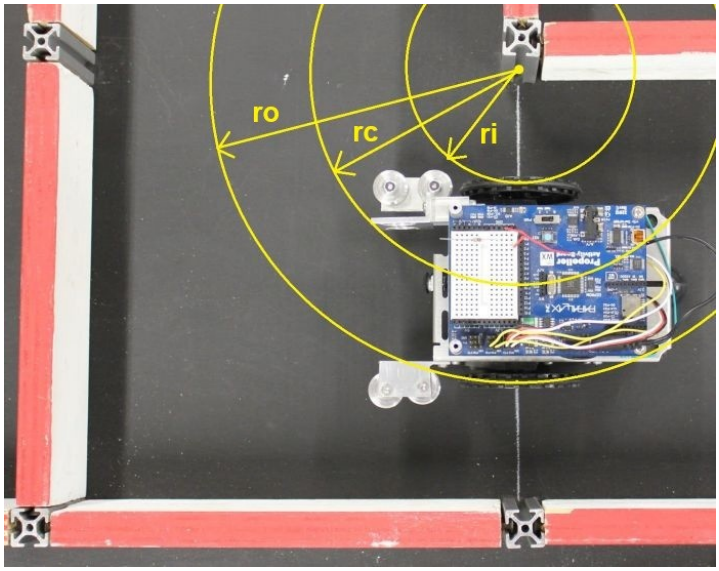
Now that you have successfully written some C language code for following the right wall, you will work on making the robot turn at a corner. It is apparent from Figure 1 that the robot must make several turns in order to make its way through the maze.

Figure 1: 2025 NRC Robot Maze - middle school

There are three types of turns the robot can execute: 1) zero-radius turn **2)** pivot turn **3)** arc turn. The zero radius turn is the sharpest turn and is accomplished by making one wheel turn backward while the other wheel turns forward, both at the same rotation speed. The pivot turn is the next sharpest type of turn, where one wheel does not spin at all while the other one spins forward. In this case the robot pivots around the stationary wheel. An arc turn is performed with both wheels moving forward with one wheel turning at a slower rate than the other. It is my opinion that the arc type of turn is best suited for this maze. Keep in mind that one of the objectives of this NRC contest is to make the robot go through the maze as quickly as possible. That means that you will want the robot to make its turns as quickly as possible. In the case of the zero radius turn, **one wheel must move backward** as the robot makes the turn, and does not make any forward progress during the turn. The pivot turn is not much better, making very little forward progress during the turn. But with the arc turn, the robot does make forward progress during the turn, which is why I believe it is the turn type of preference for this contest.

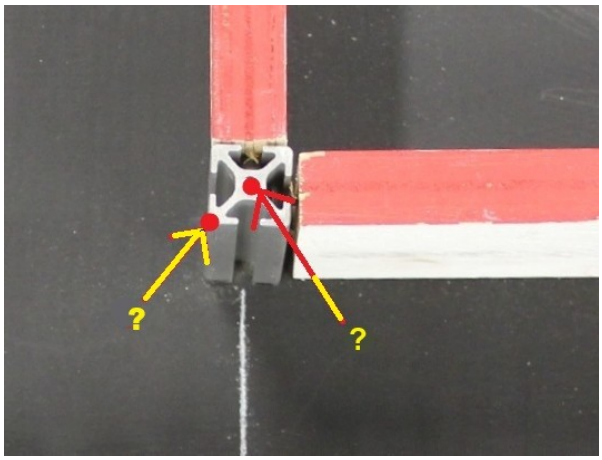
Making Arc Turns

When the robot wheels are spinning at different speeds, the robot travels a curved path. We can specify the shape of the curved path by using the mathematics of circles. The size of a circle is conveniently specified as its **radius** (r), *the distance from the center of the circle to the circumference*. In the case of the robot, we are interested in three radii, **ri,** the radius to the inner wheel, **rc**, the radius to the center of the robot and **ro**, the radius to the outer wheel. The distance between the robot wheels is about 10.2 cm. From this we can derive all three radii for a turn if we know only one radius.

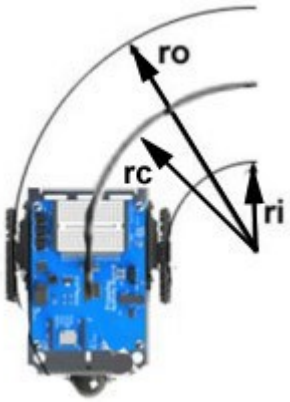


Where should the center of the circle be?

Two candidates for the circle center point are **1)** corner of the wall post or **2)** center of the wall post. I would argue that the center of the wall post is better. This allows for a larger radius turn where the speed of the inner wheel is slightly greater than with a shorter radius turn. In addition, the turn should then start earlier, at the center of the corner post (white line) rather than at the corner of the post.



Let us take the third turn in the maze as an example, a **90 degree turn to the right**. Suppose we want the robot centered in the lane at the beginning of the turn and to end up centered in the intersecting lane at the end of the turn. Then we can set rc as half the width of the lanes. The lanes are 25.4 cm wide (including half the width of each wall on the sides). Then rc = 25.4 / 2 = 12.7 cm. We need to calculate ro and ri:



ro = rc + ½ 10.2 cm = 17.8 cm
ri = rc – ½ 10.2 cm = 7.6 cm

Now calculate the circumference of the circle with the radius of the outer wheel, ro.

2 x ro x 3.14 = 112 cm

The outer wheel must travel 112 cm around the circle of radius 17.8 cm to complete one revolution (360 degrees). A 90 degree turn is one quarter of a full circle:

112 cm / 4 = 28 cm

The outer wheel must travel 28 cm to make a 90 degree turn with a radius of 17.8 cm. Now suppose we are using the drive speed block driving at speed of 64 ticks per second (20.8 cm per second) for the outer wheel (the left wheel). How much time will it take for the outer wheel to travel 28 cm?

28 cm / 20.8 cm/sec = 1.35 seconds (or 1350 milliseconds).

The pause block added after the drive speed block should have a value of 1350 for a 90 degree turn. We have one more calculation to make, which will determine the speed of the

inner wheel. We can't specify a turn unless we have speed values for both outer and inner wheels. The difference is a ratio of the two radii:

ri / ro = ratio of inner wheel
7.6 cm / 17.8 cm = 0.427

Multiply the ratio by speed of outer wheel to calculate speed of inner wheel:

0.427 x 64 ticks/sec = 27 ticks/sec

Below is code that will result in a 90 degree turn to the right with a radius of rc = 12.7 cm. In order for the robot to turn right, the left wheel must spin faster than the right wheel, as you can see in the code. The turn will take 1350 milliseconds to complete.

```
drive_speed(64, 27);     // left wheel 64, right wheel 27
pause(1350);             //pause for 1350 milliseconds
```

There is one detail I left out until now. The drive speed function includes an acceleration factor. That is, it speeds up over time to reach full speed and slows down at the end. Since the calculations I have provided for you don't account for acceleration, they are only an approximation of the degrees turned. In any case, unless you code for a slower acceleration than the default fast acceleration, the approximation method I provide will provide good values for your turns, subject perhaps to small changes in values to more closely tune a turn. A precise method to calculate a turn would require the math known as calculus and is not appropriate for this lesson.

Below you will find the code that will cause the robot to move forward for 2 seconds and then turn right 90 degrees. Copy this code and download it to your robot. Place the robot in an open space of a floor and test the program. Did the robot turn approximately 90 degrees? Alter the code so that the robot makes an 180 degree turn right. Then test it with the robot. Then alter the code so that the robot makes a 90 degree turn left. Test it with the robot.

```
#include "simpletools.h"
#include "abdrive360.h"

int main( )
   {
   drive_speed(64, 64); // drive forward
   pause(2000);
   drive_speed(64, 27); // turn right 90 degrees
   pause(1350);
   drive_speed(0, 0); // stop
   }
```

Now you know how to make the robot make a turn. In the next lesson you will work with the Ping sensors to determine WHEN to execute the turn.

File name: TurningCorner.pdf
online location: https://www.lafavre.us/robotics/TurningCorner.pdf
last update: 3/18/26