# Robots I - Week 8

Cruise Control

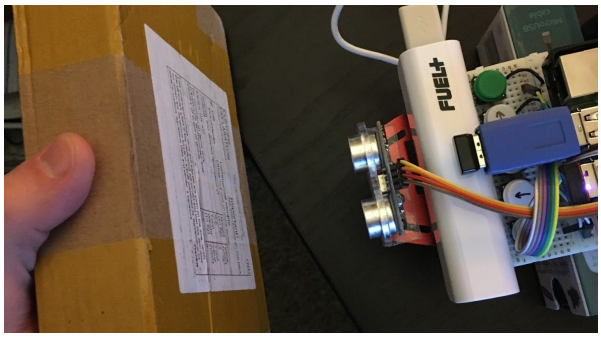
## Cruise Control

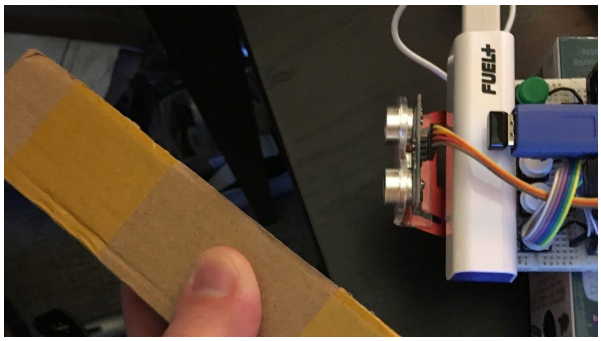
The purpose of this week is to create code for our robot that will travel the furthest in the shortest amount without getting stuck. If the robot senses that it is about to get stuck, it should autocorrect the direction and then proceed.

Our basic strategy is simple - go straight. If you get close to hitting a wall, make a correction and continue on. Since we do not have rangefinders on the left or right side of our robot, we will have to assume that a simple left or right turn will correct the problem.

## Measuring Angles

As your robot is moving the approaching angle of an object will make a difference. If the robot is directly in front of the object, the rangefinder will find the most accurate distance. However if the robot is approach at an angle, the rangefinder may not see the closest point.




You can see this first hand with the following code:

```
from time import *
from rrb3 import *

robot = RRB3(6,6)

while True:
   d = robot.get_distance()
   print d
   time.sleep(1)
```

How you respond to these distances should account for potential angles as objects may be much closer than your robot may think!

# Self Navigating

Here is very, very basic code that will allow your robot to navigate around obstacles. The code uses the fine grain motor API to adjust speed as it approaches a potential obstacle. You'll want to get this code implemented as is and then adjust to suit your needs.

```
import os
import sys
from time import *
from rrb3 import *

robot = RRB3(6,6)

# - compensate the wheels if one is moving faster
# - than the other
lcomp = 1
rcomp = 1

while True:
   d = robot.get_distance()
   print d
   if d > 75:
      robot.set_motors(1*lcomp, 0, 1*rcomp, 0)
   elif d > 25 and d <= 75:
      robot.set_motors(0.75*lcomp, 0, 0.75*rcomp, 0)
   elif d > 15 and d <= 25:
      robot.set_motors(0.6*lcomp, 0, 0.6*rcomp, 0)
   else:
      robot.set_motors(0.4*lcomp, 1, 0.4*rcomp, 0)

   time.sleep(0.5)
```

There are a few things to call out in our program. First, the lines:

```
lcomp = 1
rcomp = 1
```

can be used to tune the speed of the left and right motor. If, for instance, the left motor is faster than the right (so your robot turns a little to the left as it moves) you can adjust the speed of the left motor. So you could try:

```
lcomp = 0.90
rcomp = 1
```

Or some other number.

If there is nothing in plain sight (greater than 75 centimeters), the robot goes into full speed mode. If between 25 and 75 centimeters, it slows to 75% power, 15 and 25 it slows to 60%

power. If the rangefinder spots something within 15 centimeters, it stops and performs a slight left turn.

Now, note, these distances do not account for angles and this robot will crash into objects. Nonetheless, give it a spin and make your own adjustments.

# Contest

If time allows, our contest will be to see what robot can ride the longest without human intervention. It’s perfectly ok if it crashes into objects, so long as it doesn’t require a human to get it back on track.