EV3 tutorials

Mișcarea înainte

By Sanjay and Arvind Seshan

BEGINNER PROGRAMMING LESSON

OBIECTIVELE LECȚIEI

1. Întoarcerea cu DriveBase
2. Întoarcerea în arcuri
3. Întoarcerea pe loc (rotație și viraje pivotante)

ÎNTOARCEREA CU DRIVEBASE

Clasa DriveBase furnizează un input pentru mersul înainte asemănător blocurilor de mișcare Green EV3-G. Cu toate acestea, aceast input este definit în grade/secundă, mai degrabă decât în raportul de putere între roțile din stânga și din dreapta.

În cazul blocurilor de mișcare verzi, setarea valorii de direcție de la 50 sau 100 produce un raport de putere de 0 sau -1. Acest lucru se traduce în viraje pivotante și de tip SPIN. Acest lucru va fi un pic mai dificil cu DriveBase. Întoarcele de tip arc sunt mai ușor de implementat și va fi primul subiect abordat în această lecție.

CUM VĂ ÎNTOARCEȚI?

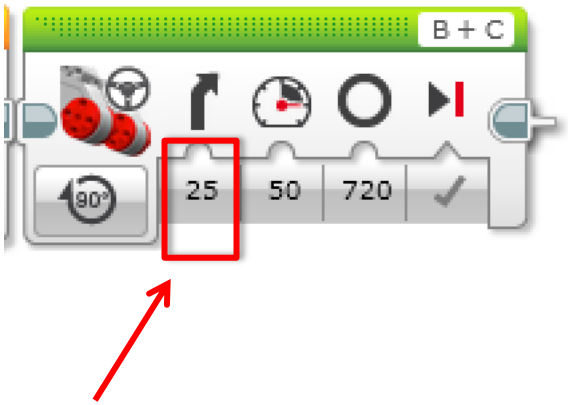
```
# Aceasta rotește 90 de grade/sec la dreapta în timp ce se deplasează 100 mm/sec.
robot.drive (100, 90)
# Aceasta rotește la 180 de grade/sec la stânga în timp ce se mișcă la 500 mm/sec timp de 2 secunde.
robot.drive_time(500, -180, 2000)
```

drive(speed, steering) → merge cu viteza `speed` în mm/sec în timp ce virează cu `steering grade`/sec până când programul se termină sau până când dați o altă comandă

drive_time(speed, steering, time) → → merge cu viteza `speed` în mm/sec în timp ce virează cu `steering` grade/sec pentru `time` millisecunde

Direcția pozitivă virează spre dreapta, iar cea negativă spre stânga.

PROVOCAREA 1: ÎNTOARCEȚI 90 DE GRADE SPRE DREAPTA DE-A LUNGUL ARC-ULUI



Direcția din EV3-G

Scopul este acela de a scrie un program python care rotește robotul la 90 de grade în timp ce se deplasează pe un cerc cu raza de $50/\pi$ cm.

Robotul se va deplasa de-a lungul arcului asociat cu ¼ de cerc. Punctul de la jumătatea distanței dintre roți va trasa acest cerc.

Rețineți că o rază de $50/\pi$ cm. reprezintă o circumferință de 100 cm. 90 de grade reprezintă un sfert de cerc. Așadar, trebuie să vă deplasați 25 cm sau 250 mm în timp ce vă întoarceți la 90 de grade.

SOLUȚIA PROVOCĂRII 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

# Inițializați două motoare cu setări implicite pe portul B și portul C.
left_motor = Motor(Port.B)
right_motor = Motor(Port.C)
# configurați diametrul roții și axle_track
wheel_diameter = 56
axle_track = 114

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

# Aceasta rotește 90 de grade/sec și se deplasează 250 mm/sec 1 secundă
robot.drive_time(250, 90, 1000)

# Acesta oprește motorul și frânează pentru precizie
robot.stop(Stop.BRAKE)
```

1

2

3

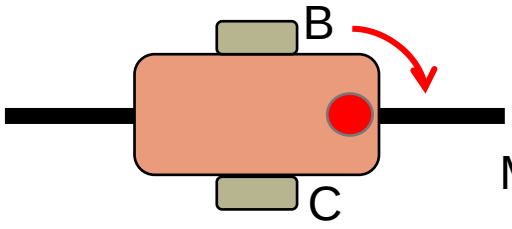
1) Mai sus este, în principiu, codul-cadru descris mai devreme. Acesta este necesar pentru a configura programul

2) Rulează motorul timp de 1 secundă la 250 mm/sec și 90 de grade/sec acesta ar trebui să se întoarcă la 90 de grade și să se deplaseze înainte cu 250 mm

3) Oprește robotul și frânează

ÎNTOARCERI DE PIVOTARE VS ROTAȚIE

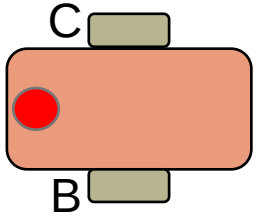
Întoarcere de pivot la 180 de grade



Poziția de start

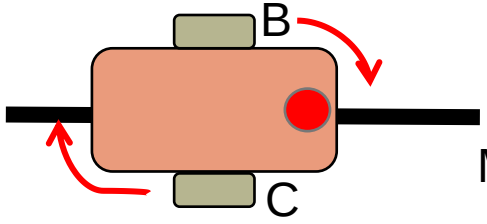
Motorul B se învârte

Poziția finală



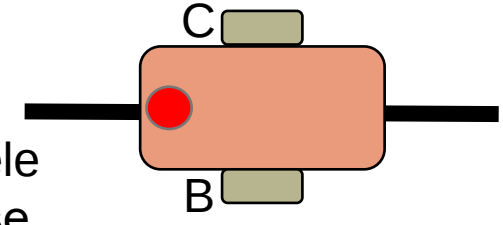
Întoarcere de tip SPIN la 180 de grade

Poziția de start



Motoarele B și C se învârt

Poziția finală

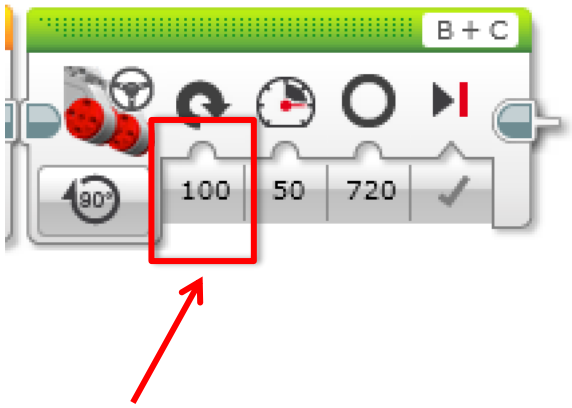


Observați unde termină robotul în ambele imagini după o întoarcere de 180 de grade.

În virajul de tip SPIN, robotul se mișcă mult mai puțin, ceea ce face ca virajele să fie extraordinare pentru spațiile strânse. Aceste întoarceri tind să fie puțin mai rapide, dar și mai puțin precise.

Așadar, atunci când trebuie să efectuați viraje, ar trebui să vă decideți ce viraj este cel mai bun pentru voi!

PROVOCAREA 2: ÎNTOARCERE PRIN ROTAȚIE DE 90 DE GRADE



Direcția din EV3-G
100 pentru viraj

În timpul unui viraj, robotul doar se întoarce și nu înaintează.

Prin urmare, obiectivul este de a scrie un program python care să rotească robotul la 90 de grade în timp ce se deplasează pe un cerc cu raza 0 cm.

SOLUȚIA PROVOCĂRII 2

```
#!/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

# Inițializați două motoare cu setări implicite pe portul B și portul C.
left_motor = Motor(Port.B)
right_motor = Motor(Port.C)
# configurați diametrul roții și axle_track
wheel_diameter = 56
axle_track = 114

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

# Acest lucru învârte 90 de grade/secundă și nu se mișcă pentru 1 secundă
robot.drive_time(0, 90, 1000)

# Acesta oprește motorul și frânează pentru precizie
robot.stop(Stop.BRAKE)
```

1

2

3

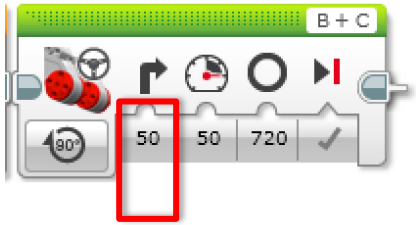
1) Mai sus este, în principiu, codul-cadru descris mai devreme. Acesta este necesar pentru a configura programul

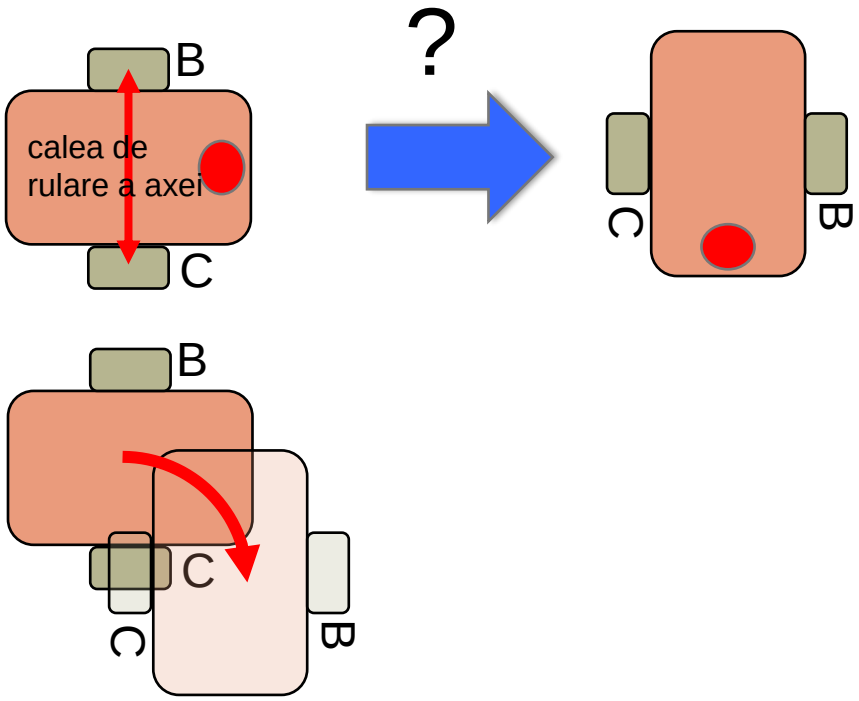
2) Rulează motorul timp de 1 secundă la 0 mm/sec și 90 de grade/sec acesta ar trebui să se rotească la 90 de grade și să rămână pe loc, adică o rotire de rotație

3) Oprește robotul și frânează

EFECTUAREA UNEI ÎNTOARCERI PIVOTANTE DE 90 DE GRADE LA DREAPTA

Direcția din EV3-G
100 pentru viraj





În timpul unui viraj cu pivot, o roată se rotește și cealaltă se mișcă. Acest lucru înseamnă că robotul se deplasează și el înainte.

Distanța parcursă este indicată de săgeata roșie din figura din stânga jos. Deoarece roata C rămâne într-un singur punct, săgeata roșie formează un cerc cu raza de ½ distanța dintre roți (adică traseul axei)

Circumferința acestui cerc este $2\pi \times ((axle\_track)/2)$. 90 de grade reprezintă ¼ dintr-un cerc. Așadar, distanța parcursă ar fi $((\pi \times axle\_track)/4)$.

SOLUȚIA PROVOCĂRII 2

```
#!/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

# Inițializați două motoare cu setări implicite pe portul B și portul C.
left_motor = Motor(Port.B)
right_motor = Motor(Port.C)
# configurați diametrul roții și axle_track
wheel_diameter = 56
axle_track = 114

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

# Acest lucru învârte 90 de grade/secundă și nu se mișcă pentru 1 secundă
robot.drive_time(math.pi * axle_track / 4, 90, 1000)

# Acesta oprește motorul și frânează pentru precizie
robot.stop(Stop.BRAKE)
```

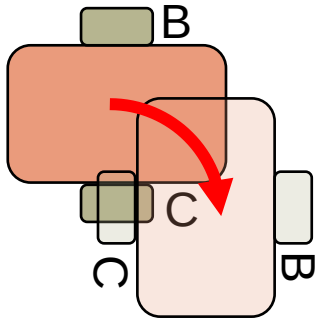
1) Este, în principiu, codul-cadru descris anterior. O modificare importantă este adăugarea "import math" pentru a accesa funcțiile/constantele matematice

2) Rulează motorul timp de 1 secundă la 90 de grade/sec. Viteza de înaintare este setată la($\pi \times axle\_track / 4$) → acest lucru ar trebui să se întoarcă la 90 de grade și o roată ar trebui să rămână pe loc, adică o întoarcere pivotantă

3) Oprește robotul și frânează

CE SE ÎNTÂMPLĂ CU ALTE ÎNTOARCERI PIVOTANTE

```
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)
```



Metoda drive_time vă permite să specificați viteza (în mm/sec), direcția (în grade/sec) și durata (în msec). În cazul virajelor cu pivot, aceste intrări sunt legate între ele.

Să presupunem că alegeți $duration = t_{msec}$. Aceasta determină cât de repede se va întoarce robotul.

Dacă doriți să vă întoarceți cu $angle$ grade.Atunci, $steering = angle \times (\frac{1000}{t_{msec}})$.

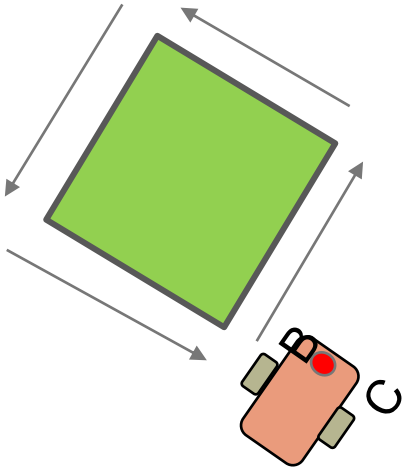
În plus, în acest timp, robotul trebuie să se deplaseze $2\pi \times \left(\frac{axle\_track}{2}\right) \times \left(\frac{angle}{360}\right)$, adică lungimea săgeții roșii din figură. Deoarece acest lucru se face prin $duration$, trebuie să setați viteza $\text{speed} = 2\pi \times \left(\frac{axle\_track}{2}\right) \times \left(\frac{angle}{360}\right) \times \left(\frac{1000}{t_{msec}}\right)$

Codul de mai sus arată cum se implementează un pivot turn în python.

MAI MULTE PROVOCĂRI DE ÎNTOARCERE

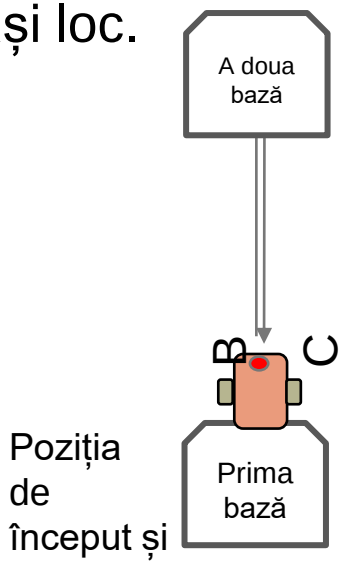
Provocarea A

- Robotul tău este un jucător de baseball care trebuie să parcurgă toate bazele și să se întoarcă la baza de start.
- Poți să îți programezi robotul să se deplaseze înainte și apoi să vireze la stânga?
- Folosiți o cutie pătrată sau o bandă adezivă



Provocarea B

- Jucătorul tău de baseball robot trebuie să alerge până la a doua bază, să se întoarcă și să revină la prima bază.
- Mergi drept. Întoarce-te la 180 de grade și întoarce-te în același loc.



GHID DE DISCUȚII ÎN CLASĂ

Ați încercat virajele de PIVOT și ROTIRE? Ce ați descoperit?

Virajele pivotante au fost bune pentru Provocarea 1, dar pentru Provocarea 2, dacă am folosi viraje pivotante, am fi mai departe de bază.

În ce situații ar funcționa unul mai bine decât celălalt?

Întoarcerile de tip SPIN sunt mai bune pentru viraje strânse (locuri unde nu există suficient spațiu) și rămâneți mai aproape de poziția inițială.

Ce este pseudocodul? De ce credeți că programatorii îl consideră util? (pseudocodul provine din fișa de lucru)

Pseudocodul le permite programatorilor să scrie codul lor într-un limbaj comun engleză/română simplă înainte de a coda într-un limbaj de programare. Acesta vă permite să planificați și să gândiți înainte de a vă așeza să programați. Vă permite să vă împărtășiți ideile cu alte persoane cu care lucrați într-un limbaj comun.

CREDITE

Această lecție de Mindstorms a fost realizată de Sanjay Seshan și Arvind Seshan.

Mai multe lecții sunt disponibile pe ev3lessons.com

Această lecție a fost tradusă în limba română de echipa de robotică FTC – ROSOPHIA #21455 RO20.