University of Washington

Computer Science and Engineering

Winter 2007

CSE 490 I: Design in Neurobotics

Lab1 Date: 1/9/2007 or 1/11/2007

Write-up Due: 1/16/2007 or 1/18/2007 (10:30am)

Goals

1. Learn to use the dataglove and the robotic arm using the software provided by the manufacturers.
2. Learn to read data from the dataglove using C/C++ and manipulate the data using MATLAB.
3. Learn to command the robot to move multiple joints simultaneously using C/C++.

Assignment:

There are four separate parts in this assignment. You need to work on 1 before 3, and 2 before 4, but otherwise work in any order that suits your group. We suggest that you spend the first few minutes reading through the entire document, as it will allow you to execute the entire lab faster (and purposefully). Remember that the lab session is two hours. Manage your time wisely.

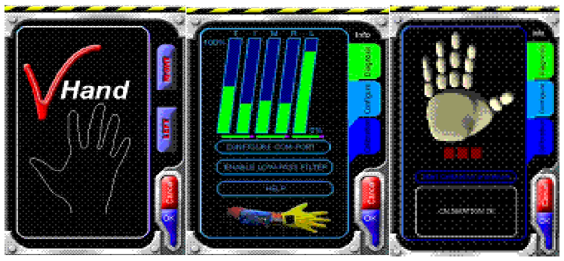
All the interface code are available on the course web page.

1. Learn to use the dataglove, DG5-VHand. There are five bending sensors, one in each finger. When the fingers of the glove are curled, the sensors change their resistive values and indicate how much the fingers are bent.

Question 1 and 2 below are related to this task.

(a) Turn on the glove. Launch the software, "VHand Setup" in the start menu folder "VHand"(Flag TA if you can't find it). Choose your glove version, whether it's left or right (you should choose "right" at first and later in the course you can decide which hand you may want to use for your project); choose from the configuration panel the COM Port which the glove is connected. Without wearing the glove, move the glove around so that you understand how the sensors work.

REMEMBER: the glove has to be turned on BEFORE to start the software.



(b) Put the glove on your right hand and go through the calibration procedure (for both lab partners). Click on the Calibrate label and start the guided calibration procedure. Follow the displayed instructions. (The enter button on the control box is the middle one.)

(c) Move the fingers around and learn about this system until you are comfortable.

2. The yellow robot on your lab bench is called Lynxmotion RIOS SSC-32. There are five controllable joints: base, shoulder, elbow, wrist and gripper. REMEMBER: make sure not to exceed the physically joint limits. Doing so can break the robot. This is especially important with the gripper servo, as it is small and strong enough to ruin its own gears. To avoid this, don't try to lift objects heavier than what is provided and make sure to grip with just enough force to keep the object. Also be careful of the cables on the arm.

Questions 3 & 4 are related to this task.

(a) Launch the software and play with the demo programs:

i. Open program "Lynxmotion RIOS SSC-32", you will be welcomed by this screen:

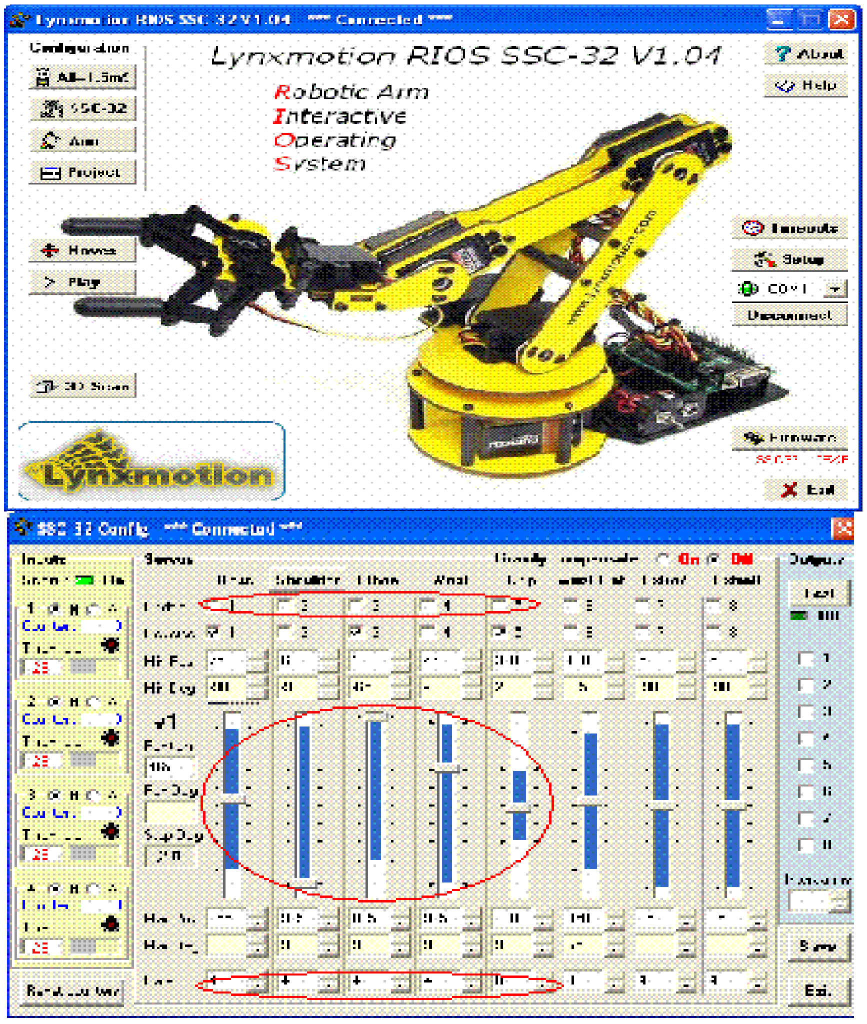
ii. Turn on the robotic arm, wait for the system to recognize the card and run RIOS. If you see a message saying "Can't find SSC-32 Card ..." Click "Yes."

iii. The robotic arm is connected when the "Connect" button (the last one in the middle of the right column) changes its label to "Disconnect." If not, try different COM port numbers. When you find the right port number, the card will auto-connected.

iv. Click on "All=1.5ms" button, then "Test", then "Yes", to set the robot in neural position.

v. Click on "SSC-32" button to open the configuration panel. Produce smooth joint movements using the five sliders on the left. Note that you can move only one joint a time in this program. "Pos deg" tells the motion of joints. "Rate" controls the speed of motion. 1 is the slowest, the larger the faster, BUT **0** is the fastest. (Refer to page 4-5 in the manual --- The manual is available by clicking "Help.") Observe the range of movement for each joint and the time delay. Note the model we use don't support wrist rotation.

This is an important exploration as you will be using these parameters to program the robot in C/C++ later. Pay special attention to the reference of each joint's position degree. It may be different from joints to joints, from robots to robots.



(b) There is a screw and a cup on your lab bench. Design a series of motions to control the arm to pick up the screw from the desk and drop it into the cup. (Always make sure that the gripper is commanded gently.) Both partners should do this task individually. Make a note about what you found difficult to do this task. If time allows, accomplish this task with two approaches: 1) you can use the configuration panel to control one joint at a time; or 2) use "move" panel to design a sequence of movement. Refer page 7 & 8 in the manual for usage of "move" panel.

3. Write a C/C++ program to read and record data from the dataglove under two conditions: one condition is when you wiggle around your fingers freely for about 3 seconds and the other condition is when you pick up the screw and drop it into the cup (with the hand wearing the glove). Save the data in two text files, one for each condition. Read all instructions (a) - (d) before you start writing the code.

Questions 5 & 6 need the data you collected in this section.

(a) Each partner should produce both files, named with the following convention: YourLastnameWiggle.txt, YourLastnamePick.txt. The .txt file should contain a matrix with each row for a record (time) and each column for a sensor. (So your matrix should have 5 columns for all the fingers, and about 300 rows for a 3-second recording.) You can separate numbers in a row by space(s), tab, or commas.

(b) Save the text files on YOUR OWN server space so that you can access them after the class.

(c) During the wiggle condition, include movements where each of the finger reaches its limits.

(d) You'll need five files for your code: DG5GloveLib.dll, DG5GloveLib.lib, DG5GloveLib.h, Glove.cpp, and Glove.h. Include Glove.h in your C/C++ file(s). You don't need to call functions in the DLL directly, but remember to add DG5GloveLib.lib to the additional dependency of your VC++ project. Use class Glove as an interface to communicate with dataglove. It provides methods to open the connection, calibrate the data (which you should do before any reading), read the state of each sensor, and close the connection safely. The sampling frequency of dataglove is 100hz. You can use function "sleep(10)" between calls of reading.

4. Write a C/C++ program to operate the robotic joints. First, implement the motion series you've designed in 2 (b), to control individual joints separately to pick up the screw and drop it off in the cup. Then, speed up the process by controlling multiple joints simultaneously. Record the movement of the robot joints.

You need data collected in this assignment to answer questions 7 & 8.

(a) It's challenging to design the motions all at once. You can write an interactive program, so that you can input one motion command at a time, and adjust according to the movement of the arm. It is OK if you fail to accomplish the task of pick up-drop off, but be patient and try a few different approaches. Play with the orientation of the screw and placement of it. Flag a TA if you cannot succeed after a few iterations.

(b) Your code should record the motion of all five joints (in the order of base, shoulder, elbow, wrist and gripper) of the robot throughout the two processes (moving individual joint separately and moving them together). Save the trace in two separate text files (YourLastnameSingle.txt and YourLastnameMulti.txt), which should have the same format as you did for the dataglove. When you move multiple joints, you may need to be careful with the timing at which the joints move.

(c) The team can work together and come up with the same two files for both partners.

(d) You need two files as the interface: Lynx.cpp and Lynx.h. Include the latter in your own file. Methods of class Lynx provide you approaches to open/close the connection, read the position of each joint, control their final positions, and the speed of the movement.

Post Lab Questions: submit the answers, code, and plots for the following questions one week from the lab session (before 10:30am). Everything should be submitted electronically unless otherwise noted.

1. Are the sensors in the glove sensitive to individual joint bending? Is there a way for you to move your finger in such a way that you can guess the finger joint angles from the sensor reading?

2. Can you propose the minimum number of sensors that you need to implement as a glove to accurately describe the movement of your hand? Draw a hand (or get a picture of a hand from the web) and mark where all these sensors need to go. Describe some sensors that may not be obvious.

3. Write down all the parameters that are controllable on this robot. This list will be useful in the future when you are writing you C/C++ code.

4. When picking up the screw and putting it in the cup, write down what was easy and what was hard to execute.

5. Write a Matlab .m file that reads YourLastnameWiggle.txt and plot all five data in one graph (with five different colors). Add legend in the plot to link the colors to the corresponding fingers. Show units of the axes. Identify minimum and maximum points of all five lines and put a star on the curve (same color as the line itself) at those locations (so there will be 10 stars on the graph). Submit this graph (you can copy and paste this figure into your report.) Note which finger spanned the largest range and explain why that may be. Do the same for the finger with the smallest range. The .m code can be a script or a function.

6. Write another .m code that reads YourLastnamePick.txt. Plot individual finger data into separate subplots (have two in the first two rows and another in the third row). Use titles on each figure to indicate which graph belongs to which finger sensor. Mark axes with units. Identify the time when (i) the hand left the lap, (ii) the fingers touched the screw, and (iii) when the fingers released the screw into the cup. Observe the data, and find out what shape of the curve or change in the sensor reading that indicate to you about these three state changing points. Write a MATLAB code to detect these points. For example, if you decide by visual inspection that the index finger reaches its minimum value at the point of contact with the screw, then write a code that detects the location of the minimum value. Improve your detection algorithm as possible. Once these three points can be reliably detected, have the program to draw vertical lines on all five graphs that indicate these points. Submit a description of your detection algorithm, well-commented .m file(s) and the graphs (in one page with 5 subplots).

7. Write an .m code that takes the YourLastnameSingle.txt and plot the five joint trajectories in degrees over time. Mark the lines with color and caption similar to the way you did with the dataglove data. Comment on the pros and cons of this strategy. If you moved the joints in the reverse order from the way you did, do you think that it would have worked the same way (except for the gripper, of course)? Explain.

8. Write another .m code that takes YourLastnameMulti.txt and plot the five joint trajectories in degrees over time. Compare this graph with the one you produced for LastnameSingle.txt and discuss the pros and cons of this strategy (not just the fact that they are faster and moving at the same time).

9. Write down the two (or more) kinds of feedback that you would like to have from the robot to execute this task even better. What would they do? Can you find an appropriate (size, sensitivity, etc) sensors on the web and suggest how you may use that with this robot.

10. Submit all the well-commented codes (in C/C++ and MATLAB) you wrote.