CSE4421: Lab 1

Burton Ma

Thu 13 Jan, 2011

1 Preliminaries

This lab teaches you the basics of operating the CRS A150 robotic arm, an industrial robot manufactured in the early 1990s.

1.1 Safety

The A150 and A255 robotic arms can seriously injure you if you do not follow reasonable safety precautions:

- Always be ready to press the big red kill switch! Do not let the robot crash into someone, something, or the table!
- Keep the tables the robots are mounted on clear of extraneous objects (coats, books, binders, etc.).
- Do not put your head (especially if you have long hair) within the range of any of the robots.
- Never use a speed of higher than 30% of the maximum (in fact during the debugging use much less that 30%)
- Do not power on a robot unless you are logged in to the workstation connected to the robot. If you are not logged in, it is possible for someone else to remotely log in and control the robot!
- Do not remotely log in to the workstations connected to the robots.
- Avoid working in the lab alone.
- If a robot malfunctions, turn off the power and immediately inform the technical staff (`tech at cse.yorku.ca`) and the instructor (`burton at cse.yorku.ca`) in person during working hours and by email otherwise.

2 Instructions to Power the A150 Arm

1. Log in to the workstation connected to the arm.

2. Start a console; in the console start the minicom program by entering the command `minicom`.

3. Power on the robot by turning the clear power switch in the upper-right corner of the control panel clockwise. The switch should turn orange, the control panel should hum, and minicom should display some information.

4. Check the big red kill switch (kills power to the arm motors). If it is pressed in, unlock it by turning it counterclockwise.

5. Manually align the robot. There are 5 sets of markers that you must align by moving the robot by hand (there is no power to the arm motors yet). The robot will be limp, so you will need to support the links of the robot by hand. All 5 sets of markers must remain aligned before you proceed to the next step.

6. Have someone else press the arm power button (located beside the big red kill switch). The button should turn orange and the robot should support itself. There is now power to the robot motors.

7. Tell the robot to home itself by entering the command `HO` in minicom. The message `LOCATED IN ITS HOME BOUNDS?` will appear. Type `y` followed by enter. The system shell has the annoying feature of auto-completing typed commands; the actual command name is `HOME`.

8. The robot will now try to set each of its 5 axes into a pre-calibrated home position. A message will appear for each axis in minicom; the message `TEST PASSED` should appear for each axis (even if you did a poor job in the previous step!); make sure the robot is in the correct home position.

9. If the robot is not correctly homed, support the robot by hand and press the kill switch. Repeat steps 3–7 to re-home the robot.

10. Once the robot is correctly homed it is ready for use. Enter the command `READ` in minicom and quit minicom using `Ctrl-A X`.

3 Instructions to Turn Off the A150 Arm

1. Support the robot by hand and turn the main power switch to the off position. Gently allow the arm to come to rest on the foam.

2. Log out of the workstation.

4 Robot Geometry

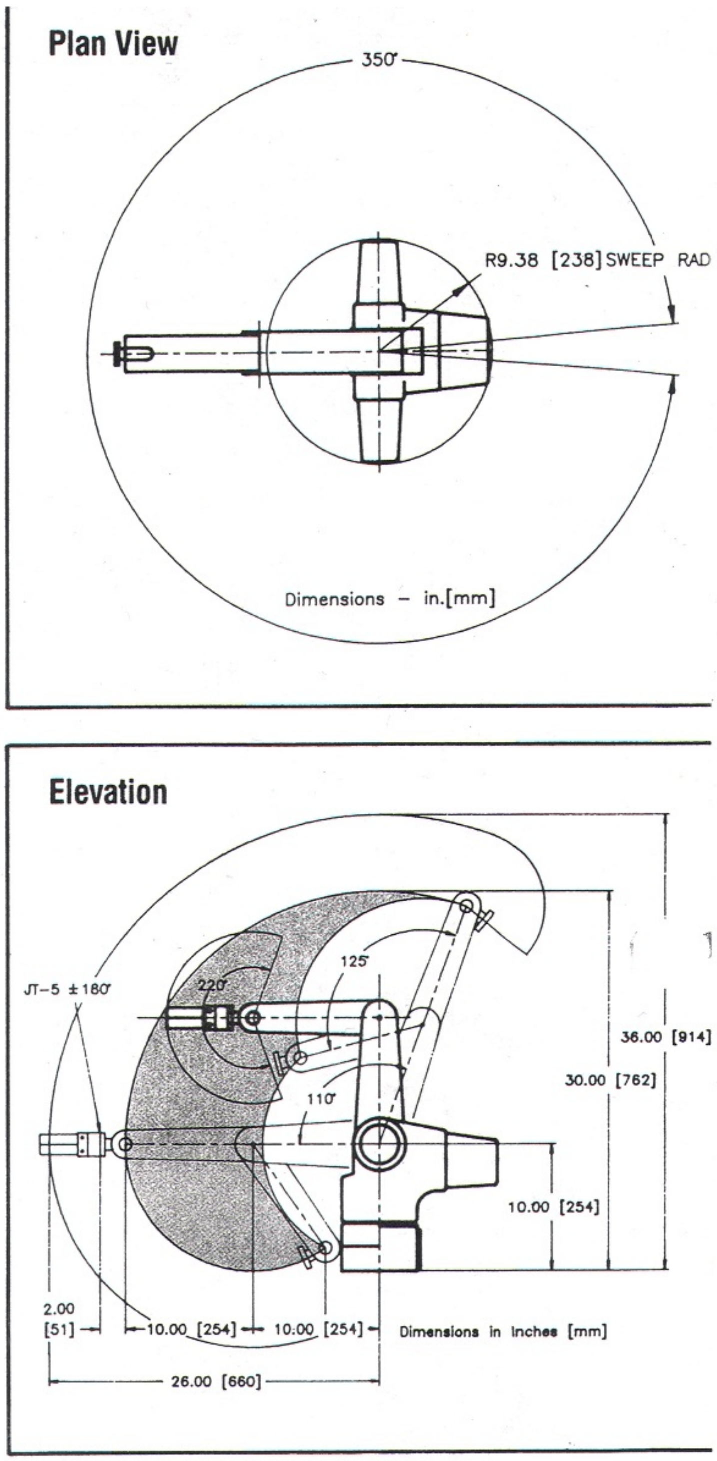

Figure 1: Dimensions of the A150 robotic arm *in inches and [mm]*.

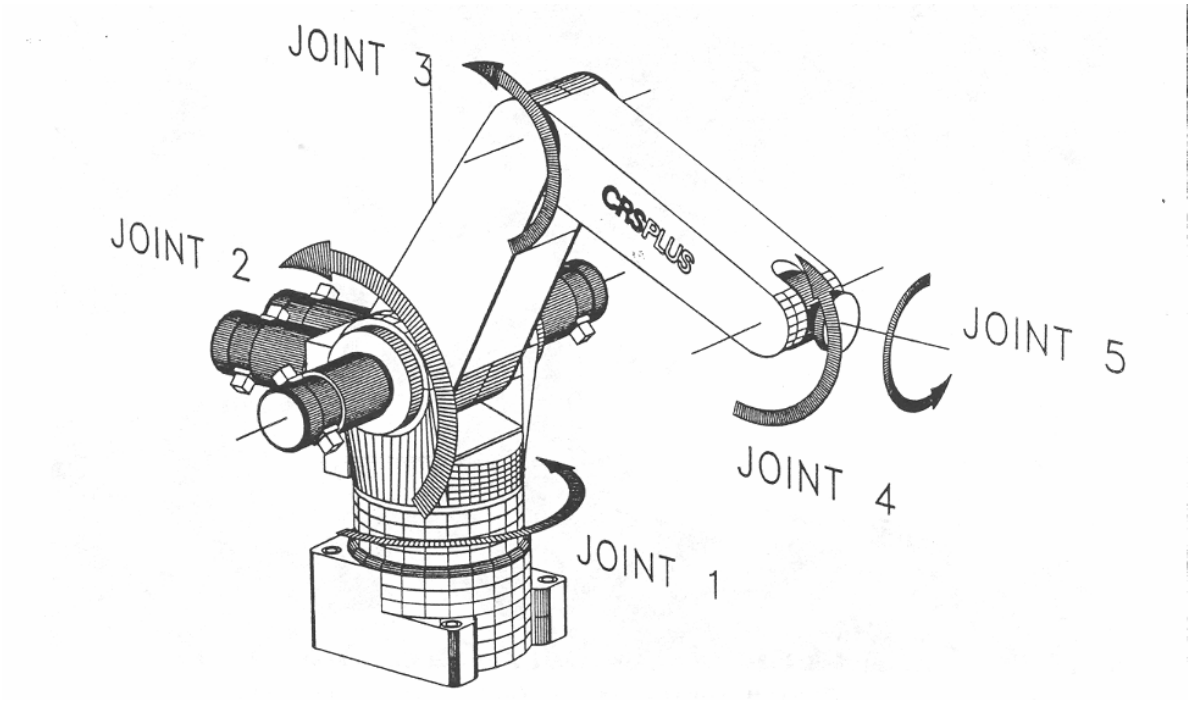

Figure 2: The five revolute joints of the A150 robotic arm.

Note that the Java simulator and controller classes refer to the joints as:

Joint	Java parameter name
1	waist
2	shoulder
3	elbow
4	wrist
5	twist

5 Source Code

Dr. Michael Jenkin has kindly provided Java software that simulates and controls the A150 arm. The simulator should run on any computer with a recent JDK installed. The controller communicates to the robot over the serial port; thus, it will only run correctly on the workstation to which the A150 is attached. A zip file containing the source code can be found at:

```
/cs/dept/www/course/4421/src/a150java.zip
```

The zip file can also be found under the *Source Code* section of the course web site.

There is some documentation in HTML format that can be accessed through the file `index.html`.

5.1 Using the Simulator Code

The Java class `ArmSimulator` creates a virtual A150 robotic arm that can be controlled using the same methods that you would use to control the physical robot. This is *very* useful for prototyping and debugging your robot programs before trying them on the actual arm.

The class `DemoSimulator` is a small demonstration program that shows how to use the class `ArmSimulator`.

5.2 Using the Controller Code

The controller code relies on a shared library file (`libserialport.so`). You should add the following line to your `.cshrc` configuration file:

`setenv LD_LIBRARY_PATH` *directory with libserialport.so*

Alternatively, you can run your Java programs by specifying the path to the library:

`java -Djava.library.path=` *directory with libserialport.so*

The Java class `ArmController` is a class that can be used to control the A150 robotic arm. If you examine the API of the class you will see that the A150 arm is controlled by setting the angles of the five revolute joints (see the methods `ma`, `madeg`, `mi`, and `mideg`). Notice that the method `move`, which moves the center of the grippers to a specified point, is not implemented.

The class `DemoController` is a small demonstration program that shows how to use the class `ArmController`.

6 The Task

First, carefully read this document. Next, study the APIs of the simulator classes and run the `DemoSimulator` program. Create a new simulator demonstration that moves the simulated arm in the following manner (starting from the home position):

1. Opens the gripper as wide as possible (2 inches).
2. Points the arm approximately straight up.
3. Rotates the wrist (joint 4) through its full range of motion (± 90 degrees) and back to 0 degrees.

4. Rotates the waist (joint 1) 90 degrees clockwise (when looking down on the robot).
5. Rotates the elbow (joint 3) so that the end of the arm is pointing through the glass into CSEB1006.
6. Rotates the shoulder (joint 2) so that arm is pointing down towards the table at 45 degrees.
7. Rotates the twist (joint 5) 90 degrees.
8. Closes the gripper.
9. Rotates the shoulder (joint 2) so that the end of the arm is pointing through the glass into CSEB1006.
10. Rotates the waist so that the end of the arm is pointing in the opposite direction from Step 9.
11. Rotates the shoulder (joint 2) so that arm is pointing down towards the table at 45 degrees.
12. Opens the gripper as wide as possible (2 inches).
13. Return the robot to the home position.

Show me your simulation once you have it working.

Now create a controller demonstration program that reproduces your simulation. *Remember to set the speed of the robot to a value around 10* (the maximum speed is 50). When you are ready, test your program using the actual robot.

7 Extending the Controller

The controller implements a small subset of the commands that can be sent to the A150. The full set of commands supported by the A150 are documented in the manual for the A150 arm (paper version only, sorry). The best way to learn what the commands do is to try them out using the instructions that follow.

After you use Minicom to connect to the A150 arm, you can send commands to the arm using the RAPL2 language; the language is fully documented in the second half of the A150 binder. You should scan over the documentation before proceeding with this part of the lab.

Power up the A150 arm as you learned in Lab 1, but do not exit Minicom. There are numerous commands that you might find useful when doing your project. Here is a partial list (I recommend you learn how to use them in this order):

- LIMP and NOLIMP
- CLOSE, OPEN, and GRIP (do not use GRIP to hold onto an object)
- W0 and W2 (what distance units are being used?)
- JOG
- HERE
- APPRO and DEPART
- CPATH

It should be possible to add these commands to the Java controller code (although I have not tried to do so myself). Note that any command that causes the arm to move will require additions and/or modifications to the following source code files:

- ArmController.java (you need to add the interface to the command you want to invoke here)
- CRSPlusA150.java (responsible for storing the joint angles of the robot)
- Armtty.java (responsible for sending commands over the serial connection)

Commands that do not cause the robot to move probably only need to modify ArmController.java and Armtty.java.

The Matlab interfaces for both robotic arms work in a different fashion; you will use these interfaces in Lab 2. They do not store the state of the robot; instead, they query the robot for its state.