Exercise 3: Move to Specified Points

Choose position to move to in x-y

Choose a position in x-y coorindates that you want to move the robot to. Remember that *x* is measured horizontally from the pulley at the top-left of the whiteboard and *y* is measured down from the line connecting the two pulleys. Choose a target position that the robot should be able to reach. Remember to define distances in meters.

Change the following values for x and y to your chosen values.

```
x = 0.19; %meters
y = 0.20; %meters
```

Convert target position from x-y to encoder counts

Previously, you created a function to convert encoder counts to an x-y position on the whiteboard. Now we want to do the conversion in the other direction. Start with a desired x-y position and figure out what the encoder counts should be at that position.

As you did in the previous task, measure the initial string lengths to determine the initial *Z* positions. Measure the distances from the left motor to the left pulley and from the right motor to the right pulley.

```
Z_i = initialPosition;
```

Define the costants for spool radius and counts per revolution.

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

Load the `Base` variable that you saved to a MAT-file in the previous exercise.

```
load RobotGeometry.mat Base
```

Use the Pythagorean Theorm to compute the *Z1* and *Z2* lengths of the target position. To simplify the code, put both values in the same variable, `Z`. This allows you to apply functions to both values at the same time.

```
Z(1) = sqrt(x^2 + y^2);
Z(2) = sqrt((Base-x)^2 + y^2);
```

Find the change in length of each hypotenuse *Z* from its calibrated position *Z_i*.

```
dZ = Z - Z_i;
```

Recall that the doubled string means that for any change in the length of *Z1* or *Z2*, the corresponding change in string length is double.

```
dStringLength = 2*dZ;
```

Convert the change in string length to an angle in radians using the definition of arclength.

```
phi = dStringLength/r_spool;
```

Use the known relationship between counts and angle to calculate the encoder counts at the target position.

```
counts = phi*countsPerRadian
```

Convert position to counts using a function

Follow the provided instructions for creating a MATLAB function `xyToCounts` that will perform the above steps. Run the function to confirm that you get the same resulting value for `counts`.

```
counts = xyToCounts([x y],Z_i,Base)
```

Connect to hardware

Connect to the Arduino and peripherals as you have done in each of the previous lessons.

```
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)
```

Load the servo position values saved in the first exercise. This file includes the variables `LeftMarker`, `RightMarker`, and `NoMarker`.

```
load ServoPositions.mat LeftMarker NoMarker
writePosition(s,NoMarker)
```

Move to target position

The function `moveToCounts` is provided for you. Call this function to move the robot to the desired position on the whiteboard.

```
moveToCounts(counts,mL,mR,eL,eR)
```

Move to new target positions

Repeat all the above steps for a new target position, using the functions you now have.

```
x = 0.15; %meters
y = 0.15; %meters
xy = [x y];
counts = xyToCounts(xy,Z_i,Base)
```

```
moveToCounts(counts,mL,mR,eL,eR)
```

Move to a sequence of positions

The `moveToCounts` function can also move the marker through a series of points. Create a set of x-y points to move the marker. Use the rectangle defined below or create your own set of points to draw.

```
x = [0.15 0.20 0.20 0.15 0.15]; %meters
y = [0.15 0.15 0.25 0.25 0.15]; %meters
xy = [x(:) y(:)];
counts = xyToCounts(xy,Z_i,Base)

% Raise the marker
writePosition(s,NoMarker)

% Move to the first point
moveToCounts(counts(1,:),mL,mR,eL,eR) %counts is 5x2. counts(1,:) is the first row of counts

% Lower the marker and move through a`ll points
writePosition(s,LeftMarker)
moveToCounts(counts,mL,mR,eL,eR)
writePosition(s,NoMarker)
```

Clear hardware variables when you are no longer using them.

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