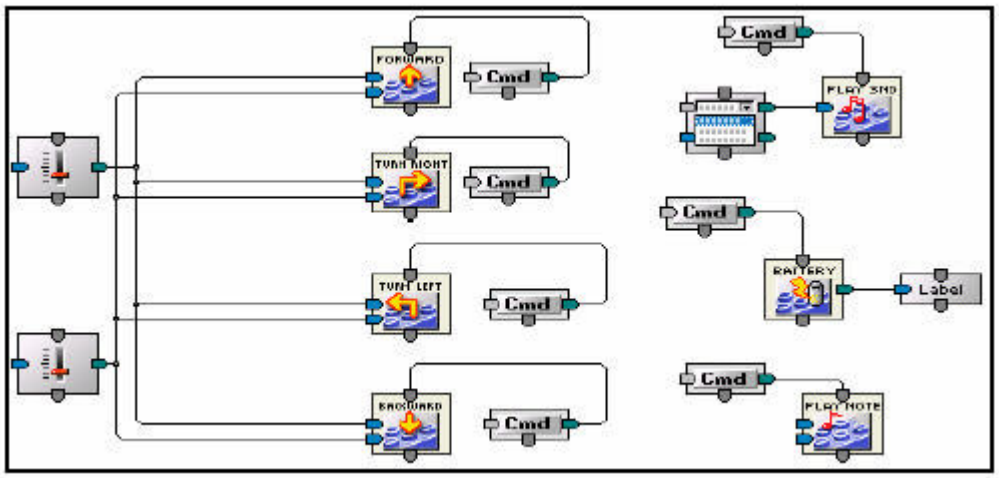

Figure 8 : Softwire programming for the graphical user interface

Adding controls to move the robot forward

1. Create a new Softwire VB Exe project in Visual Studio .NET.

2. From the GUI tab on the Softwire project, add two Sliders, and one Button. Position the Sliders as shown in Figure 7. The Button you just added will become the Forward button shown in Figure 7.
3. Right click on each of the sliders and choose "Properties". Then choose "Scale" and change the MaxValue to 8.
4. Add two Label controls from the Visual Basic Toolbox.
5. Move each label into position below a Slider control.
6. Right click on one Label and choose "Properties". Change the text property to "Speed".
7. Right click on the second Label and choose "Properties". Change the text property to "Time".
8. Now choose the Lego tab and add 1 Move Forward control.
9. Connect a wire from the "Value" output on the button to the "Control In" on the Move Forward control.
10. Take the "Output Value" pin on the Slider labeled "Time" and connect it to the "Time" input pin on the Move Forward.
11. Drag a wire from the "Output Value" pin on your Slider labeled "Speed" to the "Speed" input pin on the Move Forward control.
12. Right click on the Button and choose "Properties"
13. Set the Text property to "Forward". This should change the name on your Button to "Forward".
14. Test your program for functionality.

II. 5 Module 5: Binary tree robot

Students will populate the tree with values and give the program a value for which to search. As the tree is searched for the specified search value, the robot will travel along the tree. The project will teach students how binary trees are searched and will reinforce the use of "If" statements. In this project, the robot is demonstrating the way a binary tree is searched. The robot is not actually searching; the searching takes place in the program the program directs the robot to move in a way that simulates the traversal of a binary tree.

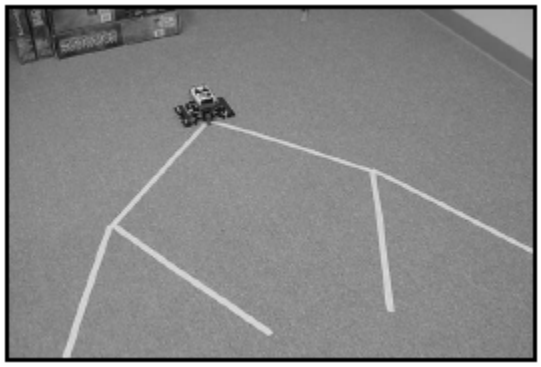

Figure 9: Robot and binary tree made from masking tape

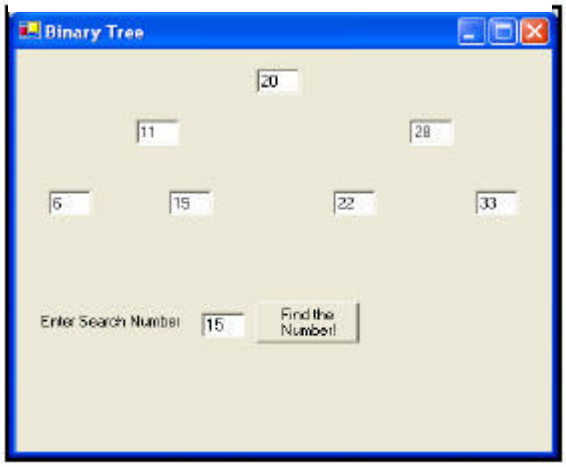

Figure 10: Binary Tree Form

Programming the Robot:

The student will learn how to connect the controls in such a manner as to control the robot to simulate the nature of a binary search. This requires the use of a number of "If Then Else" control blocks.

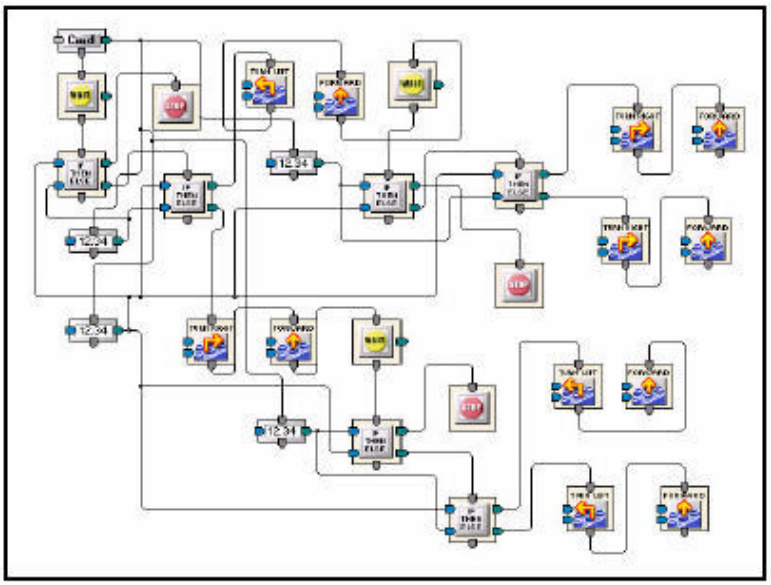

Figure 11: Binary Tree program

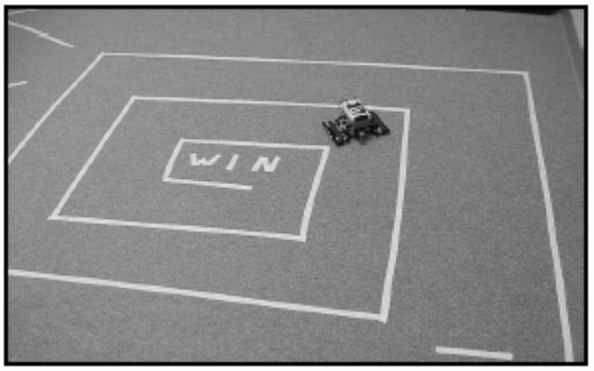

Figure 12: Maze robot, uses a mobile robot with optical sensor

II. 6 Module 6: Maze robot

The student's will create an optically guided robot that follows a maze made from tape. In this module the student will create a control program using the SoftWIRE™ technology which will use an optical input sensor to control the mobile robot by reading the contrast between the floor and tape, the student will control the robot to avoid the lines via an infrared transmitter. The student will use one of the previously built mobile robots.

Designing the graphical user interface:

The student is assigned the task of designing a graphical user interface that will start the game, track the amount of time and speed, as well as control the robot. In this manner, the student is able to connect with previous experiences of using a remote control car or other toy from their youth. A possible solution is shown in Figure 13. The student gains experience in human interface design. For example, the student would be challenged as to the reason behind using the slider control over a combo box.

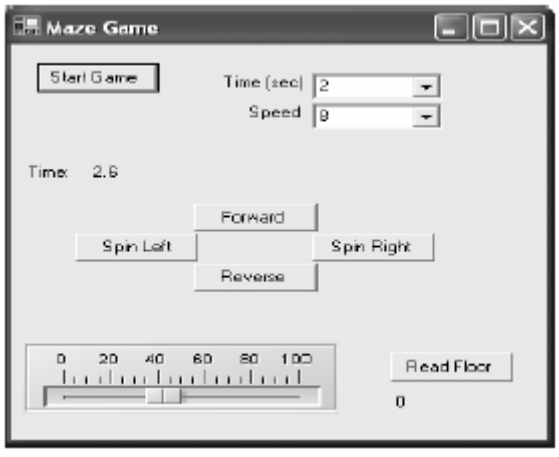

Figure13: Graphical User interface to control the robot

Programming the robot*:*

In programming the robot, the student will need to consider how the robot's optical sensor is able to detect the tape in contrast to the floor. An "If Then Else" statement is used to determine the contrast in floor versus the tape.

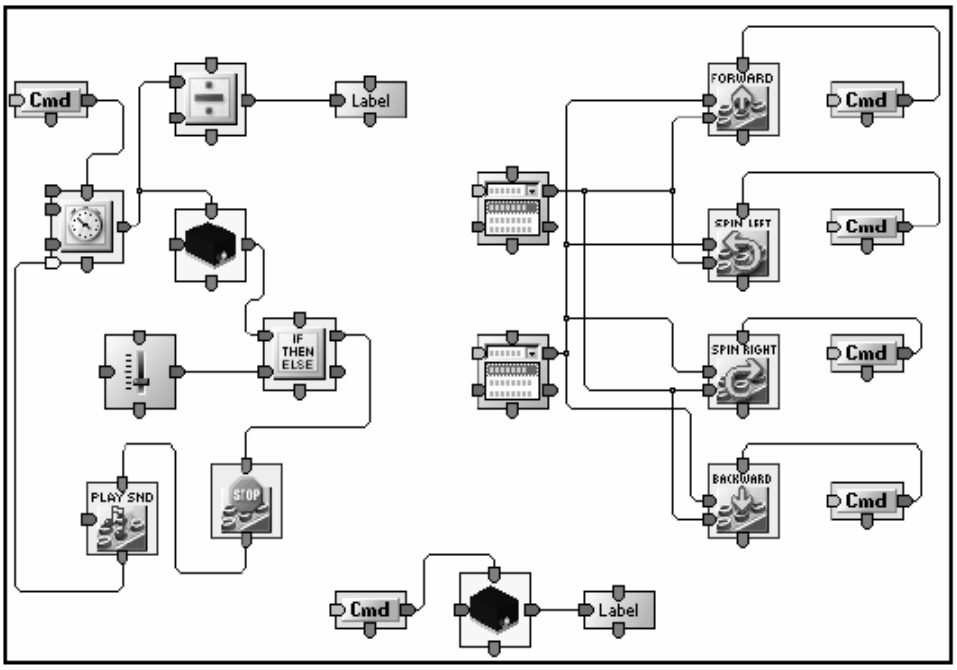

Figure 14: Programming the maze robot

II. 7. Module 7: Create an elevator robot Building the Robot:

In this case the robot is a simple winch like device, and models real-life situations found in industry, military and agricultural environments. This device is similar to the one shown in Figure 16. There are a few engineering issues such as static and dynamic forces.

Designing the Graphical User Interface:

The graphical interface, shown in Figure 15, is simple, but there are a number of considerations for the student. In designing the keypad, could an operator be confused by the position of the 10, 11 and 12 key, since this is the normal configuration for a telephone buttons?

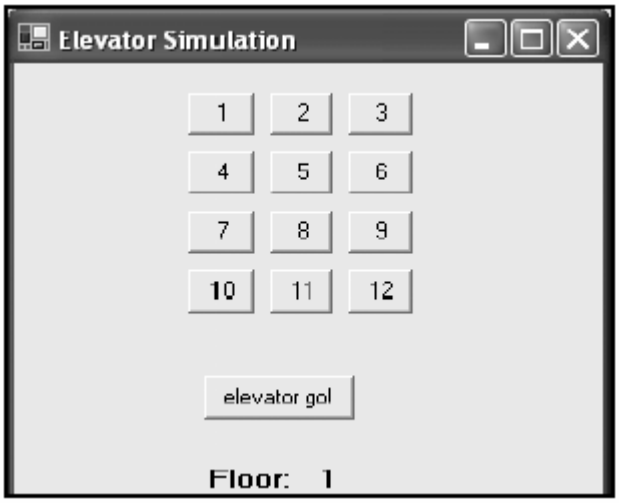

Figure15: Graphical User Interface for the Elevator Robot

Programming the elevator robot:

In this case, the student will need to consider how to store the value of the floor selected prior to the operator presses the "elevator go!" button. This shows the student the reason for using a variable in a program.

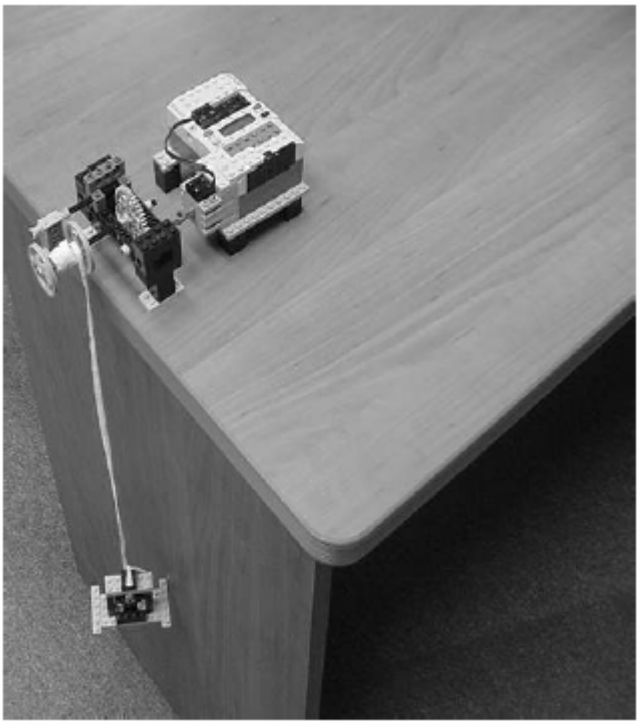

Figure 16: Elevator Robot configuration

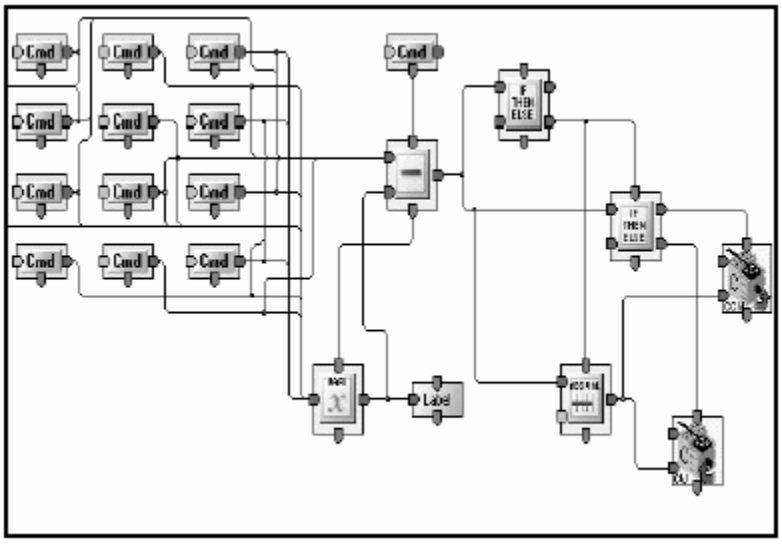

Figure 17: Softwire programming elevator

II. 8 Module 8: Creating a program that demonstrates Arrays, Looping and multiplexing

In this module, the students will "program" their robot by entering a sequence of robot instructions into an array. When the stored sequence of instructions is run, the robot will

execute the movements that have been entered into the array in the order in which they were entered. The project will reinforce the concepts of arrays, loops, and multiplexers.

Creating the graphical user interface and programming the robot:

In this case, the students use a number of active controls such as combo boxes and list boxes. The student is able to understand how to use arrays to store servomechanisms control information.

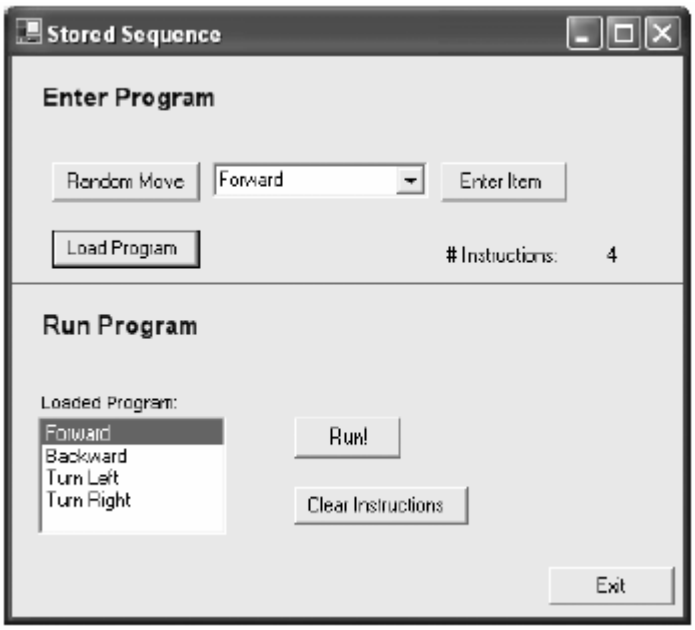

Figure 18: Stored sequence graphical user interface

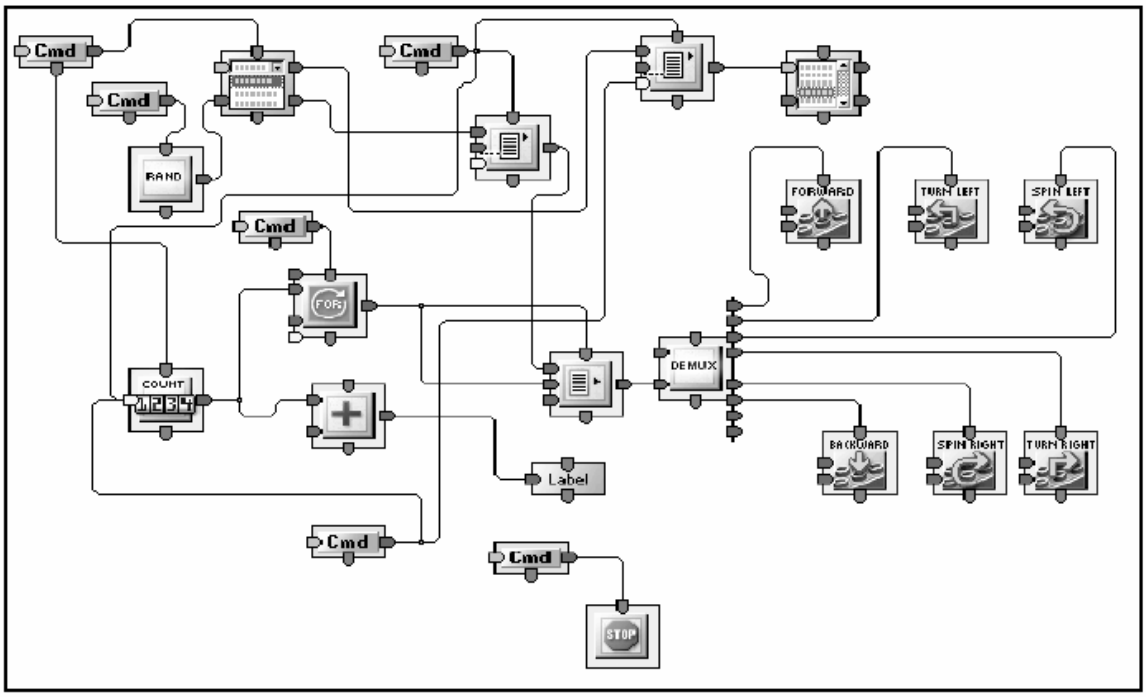

Figure 19 A Program Creating an Array

III. Conclusion

In this paper, samples of projects that could be assigned in an electrical and computer engineering introductory programming course is presented. These projects are based on the use of the Iconic Programming using servomechanisms and sensors. In this course, the Legos Mindstorm and Softwire™ iconic languages are used to build simple robotic systems, to gain familiarity with simple programming techniques and to set up simple experiments. Using this language, the students will learn programming through a number of modules. Examples of these modules and their goals are explained in the previous sections.

As the students are moved through the modules, they are also gaining insight as to what they will be covering in later classes in Computer Engineering. In this case, the students that are from cultures that may not understand what an Engineer is required to know, they will have a better idea of the expectations of the discipline. Since the roots of this introductory class is based on the previous experiences as teenagers, just as calculus, physics, etc. is, the student will be able to see the breath of the technology and that what they are learning is "real-world".

Bibliography

1. Jason Gilder, Michael Peterson, Jason Wright and Travis Doom. "A versatile tool for student projects: an ASM programming language for the Lego mindstorm ," Journal on Educational Resources in Computing, Issue 1, Volume 3, March 2003.
2. Kathryn Holliday-Darr, Michael Lobaugh, Penn State Erie,"Graphic Claymation – Visualization Through Sight And Touch**,"** Session 1606, Proceedings of the 2003 American Society for Engineering Education Annual Conference & Exposition, Nashville, Tennessee, June 22-25, 2003.
3. "Project One: RCX Control Panel – Teacher's Guide," www.softwire.com.
4. Dave Baum, Michael Gasperi , Ralph Hempel, Luis Villa , David Baum, "Extreme Mindstorms: an Advanced Guide to Lego Mindstorms," 1st edition, January 1, 1970, **ISBN:** 1893115844, APress.

Biography

HALIMA El NAGA

Halima El Naga is an Associate Professor of Electrical and Computer Engineering at California State Polytechnic University, Pomona. Her research interests include parallel processing, computer architecture and memory systems for shared memory multiprocessors. She received her B.S. degree in Electrical Engineering from Ain Shams University in 1977, M.S. in Electrical and Computer from California State University, Northridge in 1987 and her Ph.D. in Computer Engineering from the University of Southern California in 1999.

SAMUEL STOKES

Samuel Stokes is an Academic Developer Evangelist for Microsoft, Corporation based in the Southern California region. He works with 10 universities, one of which is California State University, Northridge. His previous work has been in the design of the Navigational Data Unit for the Global Positioning Satellite, Automatic Flight System for the MD-11 commercial airliner, which included being on the first flight of the aircraft as well as working with the pilots during the following 18 months. He has consulting on a large number of embedded systems in the 1990s and continues to consult with the Microsoft Windows CE and Embedded XP team. Recently he was an adjunct professor for Chapman University in Southern California.

NAGI ELNAGA

Nagi El Naga is a Professor of Electrical and Computer Engineering at California State University, Northridge. His research interests include computer architecture, computer arithmetic, multiprocessor performance evaluation and Error detecting and correcting Systems. Dr. El Naga worked as a consultant in the area of digital system design. He received his B.S. and M.Sc. degrees in Electrical Engineering from Ain Shams University in 1970 and 1974 and his Ph.D. in Computer Engineering from the University of Waterloo, Canada in 1978.