EV3 tutorials

Moving Straight

MicroPython

By Sanjay and Arvind Seshan

BEGINNER PROGRAMMING LESSON

LESSON OBJECTIVES

1. Learn how to make your robot go forward and backwards
2. Learn how to use the DriveBase class

Prerequisites:

1. Know some basic Python coding (e.g. what is a variable, how to write expressions)

GETTING STARTED

1

```
#!/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
```

2

A typical ev3 python program starts with lines like the above. The above gets created automatically with every new program.

Line (1) tells the EV3 to use micropython to run this code.

Lines marked (2) tell micropython to load particular parts of the pybricks code that you will use in this program.

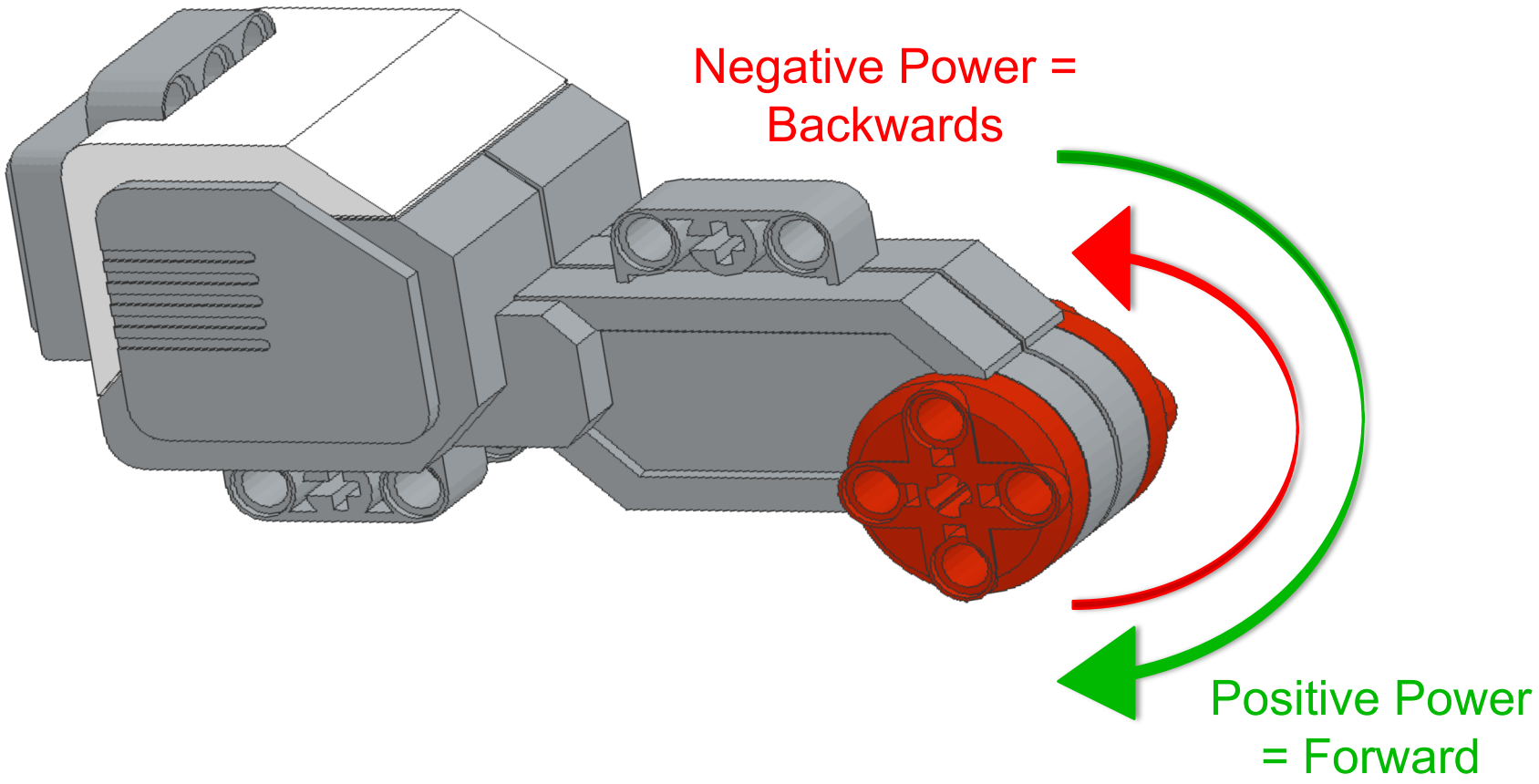
DIFFERENT WAYS TO MOVE

Just like the EV3 graphical programming environment, there are different ways to make the robot move. This lesson covers two ways:

1) DriveBase: Using the DriveBase class gives you the ability to command both motors simultaneously and have the robot steer at the same time. This is much like the “move steering” and “move tank” programming blocks in EV3-G.

2) Motor commands: Using the Motor class allows you to command each of the drive motors independently. This is much like the “large motor” and ”medium motor” programming blocks in EV3-G. This will be covered in a separate lesson.

NEGATIVE & POSITIVE POWER: BACKWARD & FORWARD



CREATING A DRIVEBASE

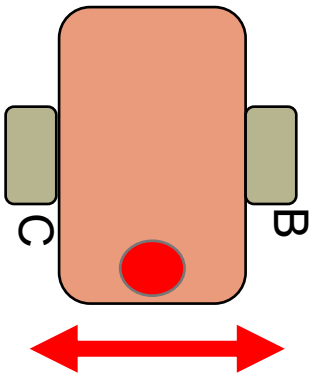
```
# Initialize two motors with default settings on Port B and Port C.

left_motor = Motor(Port.B)
right_motor = Motor(Port.C)

# The wheel diameter of the Robot Educator is 56 millimeters.
# The distance between wheels (axle_track) is 114 millimeters.

wheel_diameter = 56
axle_track = 114

# Create a DriveBase object. The wheel_diameter and axle_track values are
needed to move robot correct speed/distance when you give drive commands.

robot = DriveBase(left_motor, right_motor, wheel_diameter, axle_track)
```

1

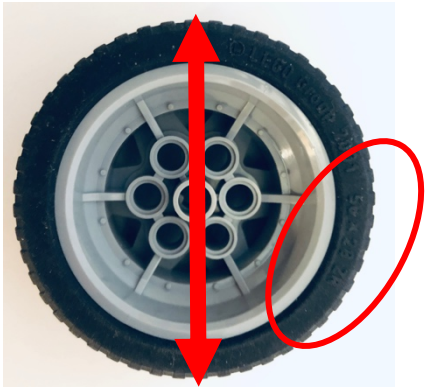
2

Before you can use simple drive commands, you need to create a DriveBase object that takes two motors, wheel diameter and axle track.

(1) Sets up some variables to store parameters and (2) creates the DriveBase.



Axle Track
(distance between wheels in mm)



Wheel Diameter
(either measure or read from tire in mm)

HOW DO YOU MOVE STRAIGHT?

```
# This drives at 100 mm/sec straight
robot.drive (100, 0)
# This drives straight backwards at 500 mm/sec for 2 seconds
robot.drive_time(-500, 0, 2000)
```

With a DriveBase object, you can drive the robot in different ways

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

What about moving for distance? or rotations? You will need to learn how to use the rotation sensor

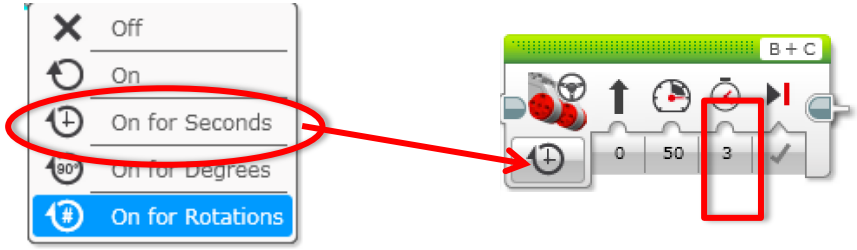
HOW DO YOU STOP?

```
# this stops any active movement and actively brakes the motor
robot.stop(Stop.BRAKE)
# this stops any active movement and leaves the motors on coast
robot.stop(Stop.COAST)
```

drive_time() will finish after the requested time. However, the motor will just coast after completion. If you want to stop the motors you must use a stop() command

The above correspond to the brake and coast commands in the EV3-G software

CHALLENGE 1: MOVE STRAIGHT (3 SECONDS)



The goal is to write a python program that moves the robot for 3 seconds at 500mm/sec and try to stop accurately at 1500mm (1.5m)

This is similar to the green block on the left. Note that the green block power is not in mm/sec but in 10s of degrees/sec. 50 power is the same as 500 degrees/sec.

CHALLENGE 1 SOLUTION: MOVE STRAIGHT (3 SECONDS)

```
#!/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 drives straight at 500 mm/sec for 3 seconds
robot.drive_time(500, 0, 3000)

# 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 3 seconds at 500mm/sec

3) Stops the robot and brakes

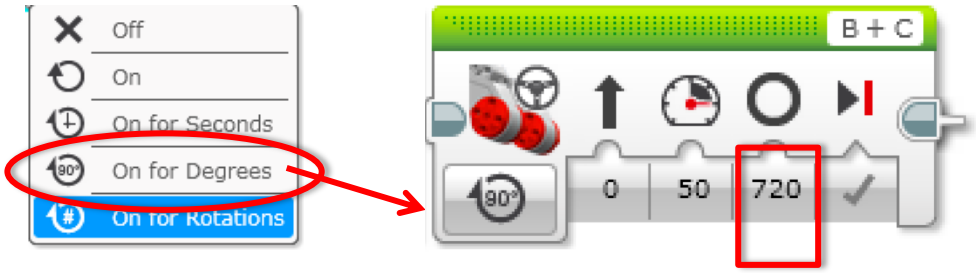
HOW DO TRAVEL A SET DISTANCE?

```
# this will reset the rotation sensor on the left motor
left_motor.reset_angle(0)
# this command reads the left motor's rotation sensor
left_motor.angle()
```

drive_time() will finish after the requested time. You can use the time and speed (in mm/sec) to estimate a distance traveled.

If you want to move a specific number of motor degrees, you will need to read the rotation sensor and use some additional python commands

CHALLENGE 2: MOVE STRAIGHT (720 DEGREES)



The goal is to write a python program that moves the robot for 720 degrees forward and stops accurately.

This is similar to the green block on the left. Note that the green block power is not in mm/sec but in 10s of degrees/sec. 50 power is the same as 500 degrees/sec.

CHALLENGE 2 SOLUTION: MOVE STRAIGHT (720 DEGREES)

```
#!/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)

# first reset the rotation sensor
left_motor.reset_angle(0)
# start the robot driving
robot.drive (200, 0)

# use a loop to wait for rotation sensor to reach 720
while left_motor.angle() < 720:
    pass
# stop the motor
robot.stop(Stop.BRAKE)
```

1) This is the same framework code used earlier

2) Reset rotation sensor and start moving. Note that unlike the EV3 you cannot do this in the opposite order.

3) Wait for rotation sensor to reach 720 degrees and then stop. Note that the wait is implemented with a loop.

NOTES

- **You cannot reset a rotation sensor while the robot is moving using the drivebase class. For example, swapping the rotation sensor reset with the robot.drive() command in challenge 2 will cause the robot to move forever.**
- **DriveBase is not completely equivalent to the green move blocks. It is missing:**
  - Wheel synchronization. If you hold one wheel the other continues to move.
  - Acceleration/Deceleration. Green move blocks include acceleration/deceleration for move degrees/rotations to improve accuracy. It is unclear if the DriveBase does this.

CREDITS

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