Exercise 1: Introduction to the drawing robot

Connect to the hardware

Once you've set up the Arduino, you can connect to it from MATLAB. You can also connect to the motor carrier, which will be used to communicate with the other peripherals on the robot.

```
a = arduino;
carrier = motorCarrier(a);
```

Control the servo

The servo motor is used to raise and lower the whiteboard markers. It should be connected to the SERVO3 port on the carrier. Create a variable in MATLAB to control it.

```
s = servo(carrier,3);
```

Keep calling writePosition, changing the value by a small amount each time. Explore different values until you figure out which values to use for lowering the left marker, lowering the right marker, and raising both markers.

```
pos = 0.23; %Change this value and continue running this section
writePosition(s,pos)
```

Once you've identified the servo values for lowering the left and right markers, store them in variables, and save them to a MAT-file. You'll load these values in future exercises that require raising and lowering the markers.

```
LeftMarker = 0.89;  %Change this to the LeftMarker value on your robot
RightMarker = 0.34; %Change this to the RightMarker value on your robot
NoMarker = mean([LeftMarker RightMarker]);

save ServoPositionws.mat LeftMarker RightMarker NoMarker
```

Control the DC motors

Connect to the DC motors and the motor encoders. These work together to move the robot around the whiteboard. For now, you don't need to hang the robot up. You can leave it lying flat on a table.

```
mL = dcmotor(carrier,2);
mR = dcmotor(carrier,3);
eL = rotaryEncoder(carrier,2);
eR = rotaryEncoder(carrier,3);
```

Also, reset the count of the encoders to 0. Now any position will be measured relative to the current motor positions.

```
resetCount(eL)
resetCount(eR)
```

Next, try controlling the DC motors. The battery included in the kit has a rated voltage of 11.1 V, although the true value may be higher or lower depending on the charge level. You can use a voltmeter to check the exact voltage or use 11.1 as an approximate value.

```
Vmax = 11.1; %Battery voltage (Volts)
```

Choose a target voltage, `Vset`, that you want to supply to the motors.

```
Vset = 3; %Target voltage (Volts)
```

The variables mL and mR have a property called `DutyCycle` that controls how fast they run and in which direction. `DutyCycle` varies between -1 and 1. It creates a PWM signal with an equivalent voltage differential of -V to V across the motor terminals, where V is the supply voltage. Set the `DutyCycle` property based on your target supply voltage.

```
mL.DutyCycle = Vset/Vmax;
mR.DutyCycle = Vset/Vmax;
```

The motors won't run until you start them. Run the following code to drive the motors at the target voltage for about three seconds.

```
start(mL)
start(mR)
pause(1) % Wait 1 second
stop(mL)
stop(mR)
```

Read the encoders

Now that the motors have moved from their starting points, you can check the encoder values to see how far they've been displaced since you last reset the counts. In the next lesson, you'll learn how to translate this value into physical distance units.

```
count1 = readCount(eL)
count2 = readCount(eR)
```

Always clear hardware variables whenever you're done using them.

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