EV3 tutorials

Moving Straight

MicroPython

By Sanjay and Arvind Seshan

BEGINNER PROGRAMMING LESSON

LESSON OBJECTIVES

1. Turning with DriveBase
2. Turning in arcs
3. Turning in place (spin and pivot turns)

TURNING WITH DRIVEBASE

The DriveBase class provides a steering input much like the Green EV3-G move blocks. However, this steering input is defined in degrees/sec rather than the ratio of power between left & right wheels.

With the Green Move Blocks, setting the steering value to 50 or 100 produces a power ratio of 0 or -1. This produces pivot and spin turns. This will be a bit more tricky with DriveBase. Arc Turns are easier to implement and will be the first topic covered in this lesson.

HOW DO YOU TURN?

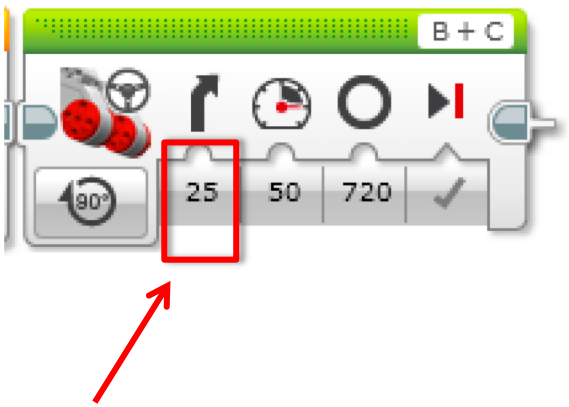
```
# This turns 90 degrees/sec right while moving 100 mm/sec
robot.drive (100, 90)
# This turns 180 degrees/sec left while moving at 500 mm/sec for 2 seconds
robot.drive_time(500, -180, 2000)
```

drive(speed, steering) → drives at `speed` mm/sec while `steering` degrees/sec until program ends or you give another command

drive_time(speed, steering, time) → drives at `speed` mm/sec while `steering` degrees/sec for `time` milliseconds

Positive steering turns to the right and negative steering turns left.

CHALLENGE 1: TURN 90 DEGREES RIGHT ALONG ARC



EV3-G Steering

The goal is to write a python program that turns 90 degrees while traveling on a circle with radius $^{50}/_{\pi}$ cm.

The robot robot will be moving along the arc associated with ¼ of a circle. The half-way point between the wheels will trace this circle.

Note that a radius of $^{50}/_{\pi}$ cm.
is a circumference of 100 cm. 90 degrees represents a quarter circle. So, you need to move 25 cm or 250mm while you turn 90 degrees.

CHALLENGE 1 SOLUTION

```
#!/usr/bin/env pybricks-micropython
from pybricks import ev3brick as brick
from pybricks.ev3devices import (Motor,TouchSensor,ColorSensor,InfraredSensor, UltrasonicSensor,
GyroSensor)
from pybricks.parameters import (Port,Stop,Direction,Button,Color,SoundFile,ImageFile,Align)
from pybricks.tools import print, wait, StopWatch
from pybricks.robotics import DriveBase

# Initialize two motors with default settings on Port B and Port C.
left_motor = Motor(Port.B)
right_motor = Motor(Port.C)
# setup wheel diameter and axle_track
wheel_diameter = 56
axle_track = 114

# setup DriveBase
robot = DriveBase(left_motor, right_motor, wheel_diameter, axle_track)

# This turns 90 deg/sec and moves 250 mm/sec 1 second
robot.drive_time(250, 90, 1000)

# This stops the motor and brakes for accuracy
robot.stop(Stop.BRAKE)
```

1

2

3

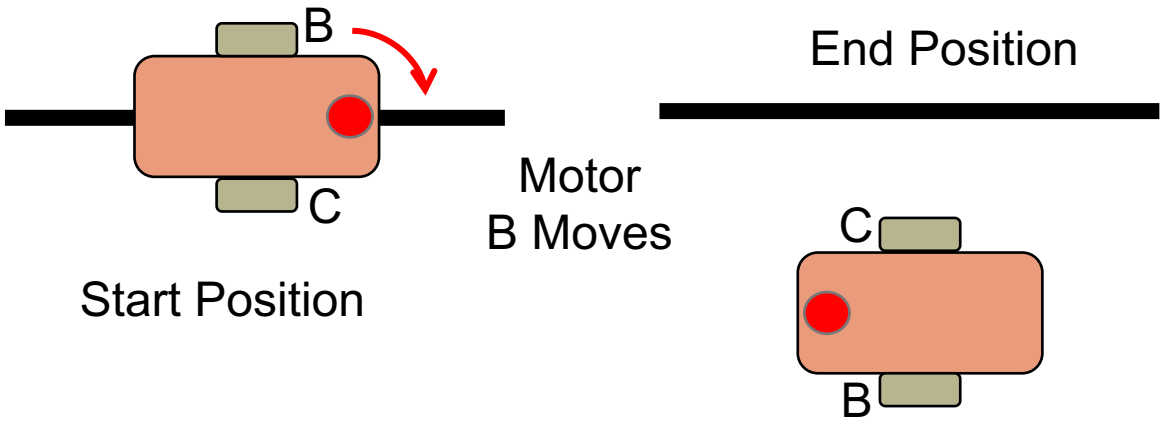
1) Above is basically the framework code described earlier. It is needed to setup the program

2) Runs the motor for 1 second at 250mm/sec and 90 deg/sec → this should turn 90 degrees and move forward 250mm

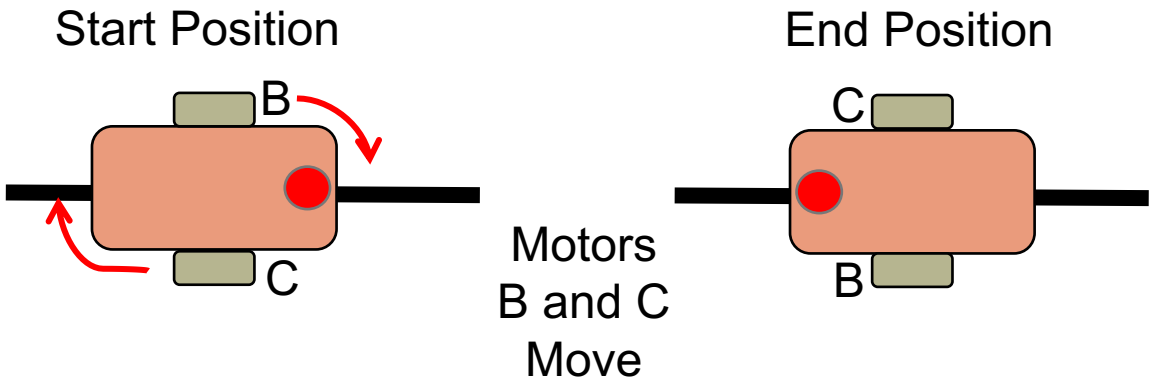
3) Stops the robot and brakes

PIVOT VS. SPIN TURNS

180 Degree Pivot Turn



180 Degree Spin Turn

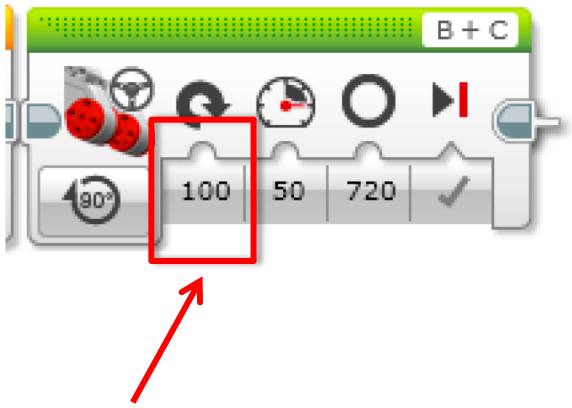


Notice where the robot ends in both pictures after a 180 degree turn.

In the Spin Turn, the robot moves a lot less and that makes Spin Turns are great for tight positions. Spin turns tend to be a bit faster but also a little less accurate.

So when you need to make turns, you should decide which turn is best for you!

CHALLENGE 2: SPIN TURN 90 DEGREES RIGHT



EV3-G Steering
100 for Spin Turn

During a spin turn, the robot only turns and doesn't move forward.

Therefore, the goal is to write a python program that turns 90 degrees while traveling on a circle with radius 0 cm.

CHALLENGE 2 SOLUTION

```
#!/usr/bin/env pybricks-micropython
from pybricks import ev3brick as brick
from pybricks.ev3devices import (Motor,TouchSensor,ColorSensor,InfraredSensor, UltrasonicSensor,
GyroSensor)
from pybricks.parameters import (Port,Stop,Direction,Button,Color,SoundFile,ImageFile,Align)
from pybricks.tools import print, wait, StopWatch
from pybricks.robotics import DriveBase

# Initialize two motors with default settings on Port B and Port C.
left_motor = Motor(Port.B)
right_motor = Motor(Port.C)
# setup wheel diameter and axle_track
wheel_diameter = 56
axle_track = 114

# setup DriveBase
robot = DriveBase(left_motor, right_motor, wheel_diameter, axle_track)

# This turns 90 deg/sec and not moving 1 second
robot.drive_time(0, 90, 1000)

# This stops the motor and brakes for accuracy
robot.stop(Stop.BRAKE)
```

1

2

3

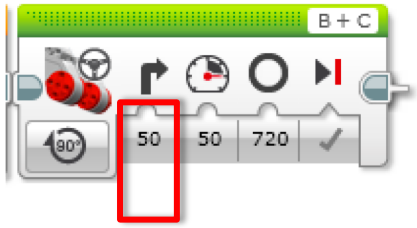
1) Above is basically the framework code described earlier. It is needed to setup the program

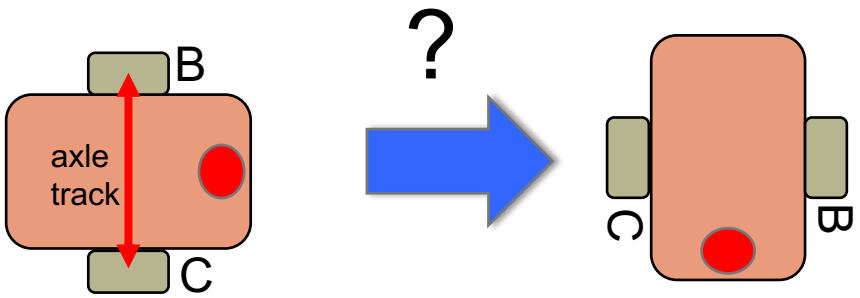
2) Runs the motor for 1 second at 0 mm/sec and 90 deg/sec → this should turn 90 degrees and stay in place, i.e. a spin turn

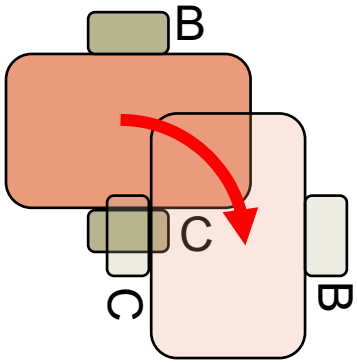
3) Stops the robot and brakes

CHALLENGE 3: PIVOT TURN 90 DEGREES RIGHT

EV3-G Steering
50 for Pivot Turn







During a pivot turn, one wheel turns and the other moves. This means that the robot moves forward as well.

The distance traveled is shown by the red arrow in the bottom left figure. Since wheel C stays at one point, the red arrow makes a circle with radius of ½ the distance between wheels (i.e. the axle track)

The circumference of this circle is $2\pi \times ({}^{axle\_track}/_{2})$. 90 degrees is ¼ of a circle. So, the distance traveled would be $({}^{\pi \times axle\_track}/_{4})$.

CHALLENGE 3 SOLUTION

```
#!/usr/bin/env pybricks-micropython
from pybricks import ev3brick as brick
from pybricks.ev3devices import (Motor,TouchSensor,ColorSensor,InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import (Port,Stop,Direction,Button,Color,SoundFile,ImageFile,Align)
from pybricks.tools import print, wait, StopWatch
from pybricks.robotics import DriveBase
import math

# Initialize two motors with default settings on Port B and Port C.
left_motor = Motor(Port.B)
right_motor = Motor(Port.C)
# setup wheel diameter and axle_track
wheel_diameter = 56
axle_track = 114

# setup DriveBase
robot = DriveBase(left_motor, right_motor, wheel_diameter, axle_track)

# This turns 90 deg/sec and not moving 1 second
robot.drive_time(math.pi * axle_track / 4, 90, 1000)

# This stops the motor and brakes for accuracy
robot.stop(Stop.BRAKE)
```

1

2

3

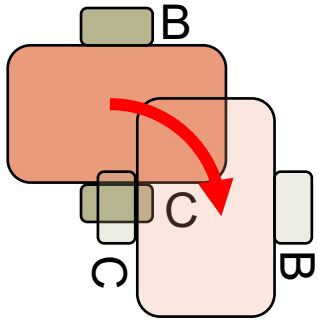
1) Is basically the framework code described earlier. One important change is the addition of ”import math” to access math functions/constants

2) Runs the motor for 1 second at 90 deg/sec. The forward speed is set to $(\pi \times axle\_track / 4)$
→ this should turn 90 degrees and one wheel should stay in place, i.e. a pivot turn

3) Stops the robot and brakes

WHAT ABOUT OTHER PIVOT TURNS

```
angle = 90
time = 1000
steering = angle * (1000/time)
dist = 2 * math.pi * (axle_track / 2) * (angle / 360) * (1000/time)
robot.drive_time(dist, steering, time)
```



The drive_time method allows you to specify speed (in mm/sec), steering (in deg/sec) and duration (in msec). For pivot turns, these inputs are related to each other.

Let's say you pick $duration = t_{msec}$. This determines how fast the robot will turn. If you want to turn $angle$ degrees. Then, $steering = angle \times (\frac{1000}{t_{msec}})$.

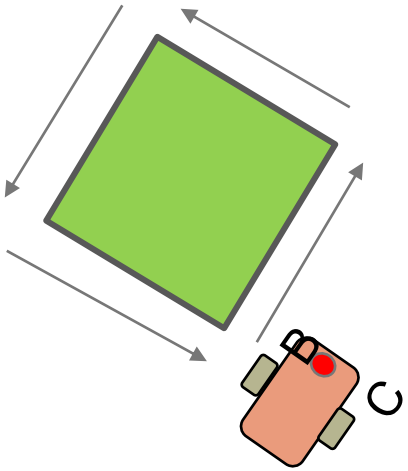
In addition, during this time, the robot must move $2\pi \times \left(\frac{axle\_track}{2}\right) \times \left(\frac{angle}{360}\right)$, i.e. the length of the red arrow in the figure. Since this in done by $duration$, you must set $\text{speed} = 2\pi \times \left(\frac{axle\_track}{2}\right) \times \left(\frac{angle}{360}\right) \times (\frac{1000}{t_{msec}})$

The code above shows how to implement a pivot turn in python.

MORE TURNING CHALLENGES

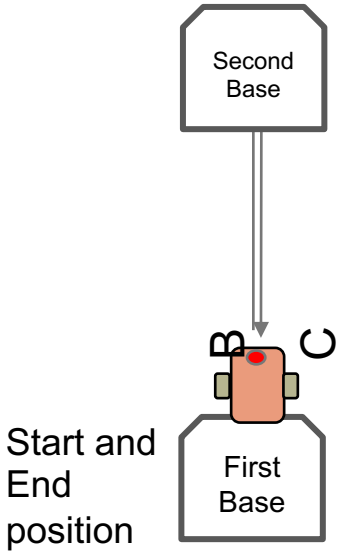
Challenge A

- Your robot is a baseball player who has to run to all the bases and go back to home plate.
- Can you program your robot to move forward and then turn left?
- Use a square box or tape



Challenge B

- Your robot baseball player must run to second base, turn around and come back to first.
- Go straight. Turn 180 degrees and return to the same spot.



CLASS DISCUSSION GUIDE

Did you try PIVOT and SPIN turns? What did you discover?

Pivot turns were fine for Challenge 1, but for Challenge 2, if we used Pivot turns, we were farther away from the base.

What situations would one work better than the other?

Spin turns are better for tight turns (places where there is not enough space) and you stay closer to your original position.

What is PSEUDOCODE? Why do you think programmers find it useful? (pseudocode is from the worksheet)

Pseudocode allows programmers to write out their code in plain English before you code in a programming language. It lets you plan and think before you sit down to code. It lets you share your ideas with others you are working with in a common language.

CREDITS

- **This tutorial was created by Sanjay Seshan and Arvind Seshan**
- **More lessons are available at www.ev3tutorials.com**