

**University of Pennsylvania**  
**Electrical & Systems Engineering Undergraduate Laboratories**

**ESE 112: Introduction to Electrical & Systems Engineering**

**Lab 4: EduBot Dance**

**Objective**

Provide students with the background to design and implement a library of functions to perform a series of complex operations with the EduBot. Students will choose music and then choreograph a dance with the EduBot Robotic platform.

**Background**

Complex locomotion is an active area of research in robotics. To date, many legged robots can only achieve simple walking motion and mostly produced behaviors that are rigid and unnatural. Human shaped robots (humanoids) are designed with more sophisticated joints allowing for motions that are more natural. For a good example of a bipedal dancing robot, see [http://www.plyojump.com/movies/qrio/qrio\\_fandance.wmv](http://www.plyojump.com/movies/qrio/qrio_fandance.wmv) One of the challenges with complex locomotion is chaining together dynamic movements with known outcomes maintaining stability.

**EduBot**

The EduBot is a hexapedal (6-legged) robot that has recently been developed with walking behaviors modeled after a cockroach. This robot has proven to be stable and versatile in various terrains and can probably achieve many behaviors such as climbing stairs, jumping, and jogging that have been implemented on its predecessor (RHex).

EduBot walks by alternating three legs at a time in two different cycles. This motion involves moving two legs on one side of the robot and one leg on the other (ie the front and back legs on the left moves together with the right middle leg). This will alternate on each side, pushing the robot to forward and at the same time forming 3 triangular contact points on the ground keeping the robot stable. The legs attached to the hips and are numbered or indexed according to figure 1 below.

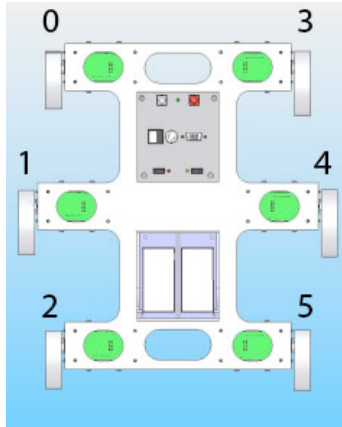


Figure 1 – EduBot Hip/Leg Numbering Convention (front of robot facing up)

## Coding

EduBot code base is complex and constantly evolving and will take months to be able to understand. The code needed for this lab is relatively simple and involves several functions that initialize the robot. Firstly, the project directory needs to be extracted. The archived zipped template for the dance lab project is located at “~/ESE112/LABS/DanceLabTemplate.tar.gz2” . Assuming you are in your home directory (/home/ese112/), copy the template file into your home directory and extract it by running the following command.

```
# cp ~/ESE112/LABS/DanceLabTemplate.tar.gz2 ~/
# tar -xjvf DanceLabTemplate.tar.gz2
```

A directory (or folder) called “DanceLabTemplate” is now extracted into the user’s home directory. The directory can be renamed to “myDanceLab” (or anything else) by moving the directory to a different name. For example:

```
# mv DanceLabTemplate myDanceLab
```

This directory will be referred to as the base directory of your project. In this directory, you should only write your code in the “OperatorCode/DanceLab.java” file. In the “dance” method in the “DanceLab.java” file your code where it says <write your code here> using a text editor. Of course, you should create methods of dance subroutine appropriately outside of this function. Your code should Do NOT change anything else in this file. The edubotGUI class implements a main function that calls the dance() function.

```
public static void dance() {
    edubotGUI.calibrate();
    edubotGUI.postureMode();
    edubotGUI.danceMode();
    <write your code here>
    edubotGUI.start();
    edubotGUI.finish();
}
```

Your code will consist of control loops, logical statements and function calls to `edubotGUI.moveLeg`. The function `moveLeg` takes 4 parameters:

```
edubotGUI.moveLeg(legnum, direction, angle, time)
```

The function parameters are described in the comments in the beginning of the `DanceLab.java` file. The `moveLeg` function can be piece together to form a dance move. Static dance move functions should be created and then called one after another to generate a dance. The dance time should add up to approximately 90-120 seconds and not legs should exceed 300 moves.

Keep in mind that a time graph of each leg position is generated from the `moveLeg` function calls and sent to the robot. Accordingly, when you move one leg and want the others to be stationary at any given time, the `moveLeg` function need to be called for all legs. Attention should be paid to make sure that the movements coded produce the expected behavior and does not damage the robot in addition to maintaining stability. The dance routine should be consistent and reproducible. There should be a very distinct hierarchy of code, where at the top most level, each function that is executed will execute a number of sub functions that will operate the robot on a lower level. To prevent unnecessary debugging effort, come up with appropriate names for your functions and comment your code well.

After a significant amount of coding is done, it is time to test your code. Compilation should be done in the base directory of your project by running the make command.

```
# make
```

Similar to the last lab where we ran the EduBot Supervisor locally on the computer to do simulations and remotely on the robot when we want to use the robot, we can do the same with the dance lab. However, a joystick needs to be plugged into the computer in order for it to work. Since, the robot code is written in C/C++, there is a JAVA wrapper around the GUI that allows JAVA code to run almost seamlessly. This JAVA to C/C++ interface virtually presses the buttons on the joystick to control the GUI just like how the user would click on the buttons with the mouse.

To run your dance code, assuming you have the Supervisor running and your code compiled without any errors, change directory into "Operator" (not `OperatorCode`) and run the following command. Remember to replace `<ip_address>` with `localhost` when running the simulation and to the robot's IP address when running it on the robot.

```
# java edubotGUI 3000 <ip_address>
```

The code is design to calibrate, get the robot in a standing position, and load the dance, you should not be clicking any buttons on the GUI until it gets to the "Dance Mode". The dance will begin when you click the "Dance Real" button.

## **Material**

Computer with Linux Operating System  
USB wireless adaptor  
Joystick  
EduBot

## **Prelab**

1. What is the command needed to set up the wireless USB adaptor and why do we need to use a wireless USB adaptor? (you may need to refer to Lab3 manual)
2. What is static stability? Why is it important in robotics? Give an example of why static stability needs to be considered in a robotic application.
3. What is dynamic stability? Why is it important in robotics? Give an example of why dynamic stability needs to be considered in a robotic application.
4. What is the maximum length a function can be for this project and how many seconds should your dance be?

## Lab instructions

1. From groups and run the sample code given in the project folder template in simulation and on the robot.
2. Choose an appropriate song to create a dance to. Your song should be rated PG13.
3. Follow the rules and guidelines laid out in this lab and come up with a few dance moves. Prepare a written explanation of this design (use of block diagrams, charts, written algorithms, pseudocode, etc) and present it to the course instructor before the start of the second lab session.
4. Test your software implementation in the simulation and ensure that it works as expected
5. Run your programmed dance on the robot and synchronize it with your music in preparation for a performance.

Note: This lab will be graded on your code design and development, sophistication and complexity of the dance movement and a small portion for the entertainment value of your dance.

## Questions

1. Describe your final software design structure. Include diagrams, algorithms, etc. to clearly convey how your code is structured. Why did you chose this structure and what are the advantages it provides?
2. Pick a move from your dance that had a relatively high risk of being unstable or one that was difficult to make stable. Describe the move (diagrams may be helpful) and discuss steps taken to make sure that move would not fail during the dance. How did your group assure stability within moves and tweak unsuccessful moves to make them more stable?
3. Each dance is made from distinct "moves". How did your group decide on a sequence of dance moves? Did the robot move as you expected based on the commands you gave it? If so, how did you achieve this? If not, why and at what points would the robot fail?
4. Describe your contribution to your group. As a percentage, how much of the work did you do? How much did everyone else do? Did you group work well together?
5. Email your code (DanceLab.java) to palsetia@seas.upenn.edu (one email per group will suffice), just write all your names at the top of the file (in comments)). Be sure to comment your code liberally so that the reader can easily understand the purpose of your code. Rename your file to should follow the following convention:

ese112\_danceLab\_XX\_XX\_XX.java

where XX is the first and last initial of a group member.