Exercise 2: Whiteboard Coordinate System

Find starting position

To compute the starting position in x-y coorindates, we'll use measured values of *Base*, *L1*, and *L2*. First, measure the *Base* distance with a tape measure, ruler, or meter stick. Be sure to record the value in meters so that units remain consistent. Record this value in the `Base` variable, and save it to a file so you can use it for future exercises.

```
Base = 0.56; %Change this to the Base distance for your robot (meters)
save RobotGeometry.mat Base
```

Use your ruler/meterstick/tape measure to measure the initial *L1* and *L2* distances, which are the distances between the motors and the pulleys. Since these distances may change every time you use the robot, use an input dialog box to ask the user to specify the new values of these measurements. This will make updating the values easier when you run the code again in the future.

```
str = inputdlg({'L1 (m)','L2 (m)'},'Enter initial string lengths.',[1 50]);
L1_i = str2double(str{1}); %meters
L2_i = str2double(str{2}); %meters
```

Compute the initial Z1 and Z2 values using the measured L1 and L2 values and the known L_arm distance, which is 11 cm.

```
L_arm = 0.075;       %meters
Z1_i = L1_i + L_arm; %meters
Z2_i = L2_i + L_arm; %meters
```

Use the Pythagorean Theorem to calculate the robot's x-y position based on the measured values of Base, Z1, and Z2.

```
x = (Base^2 + Z1_i^2 - Z2_i^2)/(2*Base) %meters
```

```
x = 0.2800
```

```
y = sqrt(Z1_i^2-x^2) %meters
```

```
y = 0.1650
```

Connect to the hardware

Now we can move the robot around the whiteboard and compute the new x-y position based on the change in string length. To begin, create variables to connect to the various robot components as you have done previously.

```
a = arduino;
carrier = addon(a,'Arduino/MKRMotorCarrier');
s = servo(carrier,3);
```

```
mL = dcmotor(carrier,2);
mR = dcmotor(carrier,1);
eL = rotaryEncoder(carrier,2);
eR = rotaryEncoder(carrier,1);
resetCount(eL)
resetCount(eR)
```

Draw on the whiteboard

A MATLAB app has been provided for your to make it easy to specify motor voltages and move the robot around the whiteboard. Try it out. See if you can drive the robot to different parts of the board.

WARNING: Be careful not to move the motors too fast or you may not be able to control them well. Also be careful moving it toward the top of the whiteboard and near the pulleys. The motors can stall if there is not enough torque to move or if they have moved all the way to the pulley. Continuing to power a motor at stalled conditions can damage it.

```
load ServoPositions
SimplePlotterApp(s,mL,mR,LeftMarker,RightMarker)
```

Calculate new robot position

After moving the robot to a new position, we can compute its new position. The encoders give motor positions in counts. We can use the known relationships between counts, angles, and distances to convert this into linear distances and use the geometry of the robot to calculate the new position in x and y. First, read the position returned by each encoder.

```
counts1 = readCount(eL)
counts2 = readCount(eR)
```

We no longer need to be connected to the hardware for the rest of this exercise, so it's a good idea to clear the hardware variables. Note, the simple plotter app will stop working, so you should close it if it is still open.

```
clear a carrier s mL mR eL eR
```

Define some constants. The motor spec tells us the gear ratio is 100.37. The encoder spec tells us there are 12 counts per revolution of the motor shaft. Multiplying these numbers, we find that there are 1204.44 counts per revolution of the output shaft. We can also define the radius of the spool based on a known value.

```
countsPerRevolution = 1200;
countsPerRadian = countsPerRevolution/(2*pi);
r_spool = 0.0045; %meters
```

The encoder counts can be converted to angles using the `countsPerRadian` scaling factor.

```
% Convert counts1 to radians
angle1 = counts1/countsPerRadian %radians
% Convert connts2 to radians
```

```
angle2 = counts2/countsPerRadian %radians
```

Using the definithion of arclength, convert the rotated angle to a linear distance. This is the amount of string that has been let out or taken in around each spool.

```
dStringLength1 = r_spool*angle1 %meters
dStringLength2 = r_spool*angle2 %meters
```

On the robot, the string loops over the pulley and attaches back on the robot body. Because the string is doubled in this way, the resulting change in distance from the pulleys (Z1 and Z2) is equal to half the change in total string length. Compute the new Z1 and Z2 distances.

```
dZ1 = dStringLength1/2 %meters
dZ2 = dStringLength2/2 %meters
Z1 = Z1_i + dZ1 %meters
Z2 = Z2_i + dZ2 %meters
```

As before, use the Pythagorean Theorem to compute x and y from Base, Z1, and Z2.

```
x = (Base^2 + Z1^2 - Z2^2)/(2*Base) %meters
y = sqrt(Z1^2-x^2) %meters
```