Lynxmotion Manipulator Notes I.

Below are some notes associated to the manipulator related to the Matlab code that exists for it. On t-square, there are two pieces of Matlab code in a zip-file, `lynx6.m` and `lynx6_setup.m`. Looking at the `lynx6_setup` code, you will see that it invokes the `lynx6.m` code. It is a partial m-file (`lynx6.m`), which you will fill in as the course progresses. Both files should be placed in the same directory that contains the other code you've been writing (where `logSE2` lives, for example).

The **lynx6** m-file is as example of what I call an interface object. It gets set up using a function call and then returns a handle. The handle points to functions that can be called; these functions are how you will interface the object. Looking at both the source code and the setup code, you should be able to interpret what I am about to write.

1 Manipulator Setup Code

The manipulator setup function is called `lynx6_setup.m` and is what you should invoke to first get acquainted with the manipulator.

Starting. This function pops open a GUI that has sliders bars to control the manipulator servo motors. The servo motors are sent numerical commands that range in $[500, 2500]$, where 500 specifies one extreme of rotation and 2500 represents the other extreme. The servo should be centered around 1500. Hence the setup function starts off with all settings at 1500. The manipulator should be plugged, powered-up, and connected to the computer via the serial cable prior to running the setup function. This is because the manipulator will be repeatedly sent the command to go to 1500 for all servos in order to activate the servo motors. If the servos do not activate, then hit the **Goto** button repeatedly until it does so (if this still fails, come see me).

There are some boxes to the right that specify the current servo command, plus the min and max values that can be commanded. The numerical limits are 500 and 2500, but some may be set lower to protect the servos. If you hit the max and you think that the servo can go more, then adjust the values and move into these new limits slowly. The min and max are editable text boxes.

It is critical that those who have the new manipulators made mostly out of aluminum remove the gripper servo motor from the gripper and have it sitting off to the side prior to invoking the setup function (do NOT lose the screw). Please still have it be connected to the servo controller board, just not engaged in the gripper. If the servo is in the gripper when it starts up and the servo was improperly installed, then activation could destroy the servo. Once the servo is activated, you can put it back into the gripper making sure that the gripper itself is at the half-way position when the servo motor is installed.

The Setup Function: Sliders. Once the manipulator is powered-up and the servos are (safely) activated, then you can slide the bars around to test out the motion limits. You must hit **Goto** every time you change the slider, if you want the manipulator to actually move. Enabling the **Continuous** button will continuously update the manipulator after changing a slider bar.

The Setup Function: Limits. When you adjust the slider bars to identify the limits, please be careful. The servos are dumb things that do what they are asked to do, even if it is incorrect or could break the manipulator. You will know you have done something wrong when a servo starting to make a nasty grinding sound. If you are close to doing something bad, the servo will start to whine louder, with the whine resembling a grinding noise as you get closer to the badness.

Figure out what the servo command values are the get close to the limits, but do not hit them. When doing this, please move the slider bar little by little until you get a feel for the appropriate limits. Also make sure that the manipulator can move freely without hitting anything during this procedure. The associated angles, as measured by a protractor relative to the zero configuration, are the joint angle limits for your manipulator. Each manipulator has different limits based on how it was put together.

2 Manipulator Interface Code: Calibration

The incomplete file `lynx6.m` is the main interface file for the manipulator. It is what you will be modifying to enhance the functionality of the manipulator as the course progresses. Prior to running the code, you will need to use the setup program to identify the operational limits and calibrate the manipulator. Calibration involves joint angle operational limits (in servo command units and degrees) plus link length measurement. Once these values have been found, the appropriate variables in `lynx6.m` will need to be modified to reflect what you've found.

Below are some of the variables you'll need to modify. If you have any questions, feel free to ask. Again, please be careful when changing and modifying the values. Be prepared to turn the manipulator off if things go funny.

alphaOrient. Changes the direction of motion in order to agree with the right-hand rule. Increasing values of joint angle should move in right-hand rule direction based on axis of rotation. If your manipulator is following a left-hand rule, then the sign here should be flipped.

alphaLims. The joint angle limits you've discovered, or have determined to be the most appropriate for your manipulator. Note that for the last joint, which is the gripper, the units will be in inches (the gripper open width).

musecLims. The actual servo signals that specify your limits. You may not want to include the full range, but enough to get angles that make sense. You know like a range of $[-90°, 90°]$ or $[-45°, 135°]$ and the like.

linklen. The link lengths that you measured. This will be used to program the forward kinematics as part of a future homework.

alphaHome. Your home position as a joint configuration. I would like for your home position to be the straigh-out position that we have been working with.

alphaSleep. Your sleep position as a joint configuration. This should either make the manipulator resemble the "C" shape, or have it point backwards so that the gripper rests on the back section of the base (sort of like a rainbow shape).

Basically, when the power is shut-off the servos deactivate and the manipulator arm may fall. Putting it into the sleep position allows for it to gently fall into a safe position, or moves it to a configuration where the manipulator won't really move much once powered off.

3 Manipulator Interface Code: Execution

This section further discusses the interface code, beyond the calibration variables. The code described here should not be run until AFTER you have configured and calibrated the manipulator and the `lynx6.m` m-file.

Invocation and Basic Usage. The main invocation is:

```
lh = lynx6();
```

which returns a handle. An optional setup parameter can be used to specify the overall configuration of the manipulator. If this is not given, then default values are used. The next thing one should do is to tell the manipulator to go to its home position until it finally does so (repeated attempts may be necessary). After that, commands will function as intended. Prior to that, the manipulator circuit board is not initialized and may not behave as commanded.

Going to home is done by invoking,

```
lh.gotoHome();
```

After this, the manipulator can be interfaced without problems, hopefully. You can determine what this home position is by modifying the appropriate internal variable.

Access to the manipulator functions are through the handle that is returned. For example to directly send a low-level PWM command to the manipulator, you would do:

```
lh.setServos( pwmvector, duration)
```

where **pwmvector** is a vector of the PWM command to send and **duration** is the desired duration of the command. The PWM command is given in servo command units $[500, 2500]$ whose limits respect the limits you've identified from the calibration step. If the internal variables of the interface object are setup correctly, you should be able to avoid sending direct servo commands and instead send joint angle commands. You can command the joints only, or the joints and the gripper,

```
lh.setArm( jointvector, duration)
```

The function will determine if you intended to send only joint angles or if you'd also like to command the gripper position.

Please note that when I say something is a vector, that means it should be a column vector. The code will ignore any argument that is not a column vector.

Termination. Because the system opens up a serial port for communication, you will have to properly shut down the interface object. Also, the manipulator should be commanded to the sleep position. The sleep position should be a joint configuration that makes sense for storing and moving the manipulator about; one that does not lead to it flopping about so much. Termination would then go as follows:

```
lh.gotoSleep();
lh.shutdown();
```

This should be enough to get started with the initial assignment.