

## Movement

# Speed and Direction **Motor Power**

*In this lesson, you will modify the existing program to make your robot move at a slower speed. This should result in more consistent movement.*

Moving at slower speeds can help your robot to be more consistent. All you need to do is alter the motor commands to turn the motors on with a power level lower than 100%.

1. Change the power levels in your motor[] commands to move at half speed.

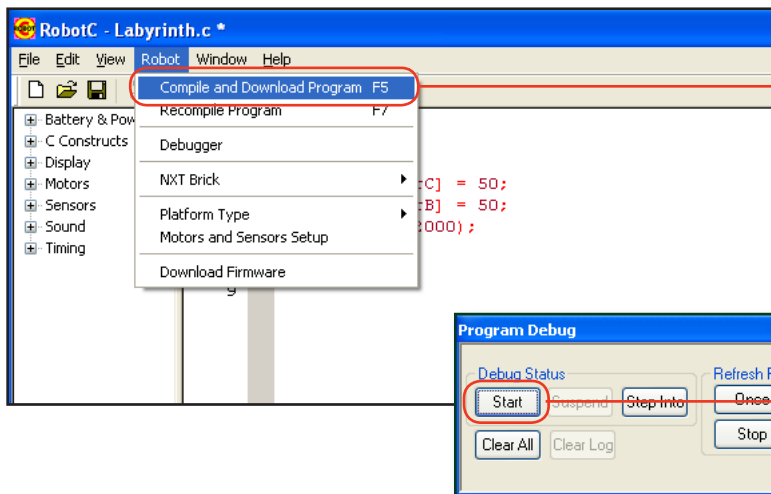
```

1 task main()
2 {
3
4     motor[motorC] = 50;
5     motor[motorB] = 50;
6     wait1Msec(2000);
7
8 }
```

### 1. Modify this code

Change the old 100 (100% power) to 50 (50% power) to make the robot move at half power. Do this for both motor commands.

2. Download and run your program. Note that downloading automatically saves your program.



### 2a. Compile and Download

Select Robot > Compile and Download Program to send your program to the robot.

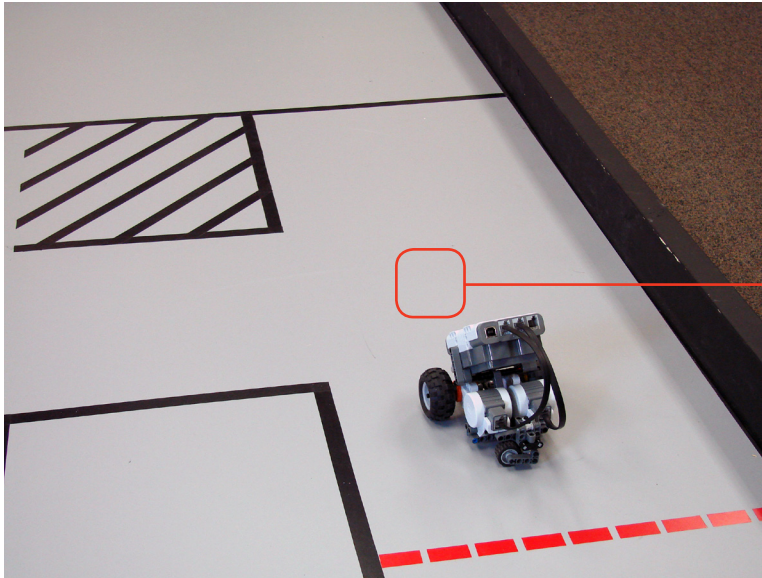
### 2b. Press Start

Press the Start button on the Program Debug menu that appears, to run the program.

## Movement

### Speed and Direction **Motor Power** (cont.)

**Checkpoint.** The numeric value assigned to each motor in the motor[] commands represents the % of power that the motors will run with. So far, we've changed them from full power to half. Since your robot is traveling slower, it will now need to travel longer to go the same distance.



#### **Distance changed**

Traveling for the same amount of time, but at a slower pace, causes the robot to stop short of its destination.

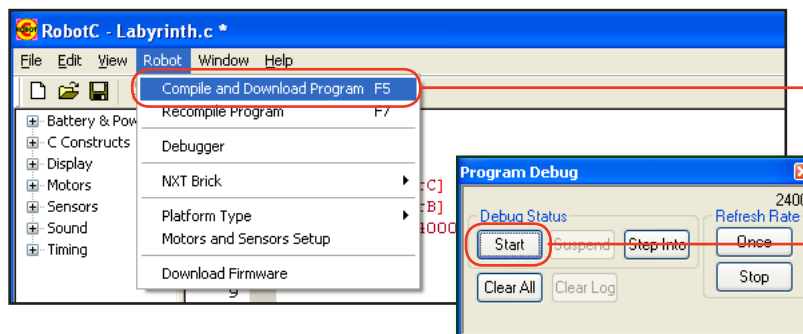
3. Since the power has been halved, try doubling the time.

```
1 task main()
2 {
3
4     motor[motorC] = 50;
5     motor[motorB] = 50;
6     wait1Msec(4000);
7
8 }
```

#### **3. Modify this code**

Since the motors are traveling at half power, double the 2000ms duration to 4000ms.

4. Download and run again.



#### **4a. Compile and Download**

Select Robot > Compile and Download Program to send your program to the robot.

#### **4b. Press Start**

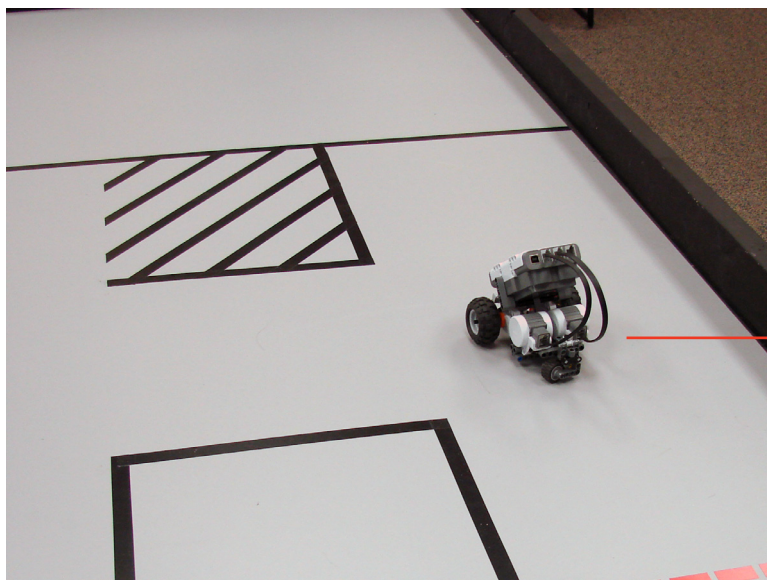
Press the Start button on the Program Debug menu that appears, to run the program.

## Movement

### Speed and Direction **Motor Power** (cont.)

#### End of Section

Your robot is traveling approximately the same distance, but at a slower speed than before. Traveling at this speed, the robot is able to maneuver more consistently, and its behaviors are easier to see and identify.



#### **Back again**

The robot now travels the correct distance again, but at a slower speed than before.

## Movement

# Speed and Direction **Turn and Reverse**

*In this lesson, you will learn how to make the robot turn and back up using different combinations of motor powers, and how to perform multiple actions in a sequence.*

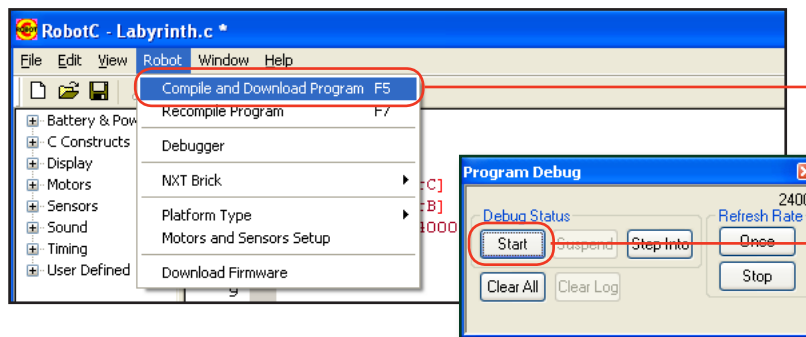
Setting both motors to half power makes the robot go slower. What do other combinations of motor powers do?

**1.** Negative numbers make the motor spin in reverse, up to -100% power.

```
1 task main()
2 {
3
4     motor[motorC] = -100;
5     motor[motorB] = -100;
6     wait1Msec(4000);
7
8 }
```

**1a. Modify this code**

Change both motors to run at -100% power.

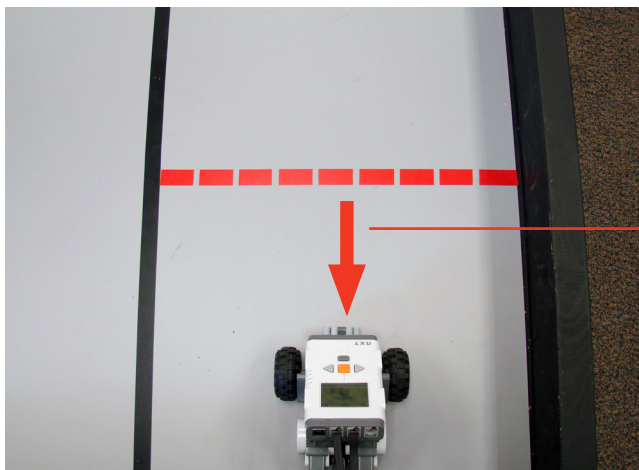


**1b. Compile and Download**

Select Robot > Compile and Download Program to send your program to the robot.

**1c. Press Start**

Press the Start button on the Program Debug menu that appears, to run the program.



**1d. Move Backward**

The robot runs in reverse with both motors set to -100% power.

## Movement

### Speed and Direction **Turn and Reverse** (cont.)

2. A motor power of 0 makes the robot stop.

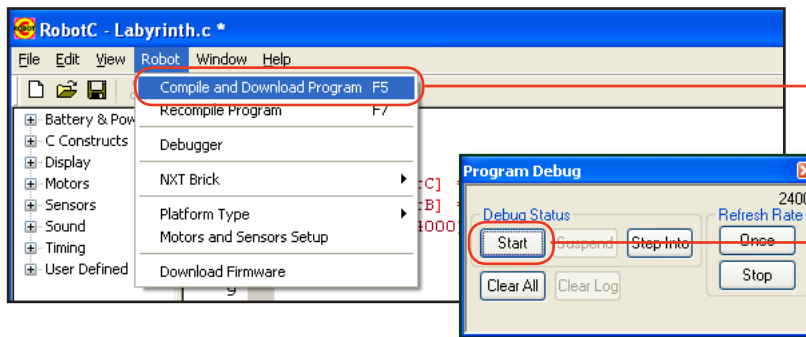
```

1 task main()
2 {
3
4     motor[motorC] = 0;
5     motor[motorB] = 0;
6     wait1Msec(4000);
7
8 }

```

**2a. Modify this code**

Change both motors to run at 0% power.

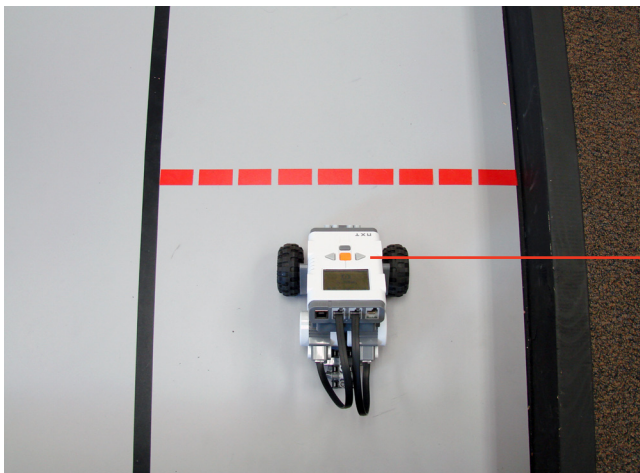


**2b. Compile and Download**

Select Robot > Compile and Download Program to send your program to the robot.

**2c. Press Start**

Press the Start button on the Program Debug menu that appears, to run the program.



**2d. Braking**

The robot holds its position and applies braking with both motors set to 0% power.

Try pushing the robot while the program is running.

## Movement

### Speed and Direction **Turn and Reverse** (cont.)

3. Giving different powers to the two motors causes the robot to turn in various ways.  
Giving them opposite powers causes the robot to turn in place.

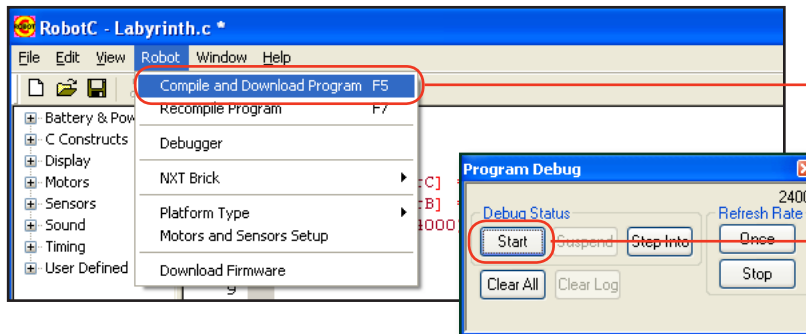
```

1 task main()
2 {
3
4     motor[motorC] = 100;
5     motor[motorB] = -100;
6     wait1Msec(4000);
7
8 }

```

**3a. Modify this code**

Change the motors to run at 100% power in opposite directions.

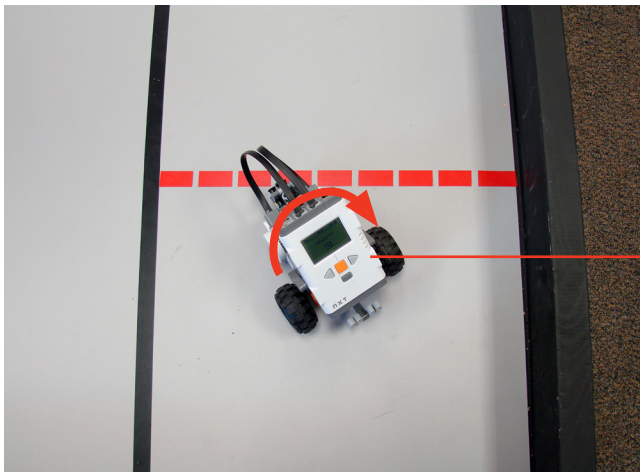


**3b. Compile and Download**

Select Robot > Compile and Download Program to send your program to the robot.

**3c. Press Start**

Press the Start button on the Program Debug menu that appears, to run the program.



**3d. Point Turn Right**

Making the left wheel go forward while the right wheel goes backward causes a "point turn" in place to the right.



## Movement

### Speed and Direction **Turn and Reverse** (cont.)

4. Making *one* wheel move while the other remains stationary causes the robot to “swing turn” with the stationary wheel acting as a pivot.

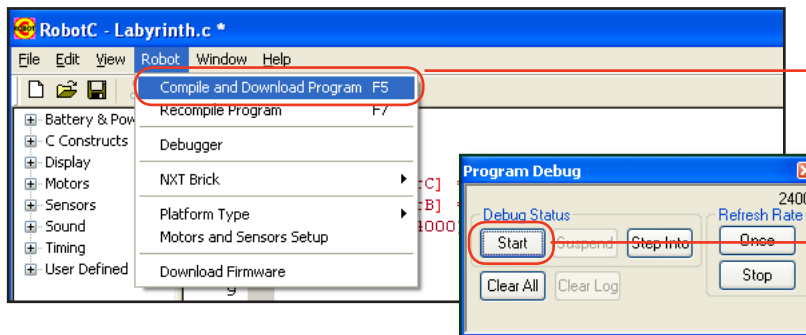
```

1 task main()
2 {
3
4     motor[motorC] = 100;
5     motor[motorB] = 0;
6     wait1Msec(4000);
7
8 }

```

**4a. Modify this code**

Make one wheel move while the holds its position.

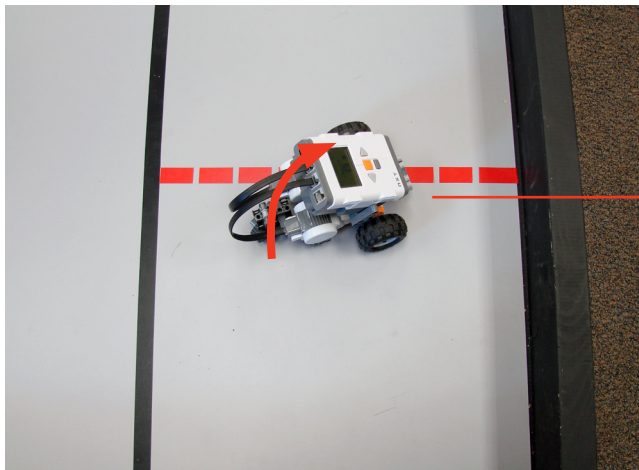


**4b. Compile and Download**

Select Robot > Compile and Download Program to send your program to the robot.

**4c. Press Start**

Press the Start button on the Program Debug menu that appears, to run the program.



**4d. Swing Turn Right**

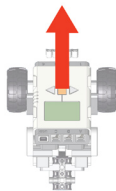
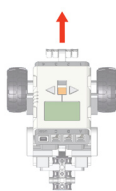
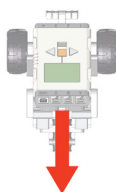
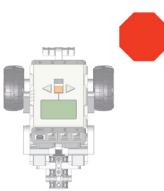
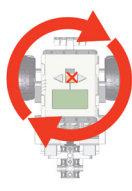
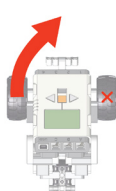
Making the left wheel go forward while holding the right wheel stationary causes a “swing turn” around the stationary wheel.

## Movement

### Speed and Direction **Turn and Reverse** (cont.)

#### Checkpoint

The following table shows the different types of movement that result from various combinations of motor powers. Remember, these commands only set the motor powers. A wait1Msec command is still needed to tell the robot how long to let them run.

Motor commands	Resulting movement
<pre>motor[motorC]=100; motor[motorB]=100;</pre>	
<pre>motor[motorC]=50; motor[motorB]=50;</pre>	
<pre>motor[motorC]=-100; motor[motorB]=-100;</pre>	
<pre>motor[motorC]=0; motor[motorB]=0;</pre>	
<pre>motor[motorC]=100; motor[motorB]=-100;</pre>	
<pre>motor[motorC]=100; motor[motorB]=0;</pre>	



## Movement

### Speed and Direction **Turn and Reverse** (cont.)

6. Finally, the robot will need to be able to perform multiple actions in a sequence. Commands in ROBOTC are run in order from top to bottom, so to have the robot perform one behavior after another, simply add the second one below the first in the code.

```

1 task main()
2 {
3
4     motor[motorC] = 50;
5     motor[motorB] = 50;
6     wait1Msec(4000);
7
8     motor[motorC] = -50;
9     motor[motorB] = 50;
10    wait1Msec(800);
11
12 }
```

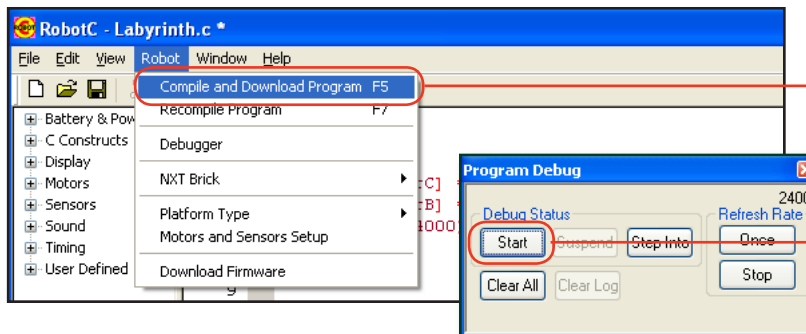
**6a. Modify this code**

Restore the first behavior to a half-power forward movement.

**6b. Add this code**

Adding a left-point-turn behavior after the moving-forward behavior will make the robot move then turn.

The turn needs only about 0.8 seconds (800ms) to complete.

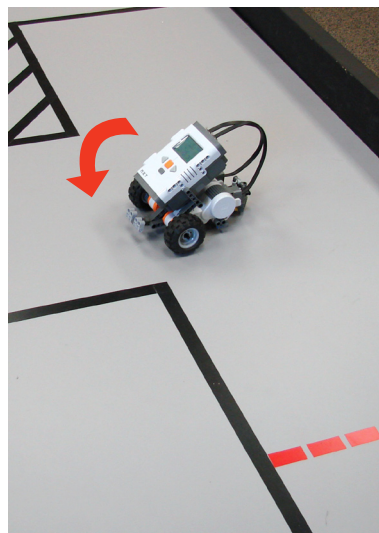
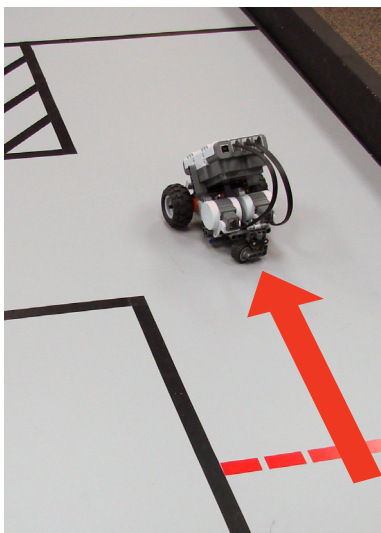


**6c. Compile and Download**

Select Robot > Compile and Download Program to send your program to the robot.

**6d. Press Start**

Press the Start button on the Program Debug menu that appears, to run the program.



**6e. Behavior Sequences**

Placing behaviors one after another in the code tells your robot to perform them in sequence.

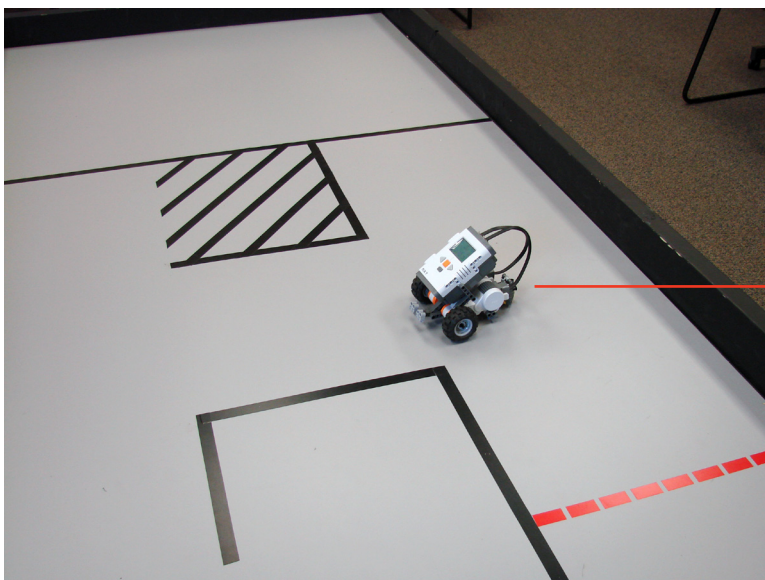
The moving-forward behavior in lines 4-6 of the program is done first (at left). The turning behavior in lines 8-10 follows immediately afterward.

## Movement

### Speed and Direction **Turn and Reverse** (cont.)

#### End of Section

You now know how to program all the necessary behaviors to navigate the Labyrinth. However, even at lowered speeds, the robot's movements are not as precise as we might like. Continue on to the Improved Movement section to learn how to clean up the robot's motion.



#### **One down...**

The robot has completed the first leg of its journey, and is ready for the next!