Teacher Answer Key: Full Speed Ahead
Introduction to Mobile Robotics > Full Speed Ahead

Construct

Observations:
1. What happened when you ran the program? 
   One of the robot's motors immediately began turning as the program ran. If the robot is sitting on the table when the motor starts running, it will start turning along a circular path on the table.

2. Which motor spun?
   The program at this point runs the motor plugged into Port C, which students set in the program. If students followed the building directions exactly, this is the robot's left motor.

3. What direction did the motor spin?
   The motor turns in the “forward” direction, because this is the default setting on the Motor block that was placed.

4. Did the motor stop spinning on its own?
   The motor does stop on its own, though rather sluggishly. This will be revisited in a later step, but for now, most students will conclude that the robot stops on its own.

5. Is this the desired behavior yet?
   No, not yet. The desired behavior is for the robot to move straight so it can go down a hall. This requires the robot to move straight down the hall, not in a circle. Therefore, this is not yet the desired behavior.

Construct: Run Second Motor

Answer the following:
6. Why is the second motor command needed?
   The second motor command makes the robot go straight instead of turning in a circle around the stationary wheel.

Construct: Stop the Robot

Answer the following:
7. Why did the robot not stop at the right place before?
   The robot didn’t come to a precise stop because it was not told to. The program didn’t have stop commands before, so as the robot reached the end of the program, it coasted to a halt rather than actually stopping on the desired spot.
8. What is the difference between downloading a program and running a program? Which one do you need to do first, and how often do you need to do it?

**Downloading a program sends the program from the computer to the NXT.** It doesn’t cause the robot to execute the code, merely store it for later use. **Running the program actually tells the NXT to execute the commands that were given.**

You must download the program from the computer to the NXT any time you write or change code, otherwise the NXT won’t have the new/modified program on it to run. You can run the program at any time after it has been downloaded.

9. Which of the following determines the order in which blocks are run in the program? Circle one.

   a. The order the blocks are placed on the workspace, regardless of where they are placed. The one you drop first runs first, etc.
   b. The order of blocks on the white Sequence beam. The program starts at the small NXT symbol, and follows the blocks in the order they are reached along the white beam.
   c. There is an order of operations of blocks. The software will always make Motor command blocks run first, then Wait For blocks, then Motor stop blocks.

(a) is incorrect. It doesn’t matter when you place any block, only where you place it. If it mattered when you placed the blocks, the program would have waited for the first motor to turn 720 degrees before starting the second, because the second Motor block was added to the program after the Wait For.

(b) is correct. The program follows the white Sequence beam and executes commands as they are reached.

(c) is incorrect. If Motor commands always executed first, there would be no way for the robot to do one thing after another – all the motor commands from the different behaviors would get smashed together and it the robot would never have a chance to actually do anything.

10. Write a brief one or two sentence explanation of what each block does in the program.

   Block 1: Turns on Motor C in the forward direction
   Block 2: Turns on Motor B in the forward direction
   Block 3: Waits until Motor C has turned 720 degrees
   Blocks 4 and 5: Turns off both motors, stopping the robot

11. Look at your program.

   i. Which icon or icons in the program controlled how far the robot went before stopping?

   The third block, the Wait For Rotations block, is the block that actually controls how far the robot goes. The two Motor blocks simply issue commands when they are reached; the robot
turns on the appropriate motor and then immediately moves on to the next block in the program. Similarly, the Motor Stop blocks just issue the stop command when they are reached. The Wait For block is the one that controls when the Motor Stop block is reached, which in turn defines the amount of time (technically, distance) the motors are left running and the robot is allowed to move.

As a test, try removing the Wait For block and seeing what happens. Your robot will appear to do nothing, because the motors are started and then immediately stopped: there is no Wait For command to space out the motor start and stop commands!

ii. Explain how you could change the program to make the robot go a longer or shorter distance.

By changing the number of degrees in the Wait For block, you will change the distance that the robot travels for.

iii. On a sheet of paper, sketch a new program that would make the robot go for twice the original distance. Make sure to include any comments or images that will identify the blocks and tell you what they do.

A program that goes for twice the distance would be the same program, with 1440 degrees of rotation on the Wait For block. Note that you can’t SEE the 1440 degrees inside the block, so it needs to be specified in either a note or an onscreen comment.

12. Describe the robot’s new movement pattern if you moved the motor plug from Port B to Port A, but did not change the program. How would you then need to change the program to make the robot go forward again?

You would see the robot start turning in place again, because only one of the motors would be running. The command to Port B would not reach a motor, and would therefore be wasted/ignored.

The program students wrote in this lesson sends commands to Ports C and B. The motor in Port C is still there to receive the motor command sent to Port C, so it will run. A movement command is also sent to Port B in the program, but there is no motor there to receive it, so nothing happens. Finally, there is no motor command sent to Port A, so even though there is a motor there, it does not move.

In order to get the motor now on Port A to turn, you need to change the Motor block that is set to “run the motor on Port B” to “run the motor on Port A”. Do not change the command to run Port C, otherwise C will no longer have the instruction to turn, and you will still only have one motor running. You will also need to change the very last Motor block to stop the motor on Port A rather than the motor on Port B. If you did not do this, the motor on port C would stop abruptly, and the motor on Port A would coast to a halt.

13. Describe the robot behavior that this program produces when run.

The robot moves forward for 720 degrees of motor rotation (two full turns of the wheels) and then stops.
14. How far will the program shown below make the robot run? Look carefully, this is trickier than it seems!

![Image of the program](image)

The robot's motors will spin for 720 degrees.

This question is designed to check for the common misconception that the comment (“Wait for 1440 degrees”) is what controls the distance the robot moves, when really it is the value entered in the Wait For block's configuration panel (720 Degrees, at the bottom right of the picture). Remember, comments don’t actually do anything to the program. They’re only there for your own reference. Changing the comment without changing the value is like putting a mustard label on a ketchup bottle and expecting to find mustard inside.

In this case, the programmer has changed the comment but not the actual value in the Wait For block. The robot will actually go for 720 degrees, as set in the panel.

Continue: Full Speed Reverse

Answer the following:

15. What program blocks are different between the moving forward and moving backward behaviors?

The Motor blocks tell the motors to move in the forward direction during the moving forward behavior, and in reverse during the moving backward behavior.

Also, the Wait For block in the moving forward behavior waits for the rotation sensor on Port C to be more than 720 degrees in the forward direction, and in the moving backward behavior it waits for the rotation sensor on Port C to be more than 720 degrees in the reverse direction.

Continue: There and Back (part 1)

Observations:

16. Did your robot perform both actions as expected? If not, what did it do instead?

If the program and robot are built exactly according to the directions, then the robot should not have performed both actions as expected. It should have gone forward for 720 degrees, but then gone backward for approximately twice that, or 1440 degrees, well past the point at which it started.
17. Why did the Rotation Sensor need to be reset?

The Rotation Sensor was reset in order for the second behavior to work properly. The second behavior is waiting for a rotation sensor in the backwards direction to be greater than 720 degrees, which is the same thing as waiting for a rotation sensor in the forward direction to be less than negative 720 degrees. To get from positive 720 to negative 720, the robot has to turn backwards 1440 degrees. But, if we reset the Rotation Sensor after the first behavior so that it reads zero, then the robot only has to get from zero to negative 720, which means it will go backwards for 720 degrees. This was the intended behavior.

18. When do you need to do this in future programs?

In future programs, it will be necessary to reset the Rotation Sensor anytime you want to use it more than once during a program. If you use it only once, it will perform the behavior as expected, but remember, all rotations are additive, which you may not expect. By resetting the sensor you ensure that each time you use it, you know exactly where it is starting and therefore how many degrees it will read, which helps you map out behaviors better.