Wall-following function J. La Favre March 2026

The image to the right is the maze pattern for the 2025 NRC Robot Maze Contest, Middle School Division. You have already learned how to make the robot follow a wall using the right-facing Ping sensor. Depending on your code for solving the maze, you might need to add wall-following code multiple times. You can reduce the number of code lines in your maze program by placing the wall-following code inside a custom function. This lesson covers making such a function.

First include the necessary libraries:

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

Variables can be created inside a function. For example, if you create a variable inside the **main()** function, it is a local variable and can only be used inside the **main()** function. If you create a variable near the top of your code, before the main**()** function, it will be a **global variable**. Then the variable can be used anywhere in your code. For this lesson you need to use global variables.

Create the global variables:

```
int left = 0;          //store number of ticks traveled by left wheel
int right = 0;         //store number of ticks traveled by right wheel
int distR = 8;         //distance to wall right of robot
```

Before the main function you must declare (name) the custom function. Suppose we name the custom function **followWall ()**.

```
int followWall();      //declare a custom function named followWall()
```

Now add the main function

```
int main ()
{                      // open brace of main function
}                      // close brace of main function
```

Below you will find an example wall-following code. Your code might be different and I encourage you to substitute your code.

```
distR = ping_cm(16);  //make a ping measurement to right and store in distR
if (distR < 8)        //if robot is closer than 8 cm to right wall
  {
  drive_speed(58, 64);               //turn slightly left
  pause(20);
  drive_getTicks(&left, &right);     //store distances traveled left and right
  }
else if (distR > 8)         //if robot is farther than 8 cm to right wall
  {
  drive_speed(64, 58);      //turn slightly right
  pause(20);
  drive_getTicks(&left, &right);
  }
else                        //i.e., if robot is 8 cm from right wall
  {
  drive_speed(64, 64);      //drive straight forward
  pause(20);
  drive_getTicks(&left, &right);
  }
```

(continued on next page)

The code below creates the custom function named **followWall()**. It is placed AFTER the close brace of the main() function.

```
int followWall()           //custom function for following a wall
{                          // open brace of followWall()
distR = ping_cm(16);
if(distR < 8)
  {
  drive_speed(58, 64);     //turn slightly left
  pause(20);
  drive_getTicks(&left, &right);
  }
else if(distR > 8)
  {
  drive_speed(64, 58);     //turn slightly right
  pause(20);
  drive_getTicks(&left, &right);
  }
else
  {
  drive_speed(64, 64);
  pause(20);
  drive_getTicks(&left, &right);
  }
}                          // close brace of followWall()
```

Now all that is needed is to call the custom function within the main() function

```
int main()
   {                       // open brace main function
     while(left < 206)
       {                   // open brace while
       followWall();
       }                   // close brace while
   drive_speed (0,0);      // stop robot
   }                       // close brace main function
```

Notice that the above code includes a while loop which runs until the robot has traveled 206 ticks, the place where the first turn should be executed. Test your code in the maze. Your robot should stop at the corner of the first turn.

This is the last lesson in the series for coding the ActivityBot360 to solve the maze. I think you should now have enough knowledge to complete a code for making the robot solve the maze. There are many ways to code a solution. Your solution may rely only on distance sensing and turn counting for making turns. Or perhaps you will also use some odometry. Experiment with the code to find what works best.

Filename: WallFollowingFunction.pdf
online link: https://www.lafavre.us/robotics/WallFollowingFunction.pdf
last update: 3/20/26