# Scorbot ER-III Robot

**by**

**Saeid Moslehpour, Ph.D., Candace Odom, Tyrell Barrett and Matt Brown**
College of Engineering Technology, and Architecture
Department of Electrical and Computer Engineering
University of Hartford
West Hartford, Connecticut, 06117

**Abstract -** *The purpose of this project is to develop a new control system for the Scorbot ER III robot. The previous method of programming this robot was to use the control software called SCORBASE, an old DOS based program that was capable of controlling the robot as well as writing programs for the robot to execute. For our project we will be replacing the control software with a more efficient, stable, and simpler one. This control system will work with the current hardware and be implemented using LabVIEW 8.0. LabVIEW is the perfect program to implement such a control system because it provides the ability to monitor the various control signals coming from the robot as well as create user friendly interfaces. A series of lab experiments will be designed to drive the Scorbot to its maximum potential.*

**Keywords-** LabView, Scorbot, Mechatronics, serial port, RS 232 and ACL

## Introduction

The idea for this project began with a meeting among our advisor Professor Moslehpour, Claudio Campana, and Professor Devdas Shetty. We met with them because we were looking for ideas for a senior project. One of the ideas they suggested was a project involving a robot called the Scorbot. The Scorbot was one of several robots that had been sitting unused in the basement of Dana Hall. The robot was completely functional but had not seen any real use since 1992. Professor Shetty suggested that we should take this robot out of the basement and find some use for it. Upon his suggestions we decided that creating software that interfaces the Scorbot to a computer would make a good project. Also, this project could benefit future students who are interested in studying robotics and Mechatronics.

LabVIEW, because of its ability to easily integrate hardware with software, was chosen as the language to implement this new software. Another advantage of LabVIEW is its ability to create very asthetically pleasing graphical interfaces. By having a simple user interface, one can control the movements of the robot by the click of a button. LabVIEW uses the VISA API for controlling communication with external serial devices. The beauty of VISA is that it allows you to communicate with a variety of instruments

without having to learn a different language for each protocol. Users will be able to control the speed, angle, and direction in which each armature moves.

The Scorbot contains five motors. Each motor controls different aspects of the robot's movement. The goal of this project is to gain an understanding of LabVIEW and to develop a program that will allow future students to work with the Scorbot. Along with learning LabVIEW, this will require us to learn about ASCII code, Serial RS-232 communication, and DC motor control.

## Statement of Problem

The purpose of this study is to replace the old, buggy DOS based control software with a new version implemented with LabVIEW 8.0. The new control software will be much more reliable as well as easier to use than the original. This software will allow future students to get a feel of how to control the robot. Also, this software provides a good starting point to allow future students to further develop our program to increase the functionality of the robot.

## Statement of SubProblems

1. Learn LabVIEW 8.0
2. Determine the appropriate command set to control the robot.
3. Discover the proper timing and transmission requirements to program the controller.
4. Develop a user interface that will allow users with little experience to control and program the robot to perform simple tasks.

## Definition of Terms

**ACL-** (Advance Control Language), is an advanced, multitasking robotic programming language developed by Eshed Robotec. ACL is programmed onto a set of EPROMs within the Controller and can be accessed from any standard terminal or PC by means of an RS232 communication channel.

**Microprocessor -** A master control circuit, usually on a single chip, that performs arithmetic, logic and control operations, customarily with the assistance of internal memory.

**LabVIEW –** A graphical program development application developed by National Instruments in 1986 to integrate engineering tasks such as interfacing computers with instruments, collecting, storing, analyzing, transmitting measured data, developing programs in a graphical environment and providing an effective user interface.

**RS232 cable -** a standard for serial binary data interconnection between a DTE (data terminal equipment) and a DCE (data communication equipment). It is commonly used in computer serial ports.

**Serial Port -** is an input/output connection on the computer that allows it to communicate with other devices in a serial fashion-data bits flowing on a single pair of wires.

**SCORBASE Level 5 -** is a robot-control software package that is supplied on diskette with the controller. Its menu-driven structure and off-line capabilities facilitate robotic programming and operation. SCORBASE runs on any PC system and communicates with ACL, the controller's internal language, by means of an RS232 channel.

**HyperTerminal -** is an easy-to-use terminal emulation program that has shipped with every version of Windows since the original release of Windows 95. Terminal emulators let a PC operate as if it were a computer terminal.

**Free Serial Port Monitor 3.31 –** This program monitors data coming from the serial port. This was the program we used to reverse engineer the Scorbot command set. The logs from this program can be found in the appendix.

## Review of Literature

In 1999, Brendon Wilson, a student from Simon Fraser University, completed a project entitled "DESIGN OF AN INTEGRATED ROBOT SIMULATOR FOR LEARNING APPLICATIONS." The REMOTE (Really Exciting Manipulator Object Tele-learning Experience) project is investigating new methods of providing hands-on experience to students using tele-learning. Tele-learning is a method of providing distance education using multimedia technologies via any combination of computer, telephone, video-conferencing, or Internet connections. The purpose of this project is to create a Java application that enables students to explore robotics programming via interactive experimentation. The completed simulator environment features a graphical user interface that displays visual simulation feedback and allows users to edit, program, and debug simulation files. The application's underlying simulation engine controls the internal representation and simulation of the robot manipulator model. Future additions to the simulator will allow students to upload completed simulation programs to a remote robot, and view the results via the Internet.

The next project researched was called "LabVIEW-Controlled Robot Climbs and Inspects Highway Lighting Towers" by Kurt Hudson. The challenge to be completed was automating and improving the inspection of high-mast lighting towers to evaluate structural integrity. The solution was creating a PC-based robotic crawler and controller for remote inspection using motion and vision hardware controlled with LabVIEW. The Virginia Transportation Research Council contracted research at the University of Virginia and Virginia Technologies, Inc, to develop a mobile robot to perform the inspection of each pole joint. The objective is providing a more complete inspection of the pole joint for cracks and other signs of fatigue without requiring visual inspection by an operator suspended near the pole. The system envisioned would include both remote visual and ultrasonic inspection capabilities. A portable computer with a graphical user

interface would provide an operator with complete control of robot movement and the capability to view and store images of the pole surface.

Sandor J. Toth's project called "NEW PC AND LABVIEW BASED ROBOT CONTROL SYSTEM." presents a new economical solution of a robot-control system for different sophisticated robot applications. The control system is based on a PC-controlled servo motor control card and an intelligent control software, which has been developed using high level graphical programming language (LabVIEW). The basic development is interface software for making connection between the control card and LabVIEW. LabVIEW gives a wide range of opportunity of utilization of the developed control system at different robot applications

A project called "ROBOT MOTION REALISATION USING LABVIEW" by Alexei SOKOLOV. We begin by explaining how any kind of robot motion can be defined by the path and the velocity along it. Various kinds of motion (time optimal, motion with constant kinetic energy, process optimal, etc.) are given by different approaches. Therefore, there is a reason to implement robot control device that can realize the motion along a given path with arbitrary velocity profile. Generally speaking, it is rather difficult to solve this task using the existing control units of industrial robots. The LabVIEW Virtual Instrument technology gives an opportunity to build different sophisticated control systems. Such an approach makes the system rather more flexible than the existing ones and allows users to get information, using the LabVIEW interface, about the inner processes in convenient form. All these advantages are very important for experimental investigations of different control methods. Using this control device time optimal robot motion was realized. It is also possible to realize robot motion along the given path with arbitrary velocity profile. The results obtained by the LabVIEW based control system can be applied to the investigations of the robot motion and to evaluate the goodness of different control principles.

The last project researched was called "WEB INTERFACED ROBOT ARM." This project will interface a Scorbot-ER V plus robot arm to the World Wide Web. Instead of using the robot controller that is supplied with the robot arm, a PC will be running and controlling the operations of the robot using LabVIEW. Data will be sent out of the parallel port of the computer and into the robot motors. A website will be created to talk to the computer that is controlling the robot, and then that computer will tell the robot what to do. Web cams will be positioned on the robot itself and used to guide the user during operation of the robot. Having a robot arm interfaced with the World Wide Web will allow any users to access and gain control of the Robot Arm from any place in the world with internet connection.

## Methodology, Hardware

The Scorbot ER-III is a JAR (Jointed Arm Revolute) robot, which means its movements are somewhat similar to that of the human arm. This robot can be very useful for industrial processes, especially where assembly lines and conveyors are involved. With this type of system, duties can be performed with a little more detail than with some other

robots. The movements made possible by the ER-III Scorbot allow for more concise orientations of products and enable it to work with a larger scope of products than a robot with a lesser degree of freedom. "Degree of freedom" is a term that describes the kind of movement a robot can make. Figure 1 illustrates schematic representation of the Scorbot ER III.

The ER III has three axes of rotation. This robot possesses five degrees of freedom. These degrees are illustrated through the movement of different parts which are

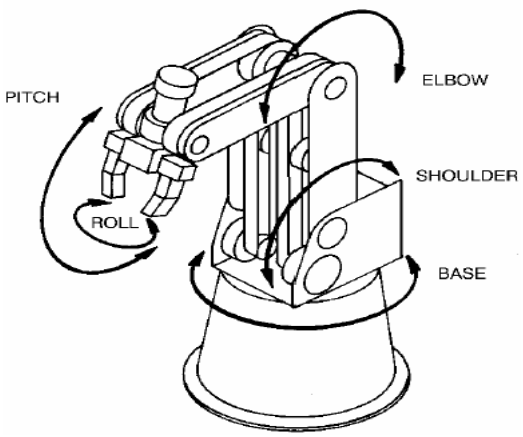


**Figure 1 Schematic Representation of the Scorbot ER III**

- Base- lower part of robot that rotates in the horizontal plane.
- Shoulder- connects to the base by way of a joint that rotates in the vertical plane.
- Elbow- connects to the shoulder by way of a joint and also rotates in the vertical plane.
- Wrist- this portion is connected to the elbow and gives the robot its final two degrees of freedom. The wrists movements are spilt into two sections, wrist roll and wrist pitch. The wrist roll is similar to rotating your palm so that it either faces up or down. This action is called wrist roll. Also the wrist may flex up or down as a human wrist would. This action is called wrist pitch.
- Gripper- this end effector is attached to the wrist and is capable of opening and closing. It emulates the action of a human using his/her index finger and thumb to grasp an object. This robot is powered via electric motors.

Figure 2 illustrates the Scorbot side view.

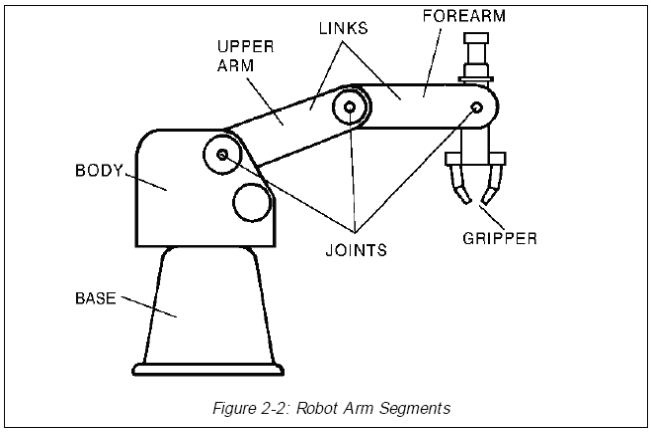


**Figure 2: The Scorbot side view**

**Robot side view**

- 6 D.C. servo motors with closed loop control
- The manipulator arm has five axes plus gripper with optical encoders on all axes.
- It has a load capacity of 1 kilogram and a repeatability of 0.5 mm and 10 different speeds with a maximum speed of 330 mm or 13 inches per second.
- The gripper is fitted with sensors to allow for measurement of objects as well as touch control.
- The Scorbot can be directly controlled by a hand-held teach pendant plugged into the controller. Also the controller provides 8 inputs as well as 8 outputs which can be programmed to provide easy integration within the system's environment.

Figure 3 shows the Scorbot Controller.

**Figure 3: Scorbot Controller**

The main controller circuit operates the robot's motors, encoders, inputs, outputs, micro switches, test routine, and communication. The controller circuit is a slave of the computer, communicating by means of the RS232 serial port.

The main components of the controller unit:

- Central Processing Unit (CPU): INTEL 8031.
- EPROM: A fixed memory of 16K bytes which contains the controller's operational software.
- Logic Components and Buffers: Transmit information to the CPU and execute instructions relating to the motors, encoders, inputs and outputs.
- Multiplexers: Expand the CPU's capability for receiving input/output information.
- Drivers: Activate motors and outputs.Include switching components such as power transistors. Drivers also interface between low-power and high-power components.
- Serial Communication Components: Receive and transmit information in serial format. This controller uses the RS232 standard (12V).

Figure 4 illustrates the Scorbot Controller Rare view.

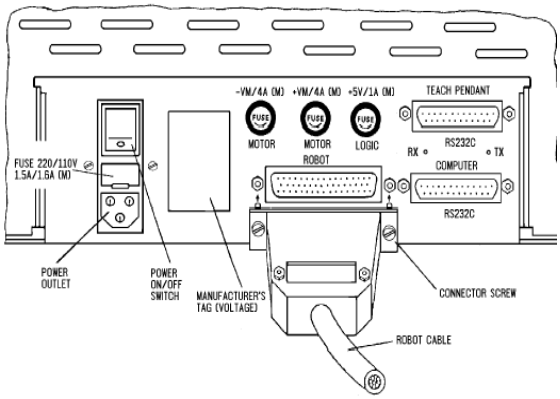


**Figure 4: Scorbot Controller Rare view**

## Display Circuitry

The display circuit is fitted to the cover of the controller housing. This circuit includes push button switches, I/O terminals and LEDs.

**Switches:**

- The MOTOR on/off switch for DC power to the motors.

- Two INPUT switches connected to inputs 1 and 2; these can be used to simulate inputs.
- The RESET switch resets the controller to its initial state.
- The LAMP TEST switch causes all the LEDs on the panel to light up.
- The MOTOR TEST switch initiates a diagnostic routine on the robot's motors.

**LEDs:**

- A green LED (POWER light) indicates 110 VAC (220 VAC) voltage.
- A green LED (on MOTOR switch) indicates DC voltage to the motors.
- Eight yellow LEDs show whether the inputs are on or off.
- Eight green LEDs show whether the outputs are on or off.

Figure 5 shows the Scorbot controller close-up.

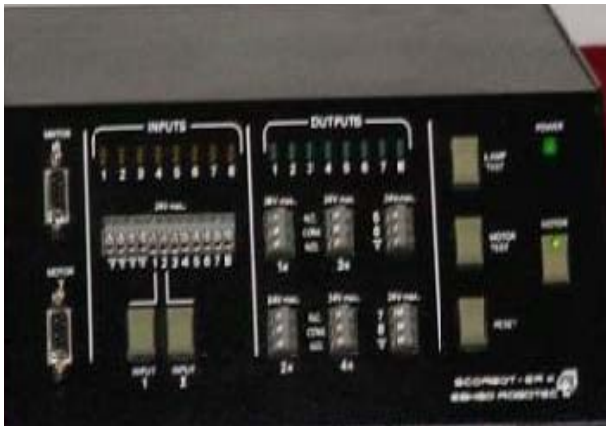


**Figure 5: Scorbot Controller Close-up**

## SCORBASE Software

SCOREBASE Level 5 is an advanced robotics control software package, for use with the SCORBOT robotic system. SCORBASE's menu-driven structure and off-line capabilities provide a user-friendly tool for robot programming. SCORBASE runs on any PC system and communicates with ACL, the controller's internal language, by means of an RS232 channel.

Some of the basic features of SCORBASE Level 5 include

- Control and real-time status display of up to 11 axes: 5 robot axes, a gripper and up to 5 peripheral axes.

- Full support and real-time status display of 16 inputs and 16 outputs, including user defined text for each I/O status.
- Position definition and display, as well as manual robot movement, in either joints coordinates (encoder units) or Cartesian coordinates (X, Y, Z, pitch and roll).
- Position and movement definition for robot and peripheral equipment either separately or together.
- Robot movement definition as regular, linear or circular with a 90 active speed settings.
- User-defined number of program lines (400 default).
- Interrupt programming for handling responses to changes in input status.
- Extensive variable programming.
- Saving and loading of programs and position tables either separately or together.
- Simulation of program execution in an off-line mode.

Figure 6 illustrates Screenshot of Scorbase software.

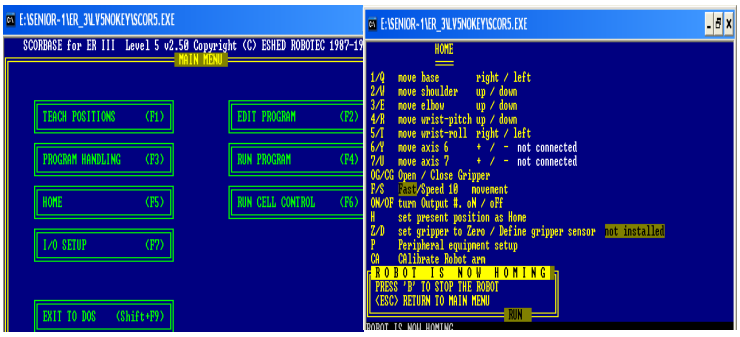


**Figure 6: Screenshot of Scorbase software**

## Controlling Methods

### Teach Pendant

The teach pendant, or control box, is a hand-held controller that allows you to operate and program the robot. This pendant plugs into the controller via the serial port. The teach pendant is best used for teaching the robot positions and writing short programs.

### Computer

For writing longer and more complex programs, it is easier to use **SCORBASE** (Level 3 or 5) after you have recorded the required positions with the teach pendant and **SCORBASE** Level 4.

### LabVIEW

LabVIEW stands for Laboratory Virtual Instrument Workbench. LabVIEW is commonly used for instrument automation and control. LabVIEW provides the flexibility of code to

work with real time signals as well as develop programs quickly and efficiently. This language is called G.

LabVIEW is a graphical language as opposed to being text-based. Execution is determined by a graphical block diagram as opposed to order in which the code is written. This language is a dataflow language, which means that a piece of code only executes when there is data present on each input. This type of language is highly multithreaded and supports parallelism. LabVIEW is a graphical language. Every object in LabVIEW is represented by an icon. These icons are located on menus called palettes. In order to place an object on the block diagram, an icon is dragged from the palette and dropped onto the diagram. In programming terms, each icon represents an instance of a particular class. Only when an object has data on all its inputs will it perform an operation. In the block diagram, variables and data are passed through "wires." Each wire represents a particular data type, such as strings, chars, or arrays. If two wires of conflicting data types are connected together, the wire is said to be "broken" and the program will not run.

Each program in LabVIEW is called a Virtual Instrument or VI. Each VI contains three parts, a front panel, a block diagram, and a connector pane. Each VI can run separately or it can be part of a larger program. One VI may contain several other VI's. Controls and indicators on the VI's front panel allow information to be passed into or retrieved from the program. The front panel is similar to the front form in Visual BASIC. The front panel is the part of the program the user interacts with. The front panel may contain buttons, indicators, controls, or graphs. LabVIEW has a number of different options for creating ascetically pleasing user interfaces. Here is the front panel from our program. Figure 7 shows the front panel.

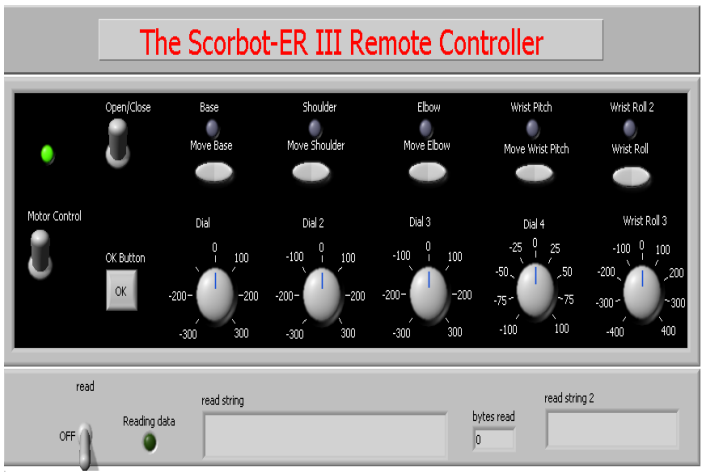


**Figure 7: Front panel**

The block diagram is where the actual "code" is written. For every control or indicator on the front panel, there is an icon on the block diagram. Controls are for inputting data while indicators are outputting data.

LabVIEW is a very deep and complex program for which there is considerable information available. For more information on LabVIEW visit www.ni.com or see references.

**VISA**

VISA stands for Virtual Instrument Standard Architecture. VISA is an industry standard in the area of instrument programming. VISA is a high level API that calls low level drivers. VISA is capable of controlling VXI, GPIB, or Serial instruments. VISA calls the appropriate drivers depending on what type of instrument is being used.

The beauty of VISA is that regardless of interface type, the VISA commands are the same allowing instrument programmers to communicate with several devices without having to learn several different languages. Thus, programs to be ported from one interface to another without having to change the code. VISA does this by strictly defining data types, so that programs remain unchanged despite being ported to computers with different size data types.

Also VISA is an object oriented language. This makes it easier to use when developing drivers for new instruments. VISA uses a very compact command set that provides all the functionality needed for instrument programming. For more information see reference.

**LabVIEW code development**

The first challenge we faced with this project was getting the computer to communicate with the controller. After spending several hours troubleshooting and going through the manual, we discovered that the controller requires a special RS232 cable and that it will not work with the standard RS 232 cable without some rewiring. Figure 3 shows the controller that the cable must have in order to work. Once we were able to get the software and hardware to talk to each other, we had to determine the command set. At first we used one of the previously written LabVIEW examples to write some data to the Scorbot. The VI was called Basic Serial Read and Write. We used the correct format listed in the manual but the Scorbot did not move. It was then we decided that we must reverse engineer the Scorbot command set.

We knew the old software would work with Scorbot, we just had to figure out what data it was sending out. We used two different methods to determine the command set. The first method we tried was to get a look at the commands using a program called Hyperterminal.

We did our test by running HyperTerminal on one computer and connecting it to another running the original SCORBASE software. We connected the computers using a (db-9 to db-9) serial cable with the transmit and receive pins crossed. That way all the data being written from the computer running SCORBASE could be displayed on the computer running Hyperterminal. We then instructed the controller to move a motor, and the command it sent out was displayed on the screen. This method worked but it was far from ideal and it was not enough to accomplish our goal. One of the problems with this method was that we had too much data and no way to save it. Hyperterminal did not

allow us to copy and paste the data to a file. Another problem with this method was that there were many undisplayable characters because Hyperterminal converts everything to a string. Another problem was that Hyperterminal only displayed one line of code at a time. Any new data just over wrote the old data. So basically this method gave a quick snapshot of the code but it did not work for discovering the command set. We needed a new way to see the code.

We decided the only possible solution was to use a sniffer. A sniffer is a program that records all data being passed between programs. After searching the internet we located a suitable program called Free Serial Port Monitor 3.31. This program was chosen for several reasons, first one being that it is free. The other reasons were that it was quite powerful and easy to use even with the free version. This program allowed us to see all data written and all data read from the controller. Also, it allowed us to what was being sent on each pin. Finally, it allowed us to see the data being written to the Scorbot and the Scorbot's response. It displayed this data in both string and hex format.

We ran our tests using this program along with the SCORBASE software. We hooked up the Scorbot to the controller and connected to the controller via the serial port. We then instructed the robot to move a specific motor and then we would log the results. We tested each motor three times and then reviewed the data. The logs from these tests can be seen in the appendix.

After reviewing this data, we able to discover the commands that make the motor move. Unfortunately, this command looked very different from what the manual said The manual was close but there were extra characters on the beginning and end of the command that we were not quite able to figure out. We decided to do some more research and we eventually found a lab involving a similar version of the Scorbot. The lab was posted by a teacher and was for a Visual Basic class. Included in the lab was a command to make the motor move. We instantly recognized it from our previous logs. We then tried outputting this command using LabVIEW, but we had no success. We were baffled. Since one of our members was proficient in Visual Basic, he decided to write a VB program that would test our command out. The command worked perfectly. Once we had one command working, it was easy to figure out the rest.

Once we had the command set down, we were on to our next challenge, getting the commands to work with LabVIEW. At this point we had a basic understanding, but we just didn't know enough. We then began researching LabVIEW and writing test programs. This procedure greatly furthered our knowledge of LabVIEW, but we still couldn't get the program to work. Then, while searching through the National Instruments website, we came across a program written in LabVIEW that worked with the Scorbot ER IX. The writer was having a similar problem to ours and he posted the solution along with his code. On a long shot, we decided to recreate his code to see if it would work with our version of the Scorbot. The read portion did not work, but the write portion did. We used this portion along with our command set and we were able to write LabVIEW code that successfully controlled the Scorbot. From that point on, it was a matter simply of designing a user interface to work with our code.


**Figure 8: The Front Panel**

Figure 8 is our front panel. This is the part of the program the user interacts with. To use this program is fairly simple. First, select how you want a particular motor to move by setting the dial. Set the motor control toggle to up. Then press the button above the dial. The motor will then move the distance determined by the state of the dial. The command being written to the Scorbot is displayed in the Read String 2 indicator box.

Each dial is set up so that the motor entire range of motion can be reached from any position. When the motor reaches its limit, a mechanical limit switch is pressed that deactivates the motor. Occasionally, the motor may stall; if that happens, press the reset button on the controller and then repeat the command.

To open or close the gripper, set the toggle switch up for open or down for close. Then press the button. If you desire to see the output from the controller, set the read toggle to on. The data read from the Scorbot will then be displayed on the read string indicator box. In the following section we describe the underlying code that makes this form work. Figure 9 illustrates LabViw bock diagram.

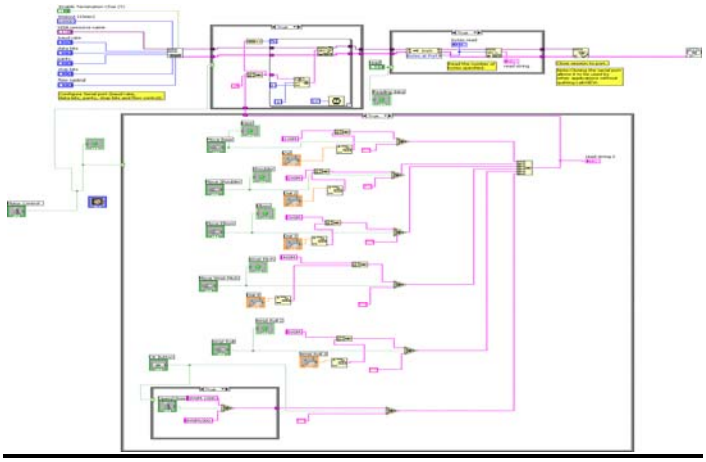

**Figure 9: The Block Diagram**

This code is broken up into two sections. One is the transmit and receive section. The other is the user interface section.

**Transmit and Receive section**

The transmit and receive section is broken into three pieces that are wired together.

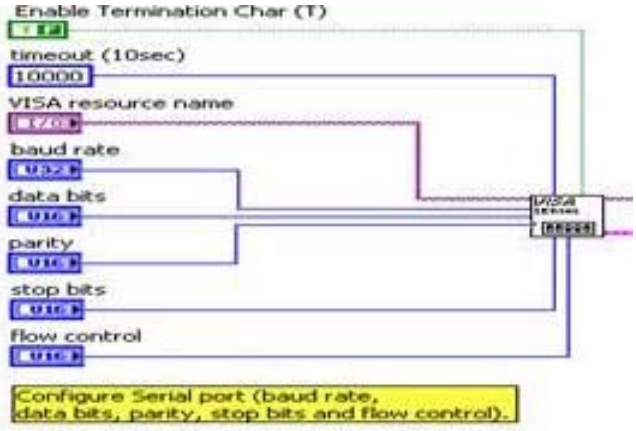


**Figure 10: Transmit and Receive Section**

This part of the block opens and configures the serial port. Here we set the resource name to COM1, the baud rate to 9600, the data bits to 8, the stop bits to 2, and the parity to none. These settings properly configure the serial port to communicate with the Scorbot. This section outputs the VISA resource name as well as the error out.

The next part of the program writes data to the Scorbot via the serial port. This section is shown in the Figure 11.

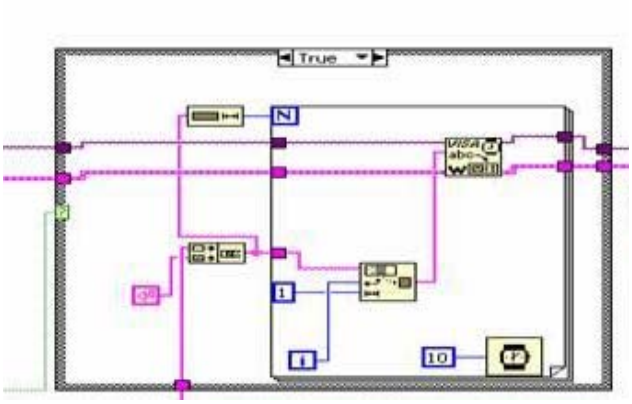


**Figure 11: Write portion**

This section is contained with a case statement. A case statement is essentially an if else statement. When this case is true, the code shown below will execute. If the case is not true, then none of this code will be active. The false case is not shown.

The state of this case is determined by a Boolean that is wired to the ? box on the frame. The wires entering from the left are the VISA resource name and the error in. The command to be written enters through the magenta wire from the bottom. The first thing that happens to the command when it enters this block is that it is concatenated with the carriage return command. The carriage return tells the Scorbot that the command is done being sent.

The block that looks like pages of paper is a "FOR" loop. The N symbol at the top left represents the number of times to go through the loop. The timer in the bottom right causes the loop to wait 10ms between iterations. The "i" symbol in the bottom left of the loop outputs the present index of the loop. This wire is wired to a substring command that takes in a string, a position, and a string length. This command outputs the string at the designated position. The string length is how many characters after the position to output. Our loop is set to output one character from the string at a time with a delay of 10ms. This command is then wired to the VISA write command, where it is outputted through the serial port to the controller. We obtained the idea for this section from a previous LabVIEW project involving a later version of the Scorbot robot. We located this previous project through the forums on www.ni.com.

The next part of the program reads data coming in from the Scorbot that is shown in figure 12.

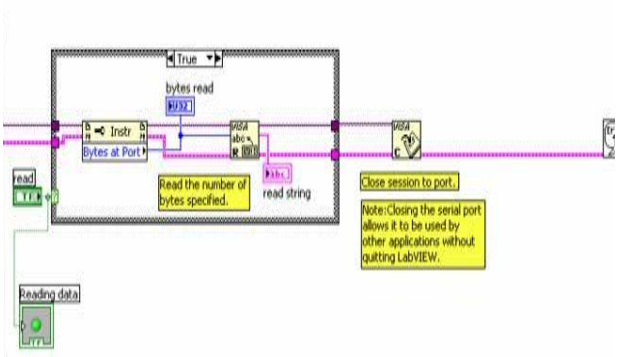


**Figure 12: The Read Section**

This block takes in the VISA resource name and error in as input. This block contains another case statement. When this statement is true, the read section will be active. Otherwise the VISA resource name and error in are propagated through the loop to the VISA close session command. This section displays on the front panel, the number of bytes of read, as well as the string read from the Scorbot.

**User Interface section**

Figure 13 illustrates the user interface section.

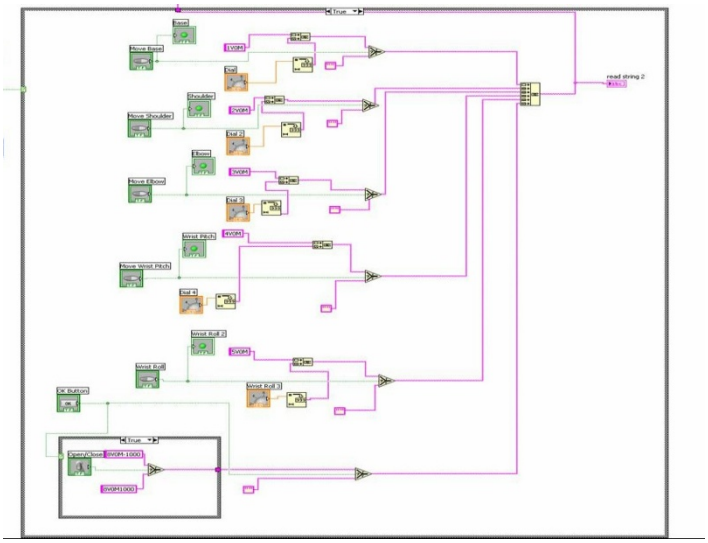

**Figure 13: The User interface section**

The second block of the program takes in data from the user interface and outputs the appropriate command. The following example shows the basic structure of how each motor is controlled.

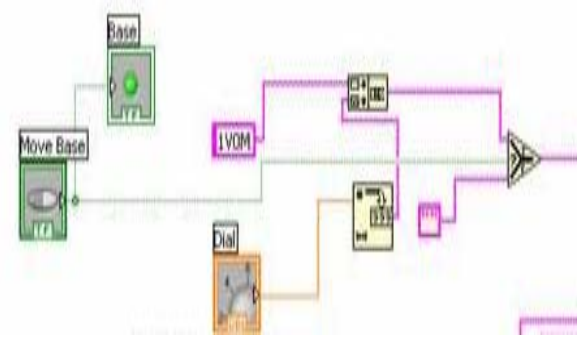


**Figure 14: The Example code**

Figure 14 shows the example code. Each piece contains the same basic elements: move button, a motor indicator, a dial, double to string function, two string constants, a concatenate string command, and a multiplexor. When the button is pressed, the motor indicator will light up, and the multiplexor will output the motor movement command. When the switch is not depressed, an empty string is outputted. Each motor movement command consists of two pieces of information. The first piece selects which motor to activate, and the second tells the controller how far the motor should move and in what direction. A positive number will rotate the motor one way, while a negative number will rotate the motor the other way. The format for the command is SMV0+/-XXXX. Where S represents the motor to be moved and XXXX represents the number which tells the controller how far to move.

The only exception is the gripper. The code for the gripper is slightly different.

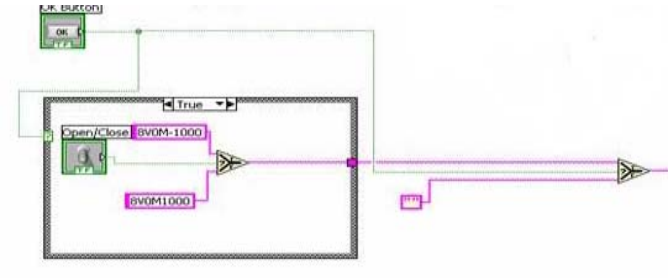


**Figure 15: The Gripper Code**

Figure 15 shows the gripper code. In this block, the motor commands are written in as string constants and a mulitplexor is used to determine which one to output. Then another multiplexor is used to output the command when the button is pressed.

## Results

We were able to successfully complete this project; however, some changes were made along the way. In order to complete this project we had to use a program called LabVIEW. This was a definitely a challenge because none of us had any previous experience with LabVIEW. After hours of research and building test programs, we became confident that we could build a project in LabVIEW to accomplish our goals.

The goal of this project was to develop a new, efficient, and easy -to- control program for the Scorbot. A lot of time was spent visualizing what would be the most effective way to layout the buttons and controls in LabVIEW. If the interface was not easy to understand, then we would be defeating the purpose of this project. With that in mind, we positioned everything in the best understandable way so that anyone could look at our interface and have an idea on how to control the Scorbot.

The most difficult part of this project was determining the command set to make the robot move. Since we did not create a new controller, we were forced to use the commands that the original software used. To determine the command set. We came up with a series of ideas on ways to determine the commands. In the end only two of the ideas actually helped us. Both of these ideas involved examining the data coming out of the Scorbase software. This procedure was how we were able to determine the commands to make our program work.

## Conclusion

This project in principle was a challenging one for us; we had to develop the knowledge of serial communication with RS-232, LabView and Scorbot robot.

We were able to successfully complete this project despite the many obstacles we faced along the way. When we started this project, we wanted to redesign the controlling unit as well as build a program in LabVIEW. But shortly after we began, we realized that we didn't have enough time to do both. We then decided to redesign the controller on just the software end. Some problems we faced were rewiring the serial cable to work with the controller, getting LabVIEW to transmit and receive commands from the Scorbot, and determining the simplest way to control the motors.

This project was a success. Throughout this semester, we learned a great deal about motor control, robotics, serial communication, and LabVIEW. We began this project with no experience in LabVIEW and now we are quite proficient at it. This project exposed us to many areas of knowledge we had no experience in such as instrumentation, robotics, and serial communication.

### References

1. Kurt Hudson, Virginia Technologies, "Labview-Controlled Robot Climbs And Inspects Highway Lighting Tower", available at http://www.optoiq.com/index/machine-vision-imaging-processing/display/vsd-articles-tools-template/_saveArticle/articles/vision-systems-design/volume-5/issue-10/technology-trends/industrial/robot-climbs-and-inspects-highway-lighting-towers.html (Accessed 4/2/07)

2. Sándor J. Tóth, "New PC And Labview Based Robot Control System" Periodica Polytechnica Ser. Mech. Eng. Vol. 43, (1999) 179–188
3. Alexei Sokolov, Robot Motion Realisation Using Labview
4. Periodica Polytechnica Ser. Mech. Eng. Vol. 43,(1999) No. 2, Pp. 131–145
5. Rob Brenner, Web Interfaced Robot Arm, ftp://iet.pct.edu/senior%20projects/web%20interfaced%20robot%20arm/website/proposal.htm (Accessed 4/2/07)
6. http://forums.ni.com/ni/board/message?board.id=170&message.id=227734&query.id=106352#m227734 (Accessed 4/2/07)