



FAKULTA
ELEKTROTECHNICKÁ
ČVUT V PRAZE

BRIAN PŘÍRUČKA

Python

JAN ŘEHÁK, ONDŘEJ VAŠÁTKO

VERZE 0.9

Obsah

Úvod

Stavba robota	3
Příprava prostředí	4
Připojení přes wifi	5
1. Hello World	6
2. Píp píp	7

Motory

1. Tak to rozjedem	8
2. Na plný plyn	9
3. Dopředu a dozadu	10
4. Zatáčka	11
5. Točím se točím	12
6. Spirála	14
7. Pilot	16

Senzory

1. Detekce stisku	18
2. Detekce stisku	19
3. Zpomalit nebo zrychlit	21
4. Zastavit, stát!	22
5. Nenabourej	23
6. Nenabourej 2	24
7. Zastavit, stát! 2	25
8. Gyro senzor	26

Pokročilé úlohy

1. Sledování černé čáry	27
2. Sledování černé čáry 2	28
3. Knob na brianovi	30
6. Simon Says	31

Neřešené úlohy

1. Sledování černé čáry 3	32
2. Překážka na čáře	33
3. Bludiště	34
4. Sumo	35

STAVBA ROBOTA

ÚKOL

Naskenujte QR kód a podle návodu postavte robota.



TIP

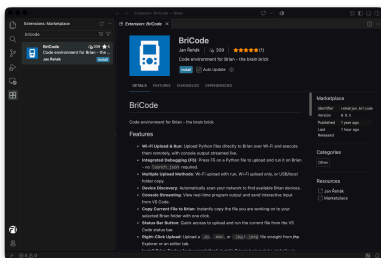
Pokud vám nefunguje odkaz, můžete jít na adresu <https://tinyurl.com/brian-line-follower>

ÚKOL

Připravte si programovací prostředí.

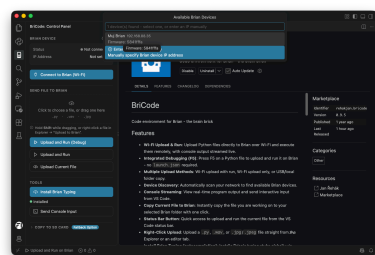
POSTUP ŘEŠENÍ

1. Otevřete si Visual Studio Code (ke stažení na <https://code.visualstudio.com/>).
2. V záložce Extensions vyhledejte a nainstalujte rozšíření BriCode.

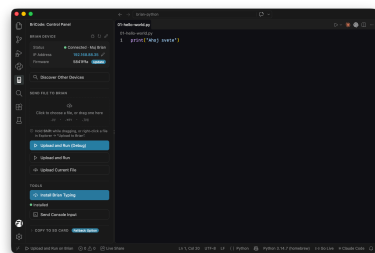


V levém panelu se objeví záložka BriCode.

3. Vytvořte si svoji pracovní složku a otevřete ji ve VS Code.
4. Připojte Briana ke stejné Wi-Fi síti jako počítač (viz kapitola Připojení přes wifi).
5. Spojte se s Brianem tlačítkem **Connect to Brian (Wi-Fi)**. Pokud se Brian nezobrazí automaticky, zadejte ručně jeho IP adresu.



6. Python soubory pak spouštíte tlačítkem **Upload and Run (Debug)**.



TIP

Rozšíření vám automaticky stáhne typing pro vaši verzi Briana.

PŘIPOJENÍ PŘES WIFI

PŘIPOJENÍ K MÍSTNÍ SÍTI

1. V menu **Settings** → **WiFi config** vyberte **Connect to...**, zobrazí se dostupné sítě.

- Brian musí být v **scan mode**, aby se seznam ukázal.
- Pokud žádné sítě nevidíte, spusťte skenování přes volby **Start WiFi**, **Disconnect** nebo **Disconnect AP**.

2. Vyberte síť a klikněte na **Connect**.

- Ikona zámku znamená, že je vyžadováno heslo.
- Pokud je potřeba, Brian vás vyzve k zadání hesla.

3. **Known (saved) networks** se zobrazují dole na obrazovce.

- V menu **Manage** (Right button) můžete síť zapomenout (**Forget**) nebo vypnout automatické připojení (**Disable Auto-join**).
- Brian si pamatuje až 12 sítí.

ZAPNUTÍ VLASTNÍ SÍTĚ

1. V menu **WiFi config** zvolte **Configure AP**.

2. Zadejte **Name (SSID)** a **password**, nebo nastavte otevřenou síť.

3. Poté spusťte (nebo restartujte) **AP**.

- Brian se odpojí od všech jiných WiFi sítí, pokud byl připojen.
- V AP režimu se mohou připojit až 3 zařízení.

(RE)BOOT BEHAVIOR

V menu **WiFi config** vyberte **Boot behavior**, kde určíte automatické zapínání WiFi.

BRIVIS PŘES WI-FI

Na připojeném počítači nebo telefonu otevřete webový prohlížeč.

Oficiálně je podporován **Google Chrome**, ale fungují i jiné hlavní prohlížeče.

1. Zadejte **IP address** zobrazenou v **WiFi config**.
2. Otevře se **BriVis**, který nabízí tyto funkce:

Funkce

- File transfer mezi prohlížečem a SD kartou Briana
- Editable file preview: náhled a úprava souborů (některé typy lze editovat jako text), BriVis soubory se otevrou přímo v blokovém editoru
- Program start/control: spouštění libovolných spustitelných souborů na Brianovi přímo z BriVisu, možnost přerušení nebo ukončení
- Firmware update check: pokud má zařízení internet, BriVis upozorní na novou verzi a nabídne odkaz na aktualizací nástroj (samotná aktualizace probíhá přes USB)
- Program console: připojení k běžícímu programu, zobrazení výstupu a posílání příkazů
- Settings: úprava uložených parametrů (jas, WiFi boot behavior atd.)
- Motor demo: vzdálené testování a spouštění motorů z prohlížeče
- Sensor demo: vzdálené čtení hodnot senzorů v prohlížeči

1. HELLO WORLD

ÚKOL

Úkolem je vypsát text "Hello World" na kostce brian.

POZOR

Nezapomeňte kostku Brian pořádně nabít a zapnout, aby byla připravená! ⚡

NÁPOVĚDA

Napište `print()` a do závorek napište ve dvojitéch uvozovkách, co chcete vytisknout

POZOR

Zkuste si vytisknout i jiné fráze, ale pozor, Brian neumí tisknout české znaky, musíte proto psát bez háčků a čárek

POSTUP ŘEŠENÍ

```
print("Hello World")
```

1. Zapněte VS Code
2. Nastavte rozšíření BriCode podle předchozí kapitoly
3. Klikněte na Upload and Run (Debug)
4. Koukněte se na obrazovku kostky

A je to! Kostka Brian by teď měla říct „Hello World“. Skvělá práce, programátore! 🥳

2. PÍP PÍP

ÚKOL

Úkolem je napsat program, který zahraje tón.



TIP

Pro každou úlohu si doporučujeme vytvořit nový soubor



NÁPOVĚDA

Na začátku použijte funkci `play_tone( . . . )`. Do závorek napište dvě čísla: první určuje výšku tónu a druhé délku tónu v milisekundách ($1000\text{ms} = 1\text{s}$).



NÁPOVĚDA

Po zavolání funkce `play_tone` nesmíme předčasně ukončit program než tón dohraje. Použijeme tedy funkci `sleep(0.5)`, která chvíli počká. Aby se funkce dala použít musí na začátku programu být `from time import sleep`.



TIP

Doporučujeme příjemnou výšku tónu 880, můžete ale experimentovat s dalšími tóny.



POZOR

Zkontrolujte, že máte na kostce zapnutou hlasitost. V menu jděte do `Settings` -> `Sound volume` a nastavte hlasitost.

POSTUP ŘEŠENÍ

```
from brian.audio import play_tone
from time import sleep

play_tone(880, 500) # tón A2 (880 Hz) po dobu 500 ms
sleep(0.5) # počkej na zahrání tónu
```



PŘIPOJENÍ MOTORU

Abychom mohli s motorem pracovat, musíme nejdříve pythonu oznámit, že ho budeme používat a poté si ho musíme pojmenovat.

```
from brian.motors import EV3LargeMotor, MotorPort # Oznámí, co budeme používat

motor_1 = EV3LargeMotor(MotorPort.B) # Řekne, kam jste motor připojili, a pojmenuje ho
```

ÚKOL

Naprogramovat jeden motor tak, aby se otočil o 360° rychlostí 500°/s.



NÁPOVĚDA

Řekněte motoru, co má udělat: napište `motor_1`, tečku a poté `rotate(...)`.

Do závorek napište dvě čísla: první je, o kolik stupňů se má otočit, druhé je jakou rychlostí.

POSTUP ŘEŠENÍ

```
from brian.motors import EV3LargeMotor, MotorPort

motor_1 = EV3LargeMotor(MotorPort.B)

motor_1.rotate(360, 500) # Otoč o 360 stupňů, rychlostí 500 stupňů za sekundu
```



ÚKOL

Naprogramovat robota tak, aby popojel vpřed o jedno otočení koly.



NÁPOVĚDA

Když nastavíme třetí parametr funkce `rotate(...)` na `0` tak se na dokončení funkce nebude čekat a můžeme tedy začít otáčet i druhým motorem. Takže ve výsledku se budou oba motory otáčet zároveň.

POSTUP ŘEŠENÍ

```
from brian.motors import EV3LargeMotor, MotorPort

motor_l = EV3LargeMotor(MotorPort.B)
motor_r = EV3LargeMotor(MotorPort.C)

motor_l.rotate(360, 500, 0) # Třetí argument 0 znamená, že se na dokončení pohybu nečeká a program
                             # hned pokračuje dalším řádkem.
motor_r.rotate(360, 500)
```



TIP

Maximální rychlost pro velký motor je `1050 deg/s` (stupňů za sekundu) a `1560 deg/s` pro střední motor

3. DOPŘEDU A DOZADU



ÚKOL

Naprogramovat robota tak, aby popojel dopředu a poté dozadu.



NÁPOVĚDA

Záporné číslo v prvním parametru funkce `rotate(...)` popožene robota dozadu.

POSTUP ŘEŠENÍ

```
from brian.motors import EV3LargeMotor, MotorPort

motor_l = EV3LargeMotor(MotorPort.B)
motor_r = EV3LargeMotor(MotorPort.C)

motor_l.rotate(360, 500, 0)
motor_r.rotate(360, 500)

motor_l.rotate(-360, 500, 0)
motor_r.rotate(-360, 500)
```



ÚKOL

Naprogramovat robota tak, aby se otočil do pravého úhlu.



NÁPOVĚDA

Aby se robot otočil, musíme jedním motorem otočit na jednu stranu a druhým na druhou.



VYZKOUŠEJTE SI

Experimentujte s rychlostí jednotlivých motorů, aby robot vytvořil různě velké zatáčky.

POSTUP ŘEŠENÍ

```
from brian.motors import EV3LargeMotor, MotorPort

motor_l = EV3LargeMotor(MotorPort.B)
motor_r = EV3LargeMotor(MotorPort.C)

# Každé kolo se otočí o 250°, tím se robot otočí asi o 90°
motor_l.rotate(250, 500, 0)
motor_r.rotate(-250, 500)
```



TIP

Hodnota 250 je jen odhad. Vyzkoušejte, o kolik stupňů se musí kola otočit, aby se robot otočil přesně o 90°.

PROČ OPAKOVAT PŘÍKAZY

Představte si, že chcete robotovi 10× říct „Ahoj, Briane!“. Místo psaní stejného příkazu pořád dokola mu můžeme dát úkol zopakovat ho za nás. Slouží k tomu **for cyklus**.

```
for i in range(10):  
    print("Hello Brian!")
```

CO SE DĚJE UVNITŘ

Všechno, co má být součástí cyklu, musí být odsazené (tabulátor nebo čtyři mezery). Díky tomu Python ví, co má opakovat.

```
for i in range(3):  
    print("Jsem ve for-cyklu...")  
print("...a já už ne.")
```

ZKOUŠÍME ČÍSLA

Proměnná `i` postupně nabývá čísel od začátku rozsahu až po jeho konec (ten už se nevypíše). Můžeme si hrát s tím, kde začneme, kde skončíme nebo o kolik budeme přeskakovat.

```
for i in range(4, 10):  
    print(i)  
  
for i in range(0, 10, 2):  
    print(i)
```

Často číslo v programu ani nepotřebujeme. Pak místo něj používáme podtržítka.

```
for _ in range(5):  
    print("Jdeme na to!")
```

TIP PRO PŘERUŠENÍ

Když chceme cyklus zastavit dřív, použijeme `break`. Pokud naopak chceme přeskočit jen jednu část a pokračovat dál, použijeme `continue`.

ÚKOL

Úkolem je naprogramovat robota tak, aby objel čtverec. Použijte *for cyklus*.

POSTUP ŘEŠENÍ

```
from brian.motors import EV3LargeMotor, MotorPort

motor_l = EV3LargeMotor(MotorPort.B)
motor_r = EV3LargeMotor(MotorPort.C)

for _ in range(4): # Proměnou nepotřebujeme, vyplníme jí proto _ (podtržítkem)
    # Popojedeme dopředu
    motor_l.rotate(360, 500, 0)
    motor_r.rotate(360, 500)

    # A teď se otočíme o 90°
    motor_l.rotate(250, 500, 0)
    motor_r.rotate(-250, 500)

# A celé to zopakujeme 4x
```



TIP

Hodnota 250 je jen odhad. Vyzkoušejte, o kolik stupňů se musí kola otočit, aby se robot otočil přesně o 90°.

INTRO

Uvnitř *for* cyklu mohou být libovolné příkazy a to i další *for* cyklus. V takovém případě se mu říká vnořený *for* cyklus. Pojďme se podívat na to, jak funguje:

```
for i in range(4):  
    for j in range(i): # Vnořený cyklus  
        print(j)  
    print("---")
```

Výstup tohoto programu bude:

```
---  
0  
---  
0  
1  
---  
0  
1  
2  
---
```

V prvním průběhu vnějšího *for* cyklu je `i=0`, tedy vnořený *for* cyklus neproběhne ani jednou a žádné číslo se nevypíše. V druhém kroku vnějšího *for* cyklu, kdy je `i=1` už vnořený *for* cyklus proběhne právě jednou a vytiskne se `0`.

ÚKOL

Naprogramovat robota tak, aby se točil do spirály. Použijte vnořený *for* cyklus.



NÁPOVĚDA

Vnější *for* cyklus můžeme napsat třeba jako `for length in range(200, 601, 200):`, budeme tak mít proměnnou délku, která se bude prodlužovat o 200 každou iteraci. Vnitřní cyklus pak každou délku ujede dvakrát, takže vznikne spirála.

POSTUP ŘEŠENÍ

```
from brian.motors import EV3LargeMotor, MotorPort

motor_l = EV3LargeMotor(MotorPort.B)
motor_r = EV3LargeMotor(MotorPort.C)

for length in range(200, 601, 200): # Horní limit musí být o jedna větší než požadovaná poslední
    hodnota v cyklu, rozmysli si proč
    for _ in range(2): # Každou délku ujedeme dvakrát, pak se prodlouží
        motor_l.rotate(length, 500, 0)
        motor_r.rotate(length, 500)

        motor_l.rotate(250, 500, 0) # Otočka o 90°
        motor_r.rotate(-250, 500)
```



TIP

Hodnota 250 je jen odhad. Vyzkoušejte, o kolik stupňů se musí kola otočit, aby se robot otočil přesně o 90°.

PILOT

Doteď jsme každý motor řídili zvlášť a počítali, o kolik stupňů se má otočit. Jednodušší je použít **pilota**. Ten řídí oba motory najednou: řekneme mu, kolik milimetrů má ujet nebo o kolik stupňů se má otočit robot, a pilot si sám spočítá, jak mají jet motory.

Abyste spočítal, musí znát dva rozměry robota: **vzdálenost mezi koly** (měří se od středu jednoho kola ke středu druhého, u našeho robota 110 mm) a **průměr kol** (40 mm).

```
from brian.motors import EV3LargeMotor, MotorPort
from brian.pilot import Pilot # Oznámí, že budeme používat pilota

motor_l = EV3LargeMotor(MotorPort.B)
motor_r = EV3LargeMotor(MotorPort.C)

# Pilot řídí oba motory najednou. Musí vědět, jak je robot velký.
pilot = Pilot(motor_l, motor_r, wheelbase_mm=110, wheel_diameter_mm=40)
pilot.set_speed(200) # Rychlost v milimetrech za sekundu
```

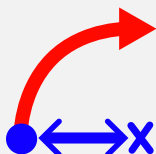
- `pilot.forward(200)` – ujet 200 mm dopředu
- `pilot.backward(200)` – ujet 200 mm dozadu
- `pilot.turn(0, 90)` – otočit se na místě o 90° doprava (záporný úhel otáčí doleva)
- `pilot.forward()` – jed' dopředu, dokud pilota nezastavíme pomocí `pilot.brake()`

OTÁČENÍ, OBLOUK A ZATÁČENÍ

Pilot umí zatáčet třemi způsoby. Liší se tím, co mu zadáme: o kolik se má robot otočit, po jak velké kružnici má jet, nebo jak moc má zatáčet.



Otáčení otočí robota o zadaný úhel. S poloměrem 0 se točí na místě kolem svého středu.



Oblouk jede po kružnici se zadaným poloměrem, dokud neujede zadanou vzdálenost. Záporný poloměr zatáčí doleva.



Zatáčení funguje jako volant: 0 jede rovně, 0.5 jedno kolo stojí, 1 otáčí na místě. Záporné číslo zatáčí doleva.

V Pythonu: `pilot.turn(0, 90)`, `pilot.arc(150, 300)` a `pilot.travel(0.3, 500)`.



TIP

Od teď budeme jezdit s pilotem. Motory zvlášť použijeme jen tam, kde potřebujeme každé kolo řídit jinak, například u regulátorů.

ÚKOL

Objedte znovu čtverec z úlohy 5, tentokrát s pilotem. Strana čtverce bude 20 cm.

POSTUP ŘEŠENÍ

```
from brian.motors import EV3LargeMotor, MotorPort
from brian.pilot import Pilot

motor_l = EV3LargeMotor(MotorPort.B)
motor_r = EV3LargeMotor(MotorPort.C)

pilot = Pilot(motor_l, motor_r, wheelbase_mm=110, wheel_diameter_mm=40)
pilot.set_speed(200)

for _ in range(4):
    pilot.forward(200) # Jeď 20 cm dopředu
    pilot.turn(0, 90) # Otoč se na místě o 90° doprava
```

TLAČÍTKO

První senzor, který si ukážeme, je `TouchSensor`.

Robot dokáže poznat, jestli je zmáčknuté. Díky tomu s ním můžete třeba spouštět různé funkce nebo zjistit, že robot narazil do zdi.



PŘIPOJENÍ TLAČÍTKA

Abychom mohli s tlačítkem pracovat, musíme nejdříve pythonu oznámit, že ho budeme používat a poté si ho musíme pojmenovat. Stejně jak u motoru.



UPOZORNĚNÍ

Když nebude senzor připojen do daného portu, tak se program spustí a zůstane běžet.

```
from brian.sensors import SensorPort # Oznámí, co budeme používat
from brian.sensors.EV3 import TouchSensorEV3

button = TouchSensorEV3(SensorPort.S3) # Řekne, kam jste senzor připojili, a pojmenuje ho
```

ÚKOL

Počkejte na zmáčknutí tlačítka a potom napište na obrazovku Briana `Zmacknuto`.



NÁPOVĚDA

Použijte funkci `wait_for_press()`

POSTUP ŘEŠENÍ

```
from brian.sensors import SensorPort
from brian.sensors.EV3 import TouchSensorEV3

button = TouchSensorEV3(SensorPort.S3)

button.wait_for_press()
print('Zmacknuto')
```



WHILE CYKLUS

V některých případech dopředu nevíme, kolikrát budeme muset nějaký příkaz nebo nějaké příkazy vykonat, a nemůžeme proto použít *for*-cyklus, který má pevně daný počet opakování. Pro tyto případy můžeme použít *while* cyklus, který opakuje příkazy ve svém těle dokud je splněna nějaká podmínka.

Použití *while* cyklu si ukážeme na jednoduchém příkladu, kde se na obrazovku Briana bude tisknout `Zmacknuto`, dokud budeme držet tlačítko.

```
from brian.sensors import SensorPort
from brian.sensors.EV3 import TouchSensorEV3

button = TouchSensorEV3(SensorPort.S3)

button.wait_for_press() # Počkáme na začátek stisku tlačítka
while button.is_pressed(): # Dokud je tlačítko zmáčknuto, cyklus pokračuje
    print("Zmacknuto.") # A tiskne pořát dokola
print("Ted je pusteno.")
```

Může se stát, že chceme aby cyklus běžel pořád, bez ukončení. Můžeme na to použít slovo `True` v podmínce. Opět si ukážeme jednoduchý příklad.

```
from brian.sensors import SensorPort
from brian.sensors.EV3 import TouchSensorEV3

button = TouchSensorEV3(SensorPort.S3)

button.wait_for_press() # Počkáme na začátek stisku tlačítka
while True: # Cyklus se nikdy nezastaví
    print("Zmacknuto.") # A tiskne pořát dokola
# Sem kód už nikdy nedorazí
```



VYZKOUŠEJTE SI

V následujících úlohách se může stát, že *while* cyklus nepůjde opustit a program se sám nezastaví. Zastavit ho ale můžete stisknutím obou horních tlačítek kostky současně.



VYZKOUŠEJTE SI

Všechny programy zapsané pomocí *for* cyklu je možné přepsat i pomocí *while* cyklu, ne obráceně.

ÚKOL

Rozjedte se dopředu a po zmáčknutí tlačítka otočte směr jízdy.



NÁPOVĚDA

Bude potřeba proměnná, která si bude pamatovat, jakým směrem robot jede.

POSTUP ŘEŠENÍ

```
from brian.motors import EV3LargeMotor, MotorPort
from brian.pilot import Pilot
from brian.sensors import SensorPort
from brian.sensors.EV3 import TouchSensorEV3

button = TouchSensorEV3(SensorPort.S3)

motor_l = EV3LargeMotor(MotorPort.B)
motor_r = EV3LargeMotor(MotorPort.C)
pilot = Pilot(motor_l, motor_r, wheelbase_mm=110, wheel_diameter_mm=40)
pilot.set_speed(200)

pilot.forward()
forward = True

while (True):
    button.wait_for_press()
    if forward:
        pilot.backward() # Jeď dozadu
        forward = False
    else:
        pilot.forward() # Jeď dopředu
        forward = True
```



TIP

Použijte funkci `wait_for_press()`

3. ZPOMALIT NEBO ZRYCHLIT



ÚKOL

Rozjedte se dopředu. Zmáčknutím jednoho tlačítka by se měl robot zrychlit a zmáčknutím druhého zase zpomalit.

POZOR

Rychlost pilota zadáváme v milimetrech za sekundu. S našimi koly robot zvládne nejvýše asi 360 mm/s a rychlost by neměla klesnout až na 0.

POZOR

Ve smyčce použijte `sleep(0.1)`, aby se jedno stisknutí tlačítka nezaznamenalo jako několik naráz. Nezapomeňte na začátku programu přidat `from time import sleep`

NÁPOVĚDA

Budete potřebovat proměnnou, která si bude pamatovat rychlost jízdy.

POSTUP ŘEŠENÍ

```
from brian.motors import EV3LargeMotor, MotorPort
from brian.pilot import Pilot
from brian.sensors import SensorPort
from brian.sensors.EV3 import TouchSensorEV3
from time import sleep

# Velkými písmeny si hned na začátku určím konstanty (neměnné hodnoty)
LOWER_LIMIT = 10
UPPER_LIMIT = 360 # mm/s, rychleji naše kola nezvládnou

button1 = TouchSensorEV3(SensorPort.S3)
button2 = TouchSensorEV3(SensorPort.S4)

motor_l = EV3LargeMotor(MotorPort.B)
motor_r = EV3LargeMotor(MotorPort.C)
pilot = Pilot(motor_l, motor_r, wheelbase_mm=110, wheel_diameter_mm=40)

speed = 200
pilot.forward(speed=speed)
while (True):
    if button1.is_pressed() and speed < UPPER_LIMIT:
        speed += 10 # Přičtu 10 k rychlosti
        pilot.forward(speed=speed) # Jeď dál novou rychlostí
    if button2.is_pressed() and speed > LOWER_LIMIT:
        speed -= 10 # Odečtu 10 od rychlosti
        pilot.forward(speed=speed)
    sleep(0.1)
```

4. ZASTAVIT, STÁT!



ÚKOL

Upevněte tlačítko na předek robota. Rozjedte se dopředu a jakmile tlačítko narazí na zeď, měl by robot automaticky zastavit.

POSTUP ŘEŠENÍ

```
from brian.motors import EV3LargeMotor, MotorPort
from brian.pilot import Pilot
from brian.sensors import SensorPort
from brian.sensors.EV3 import TouchSensorEV3

button = TouchSensorEV3(SensorPort.S3)

motor_l = EV3LargeMotor(MotorPort.B)
motor_r = EV3LargeMotor(MotorPort.C)
pilot = Pilot(motor_l, motor_r, wheelbase_mm=110, wheel_diameter_mm=40)
pilot.set_speed(200)

pilot.forward()

button.wait_for_press() # Počkej, až se zmáčkne tlačítko

pilot.brake()
```

ULTRASONICKÝ SENZOR

Dost často potřebujeme měřit vzdálenost. K tomuto účelu slouží `UltrasonicSensor`.



Ten pomocí zvuku a jeho zpětného odražení dokáže měřit vzdálenost.



POZOR

Protože senzor používá zvuk, nemusí dávat ideální výsledky při odražení zvuku od látky.

ÚKOL

Rozjedte se dopředu a měřte vzdálenost před robotem. Když je vzdálenost menší než `100 mm` (10 cm), zastavte, aby robot nenaboural.

POSTUP ŘEŠENÍ

```
from brian.motors import EV3LargeMotor, MotorPort
from brian.pilot import Pilot
from brian.sensors import SensorPort
from brian.sensors.EV3 import UltrasonicSensorEV3
from time import sleep

ultrasonic = UltrasonicSensorEV3(SensorPort.S2)

motor_l = EV3LargeMotor(MotorPort.B)
motor_r = EV3LargeMotor(MotorPort.C)
pilot = Pilot(motor_l, motor_r, wheelbase_mm=110, wheel_diameter_mm=40)
pilot.set_speed(200)

pilot.forward()

while ultrasonic.distance_mm() > 100: # Počkej, až senzor uvidí zeď
    sleep(0.1) # Mezitím nic nedělej

pilot.brake() # A poté zastav.
```



ÚKOL

Rozjedte se dopředu a měřte vzdálenost před robotem. Když je vzdálenost menší než 100 mm (10 cm), pootočte robota a objedte překážku.

POSTUP ŘEŠENÍ

```
from brian.motors import EV3LargeMotor, MotorPort
from brian.pilot import Pilot
from brian.sensors import SensorPort
from brian.sensors.EV3 import UltrasonicSensorEV3
from time import sleep

ultrasonic = UltrasonicSensorEV3(SensorPort.S2)

motor_l = EV3LargeMotor(MotorPort.B)
motor_r = EV3LargeMotor(MotorPort.C)
pilot = Pilot(motor_l, motor_r, wheelbase_mm=110, wheel_diameter_mm=40)
pilot.set_speed(200)

pilot.forward()

while True:
    if ultrasonic.distance_mm() < 100:
        pilot.turn(0, 45) # Otoč se na místě o 45° doprava
        pilot.forward()
    sleep(0.1)
```

BAREVNÝ SENZOR

Třetí senzor co si ukážeme je `ColorSensor`.

- Senzor vydává světlo (červené, nebo bílé) směrem dolů či dopředu a poté měří, kolik toho světla se od něčeho odrazí. To je `reflected light` (odražené světlo).
- Může také měřit `ambient light` (okolní světlo), tedy jak je světlý prostor kolem senzoru bez aktivního osvětlení.
- Senzor také umí rozeznat barvy – třeba červenou, zelenou, modrou, černou, bílou a další.



! POZOR

Povrch materiálu hodně ovlivňuje měření – lesklé plochy odrážejí světlo víc, matné méně.

💡 TIP

Umístěte světelný senzor co nejbliž k podlaze, aby nepropustilo moc světla bočními cestami.

ÚKOL

Jedte dopředu, dokud barevný senzor nenajde černou čáru, potom automaticky zastavte.

POSTUP ŘEŠENÍ

```
from brian.motors import EV3LargeMotor, MotorPort
from brian.pilot import Pilot
from brian.sensors import SensorPort
from brian.sensors.EV3 import ColorSensorEV3
from time import sleep

color = ColorSensorEV3(SensorPort.S1)

motor_l = EV3LargeMotor(MotorPort.B)
motor_r = EV3LargeMotor(MotorPort.C)
pilot = Pilot(motor_l, motor_r, wheelbase_mm=110, wheel_diameter_mm=40)
pilot.set_speed(200)

pilot.forward()

while color.reflected_value() > 50: # Dokud nevidíš černou
    sleep(0.1) # Tak nic nedělej

pilot.brake() # A poté zastav.
```

GYROSKOPICKÝ SENZOR

Posledním senzorem který si ukážeme je GyroSensor. Ten slouží k měření otáčení robota.



! VAROVÁNÍ

Pokud GyroSensor, tak je třeba zkalibrovat. V Demo programs -> Sensor view -> Port X -> Modes (tlačítko vpravo) -> Na konci seznamu Reboot (fix drift)

Při kalibraci se s Brianem nesmí hýbat, jinak se na na gyru začne nasčítávat úhel.

ÚKOL

Udělat funkci, která jede s robotem rovně, poté se pomocí gyra otočí o 90°. A tuto funkci zopakovat 4x po sobě.

POSTUP ŘEŠENÍ

```
from brian.motors import EV3LargeMotor, MotorPort
from brian.pilot import Pilot
from brian.sensors import SensorPort
from brian.sensors.EV3 import GyroSensorEV3
from time import sleep

gyro = GyroSensorEV3(SensorPort.S4)

motor_l = EV3LargeMotor(MotorPort.B)
motor_r = EV3LargeMotor(MotorPort.C)
pilot = Pilot(motor_l, motor_r, wheelbase_mm=110, wheel_diameter_mm=40)
pilot.set_speed(200)

def forward_and_rotate():
    pilot.forward(200) # Popojed 20 cm
    sleep(0.5) # Počkej, až robot úplně zastaví
    gyro.set_zero_point() # Začneme měřit úhel od nuly
    pilot.turn(0, turn_rate=90) # Začni se točit na místě doprava
    while gyro.angle() > -90:
        sleep(0.01)
    pilot.brake()

for _ in range(4):
    forward_and_rotate()
```

ÚKOL

Naprogramujte robota, co bude sledovat černou čáru.



NÁPOVĚDA

Robot nemusí sledovat střed čáry. Skvěle se mu bude sledovat hranice čáry, kde uvidí černou na jedné straně a bílou na druhé. Program může mít logiku, pokud vidím černou, zatoč doprava, pokud bílou, zatoč doleva. Robot přitom pořád jede dopředu, jen jedno kolo pojede rychleji než druhé, takže opíše oblouk a nebude se točit na místě.

POSTUP ŘEŠENÍ

```
from brian.motors import EV3LargeMotor, MotorPort
from brian.sensors import SensorPort
from brian.sensors.EV3 import ColorSensorEV3

color = ColorSensorEV3(SensorPort.S1)

motor_l = EV3LargeMotor(MotorPort.B)
motor_r = EV3LargeMotor(MotorPort.C)

SPEED = 300 # Základní rychlost obou kol
TURN = 150 # 0 kolik jedno kolo zrychlíme a druhé zpomalíme

while True:
    if color.reflected_value() > 50: # Vidím bílou, zatáčím doleva
        motor_l.run_at_speed(SPEED - TURN)
        motor_r.run_at_speed(SPEED + TURN)
    else: # Vidím černou, zatáčím doprava
        motor_l.run_at_speed(SPEED + TURN)
        motor_r.run_at_speed(SPEED - TURN)
```

P REGULACE

Už umíme naprogramovat robota tak, aby následoval černou čáru. Akorát vidíme, že robot jezdí zig-zag. I když by měl jet rovně se tak se kroutí. Proto použijeme **P regulaci** (proporcionální regulaci). Je to chytrý způsob, jak robota naučit, že čím víc se odchýlí od čáry, tím víc se musí stočit zpátky. Pokud je odchylka malá, opraví se jen trochu. Pokud velká, zatočí víc.

Jak na to v programu:

1. **Najděte středovou hodnotu** senzoru – mezi černou a bílou (např. průměr).
2. **Změřte aktuální hodnotu** senzoru.
3. **Spočítejte chybu**: $\text{chyba} = \text{aktuální_hodnota} - \text{středová_hodnota}$.
4. **Vynásobte chybu zesílením (K_p)**: $\text{oprava} = K_p * \text{chyba}$.
5. **Upravte rychlost motorů**:
 - levý motor = základní_rychlost – oprava
 - pravý motor = základní_rychlost + oprava

Úkolem bude vyzkoušet různé hodnoty K_p , dokud robot nepojede plynule.



VYZKOUŠEJTE SI

Pozorujte, co se stane, když je K_p příliš malé nebo příliš velké.



NÁPOVĚDA

Středovou hodnotu senzoru můžete naměřit po zapojení barevného senzoru v sekci Demo programs -> Sensor view.

ÚKOL

Naprogramujte robota, co bude sledovat černou čáru pomocí P regulátoru.

POSTUP ŘEŠENÍ

```
from brian.motors import EV3LargeMotor, MotorPort
from brian.sensors import SensorPort
from brian.sensors.EV3 import ColorSensorEV3

color = ColorSensorEV3(SensorPort.S1)

motor_l = EV3LargeMotor(MotorPort.B)
motor_r = EV3LargeMotor(MotorPort.C)

Kp = 4 # Otestovaná hodnota
WHITE = 80 # Změřené hodnoty
BLACK = 0
target = ( WHITE + BLACK ) // 2 # Celočíselné dělení, motory chtějí celá čísla

while True:
    lightValue = color.reflected_value()
    error = lightValue - target
    turn = Kp * error
    motor_l.run_at_speed(200 - turn) # levý = základní rychlost - oprava
    motor_r.run_at_speed(200 + turn) # pravý = základní rychlost + oprava
```

3. KNOB NA BRIANOVÍ

ÚKOL

Udělat výběr čísla pomocí knobu (otáčivého tlačítka) na Brianovi. Pro potvrzení čísla zmáčknete knob.



WARNING

Tato funkce ještě není podporována aplikací BrVis.

POSTUP ŘEŠENÍ

```
import brian.buttons as buttons
from time import sleep

buttons.knob.reset_absolute_rotation() # Začneme počítat od nuly

while not buttons.knob.is_pressed(): # Dokud knob nezmáčkneš
    print(buttons.knob.turned_to()) # Vypiš, kam je knob otočený
    sleep(0.1)

print("hodnota:", buttons.knob.turned_to())
```

6. SIMON SAYS



ÚKOL

Naprogramujte hru Simon Says na kostce. Kostka vždy řekne jaké tlačítka musí hráč ve správném pořadí zmáčknout.

ŘEŠENÍ



ŘEŠENÍ

Řešení této úlohy najdete pouze v Materiálech na <https://gitlab.fel.cvut.cz/rehakj16/brian-prirucka-assets>.



PID REGULACE

P regulace už zvládne držet čáru, ale někdy se robot rozkmitá nebo nejede plynule. Proto přidáme **PID regulaci**:

- **P (proporcionální)** – okamžitá oprava podle chyby.
- **I (integrální)** – sčítá chyby a vrací robota zpět, když je dlouhodobě mimo.
- **D (derivační)** – sleduje rychlost změny chyby, aby se nepřekmital.

Jak na to v programu:

```
1. chyba = aktuální - středová
2. suma_chyb += chyba
3. delta_chyby = chyba - předchozí
4. P = Kp * chyba
   I = Ki * suma_chyb
   D = Kd * delta_chyby
5. oprava = P + I + D
6. levý = základ - oprava
   pravý = základ + oprava
```



TIP

Začněte jen s **P** ($K_i=0$, $K_d=0$).

K_p moc velké → kmity.

K_i moc velké → pomalé „plavání“.

K_d moc velké → trhaná jízda.

S PID regulací jede robot plynuleji a chytřeji.

ŘEŠENÍ



ŘEŠENÍ

Řešení této úlohy najdete pouze v Materiálech na <https://gitlab.fel.cvut.cz/rehakj16/brian-prirucka-assets>.

2. PŘEKÁŽKA NA ČÁŘE



ÚKOL

Naprogramujte robota, co bude sledovat černou čáru, když na čáře bude překážka, objede ji a vrátí se na černou čáru za překážkou.

ŘEŠENÍ



ŘEŠENÍ

Řešení této úlohy najdete pouze v Materiálech na <https://gitlab.fel.cvut.cz/rehakj16/brian-prirucka-assets>.

3. BLUDIŠTĚ



ÚKOL

Naprogramujte robota, který najde cestu z bludiště z kostek.

ŘEŠENÍ



ŘEŠENÍ

Řešení této úlohy najdete pouze v Materiálech na <https://gitlab.fel.cvut.cz/rehakj16/brian-prirucka-assets>.

4. SUMO



ÚKOL

Uspořádejte sumo zápasy robotů. Stačí vyhrazené pole na zemi. Vždy se postaví dva roboti proti sobě a kde zůstane v poli déle, vyhrál.

ŘEŠENÍ



ŘEŠENÍ

Řešení této úlohy najdete pouze v Materiálech na <https://gitlab.fel.cvut.cz/rehakj16/brian-prirucka-assets>.