FAKULTA
ELEKTROTECHNICKÁ
ČVUT V PRAZE

BRIAN
PŘÍRUČKA

JAN ŘEHÁK, ONDŘEJ VAŠÁTKO

VERZE 0.6

OBSAH

ÚVOD

MOTORY

SENZORY

POKROČILÉ ÚLOHY

NEŘEŠENÉ ÚLOHY

ÚKOL

Naskenujte QR kód a podle návodu postav robota.

💡 **TIP**

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

PŘÍPRAVA PROSTŘEDÍ

ÚKOL

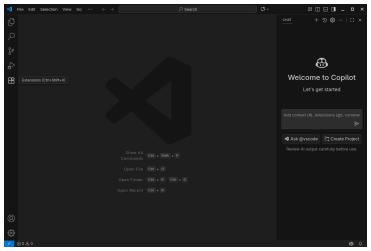
Připravte si programovací prostředí.

TIP

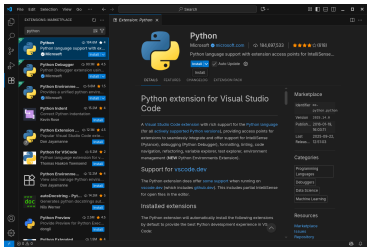
Pro úplné začátky s programováním doporučujeme začít s BriVis

POSTUP ŘEŠENÍ

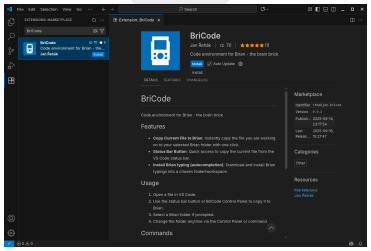
1. Otevřete si visual studio code



2. V záložce extensions stáhněte extension pro `python`



a extension `BriCode`

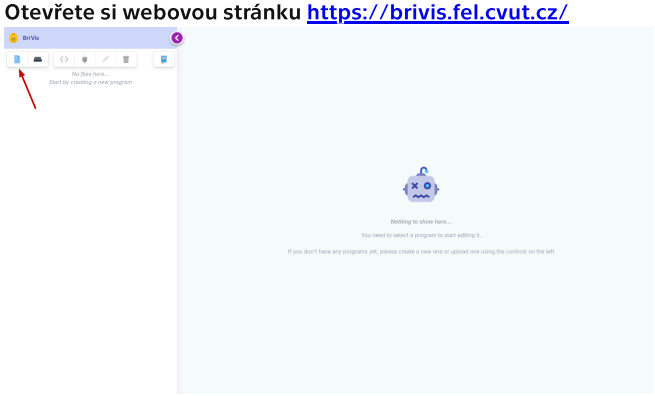


3. Vytvořte si svojí pracovní složku a tu otevřete ve vscode.
4. Pomocí tlačítka `Install Brian typing (autocomplete)` v záložce `BriCode` v levém panelu, stáhněte pomocné soubory pro vývoj.
5. Připojte Briana k počítači pomocí USB a tlačítkem `Select Brian Folder` otevřete připojeného Briana.

TIP

Visual studio code si můžete stáhnout na stránce https://code.visualstudio.com/

Otevřete si webovou stránku https://brivis.fel.cvut.cz/



Tlačítkem vytvořte nový soubor. Po vytvoření souboru, můžete zvesela programovat.

PŘIPOJENÍ PŘES WIFI

ZAPNUTÍ

V hlavním menu jděte do `Settings` → `WiFi config` a vyberte `Enable`.
WiFi se spustí ve zvoleném režimu: `Station mode` nebo `AP mode`.

- `Station mode:` Brian se připojí k existující WiFi síti.
- `AP mode (Access Point):` Brian vytvoří vlastní WiFi síť, do které se mohou připojit jiná zařízení.

Pro použití webové aplikace **BRemote** si poznamenejte „IP address" zobrazenou v menu `WiFi config` a v prohlížeči na zařízení připojeném ke stejné síti otevřete `http://<ip-adresa>`.
Otevře se BRemote s různými funkcemi.

STATION MODE

1. V menu `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, budete vyzváni 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.

AP MODE

1. V menu `WiFi config` zvolte `Configure AP`.
2. Zadejte `network 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.

Start on boot?

- `always start` — WiFi se spustí při každém startu a připojí se / spustí AP.
- `if previously on (default)` — WiFi se spustí jen tehdy, pokud bylo zapnuté před posledním vypnutím.
- `don't autostart` — WiFi se spravuje ručně.

Startup mode?

- `scan & connect (default)` — Neustále hledá dostupné sítě a připojí se k té se nejsilnějším signálem. Pokud Brian opustí dosah sítě, znovu skenuje a připojuje se.
- `fallback to AP (default on)` — Po několika neúspěšných pokusech o připojení Brian spustí vlastní AP. Jinak stále skenuje.
- `only start AP` — Nepokouší se připojit k žádné síti, rovnou spustí AP.

BRemote (WiFi Brian Remote app)

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 aplikace **BRemote**, která 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 otevřou přímo v BriVis (pokud má počítač internet)
- Program start/control: spouštění libovolných spustitelných souborů na Brianovi přímo z BRemote, možnost přerušení nebo ukončení
- Firmware update check: pokud má zařízení internet, BRemote upozorní na novou verzi a nabídne odkaz na aktualizační 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 vypsat text "Hello World" na kostce brian.

⚠️ **POZOR**

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

💡 **TIP**

Pro každou úlohu si doporučujeme vytvořit nový soubor

💡 **NÁPOVĚDA**

Napište `print()` a do závorek napište ve dvojtých uvozovkách co chceš vytisknout

⚠️ **POZOR**

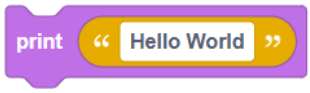
Zkuste si vytisknout i jiné fráze, ale pozor, Brian neumí tisknout české znaky, musíš proto psát bez háčků a čárek

POSTUP ŘEŠENÍ

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

1. Otevřete Visual studio code
2. Připojte kostku pomocí USB-C kabelu
3. V rozšíření BriCode klikněte na tlačítko *Vyber složku Briana* a otevřete uložiště Briana
4. V rozšíření BriCode klikněte na tlačítko *Upload do Briana*
5. Na kostce Brian klikněte na *SD karta*, najděte soubor a vyberte *START*

1. V menu BriVise klikněte na tlačítko Download
2. Překopírujte soubor který se vám stáhl do uložiště kostky Brian
3. Na kostce Brian klikněte na *SD karta*, najděte soubor a vyberte *START*



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.

NÁPOVĚDA

Použijte funkci `play_tone(...)`. Do závorek napište dvě čísla: první určuje výšku tónu a druhé délku tónu v milisekundách (1000ms = 1s).

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`.

NÁPOVĚDA

Použijte blok `začni přehrávat __ Hz tón po dobu __ ms`. Do mezer doplň dvě čísla: první určuje výšku tónu a druhé délku tónu v milisekundách (1000ms = 1s).

NÁPOVĚDA

Po tom, co necháme hrát tón nesmíme předčasně ukončit program než tón dohraje. Použijte tedy blok `počkej __ s`, která chvíli počká, než tón dohraje.

TIP

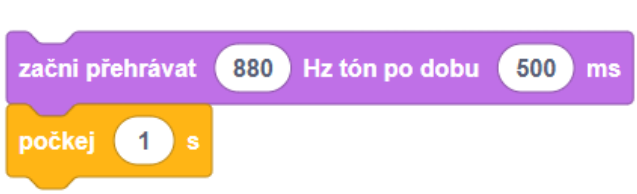
Doporučujeme příjemnou výšku tónu 880, můžete ale experimentovat s dalšími tóny.

POZOR

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

POSTUP ŘEŠENÍ

```python
import brian.audio as audio
from time import sleep

audio.play_tone(880, 500)
sleep(0.5)
```

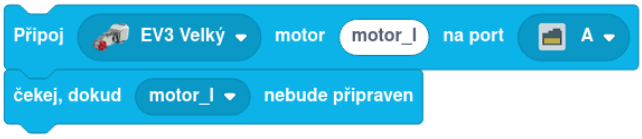


PŘIPOJENÍ MOTORU

Abychom mohli s motorem pracovat, musíme nejdříve pythonu oznámit, že ho budeme používat, poté si ho musíme pojmenovat a nakonec počkat až se připraví.

```
import brian.motors as motors # Oznámí, že
        budeš používat motory

motor_l =
        motors.EV3LargeMotor(motors.MotorPor
        t.A) # Řekne, kam jste motor
        připojil/a a pojmenuje ho

motor_l.wait_until_ready() # Zkontroluje
        jestli je motor připojen a připraven
```

Abychom mohli s motorem pracovat, musíme si ho pojmenovat.



ÚKOL

Naprogramovat jeden motor tak, aby se otočil o 360° rychlostí 300°/s.

NÁPOVĚDA

Řekni motoru A, co má udělat: napište `motor_l`, tečku a poté `rotate_by_angle(...)`.
Do závorek napište dvě čísla: první je, o kolik stupňů se má otočit, druhé je jakou rychlostí.

NÁPOVĚDA

Použijte blok `otoč __ o __ rychlostí __ °/s`

POSTUP ŘEŠENÍ

```
import brian.motors as motors

motor_l =
        motors.EV3LargeMotor(motors.MotorPor
        t.A)
motor_l.wait_until_ready()

motor_l.rotate_by_angle(360, 300) # Otoč o
        360 stupňů, rychlostí 300 stupňů za
        sekundu
```

2. NA PLNÝ PLYN

ÚKOL

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

NÁPOVĚDA

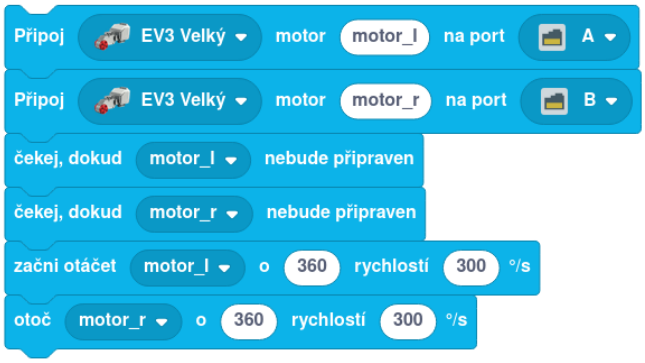
Když nastavíme třetí parametr funkce `rotate_by_angle(...)` na `0` tak se na dokončení funkce nebude čekat a můžeme tedy začít otáčet i druhým motorem. Takže ve výsledku se budou oba motory otáčet zároveň.

NÁPOVĚDA

Pro rozjetí obou motorů najednou, je potřeba pro první motor použít blok `začni otáčet __ o __ rychlostí __ °/s`. Pro druhý motor použít stejný blok jako v předchozím úkolu.

POSTUP ŘEŠENÍ

```python
import brian.motors as motors

# Levý motor
motor_l =
        motors.EV3LargeMotor(motors.MotorPor
        t.A)
motor_l.wait_until_ready()

# Pravý motor
motor_r =
        motors.EV3LargeMotor(motors.MotorPor
        t.B)
motor_r.wait_until_ready()

motor_l.rotate_by_angle(360, 300, 0) # 0 v
        poslením parametru říká, že se nemá
        čekat na dokočení otáčení motoru,
        jelikož chceme zároveň točit i
        druhým motorem
motor_r.rotate_by_angle(360, 300)
```



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

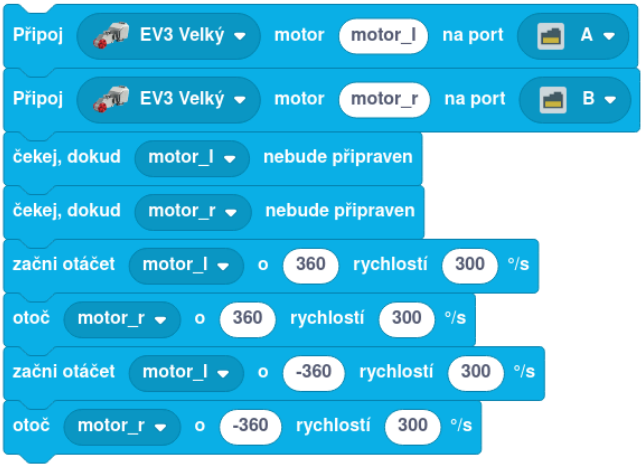
Záporné číslo v prvním parametru funkce `rotate_by_angle(...)` popožene robota dozadu.

NÁPOVĚDA

Záporné číslo v políčku rychlosti popožene robota dozadu.

POSTUP ŘEŠENÍ

```
import brian.motors as motors

motor_l =
        motors.EV3LargeMotor(motors.MotorPor
        t.A)
motor_l.wait_until_ready()

motor_r =
        motors.EV3LargeMotor(motors.MotorPor
        t.B)
motor_r.wait_until_ready()

motor_l.rotate_by_angle(360, 300, 0)
motor_r.rotate_by_angle(360, 300)

motor_l.rotate_by_angle(-360, 300, 0)
motor_r.rotate_by_angle(-360, 300)
```



4. ZATÁČKA

ÚKOL

Naprogramovat robota tak, aby se otočil do pravého úhlu.

NÁPOVĚDA

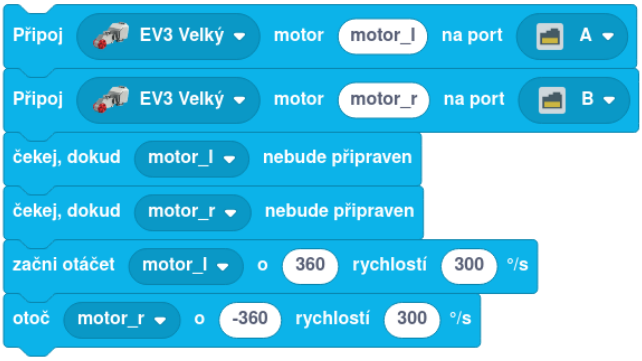
Aby se robot otočil, musíme jedním motorem otočit na jednu stranu a druhým na druhou.

VYZKOUŠEJTE SI

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

POSTUP ŘEŠENÍ

```python
import brian.motors as motors

motor_l =
        motors.EV3LargeMotor(motors.MotorPor
        t.A)
motor_l.wait_until_ready()

motor_r =
        motors.EV3LargeMotor(motors.MotorPor
        t.B)
motor_r.wait_until_ready()

motor_l.rotate_by_angle(360, 300, 0)
motor_r.rotate_by_angle(-360, 300)
```



5. TOČÍM SE TOČÍM

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 podtržítko.

```
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

```python
import brian.motors as motors

motor_l =
        motors.EV3LargeMotor(motors.MotorPor
        t.A)
motor_l.wait_until_ready()

motor_r =
        motors.EV3LargeMotor(motors.MotorPor
        t.B)
motor_r.wait_until_ready()

for _ in range(4): #  Proměnou
        nepotřebujeme, vyplníme jí proto _
        (podtržítkem)
    # Popojedeme dopředu
    motor_l.rotate_by_angle(360, 300, 0)
    motor_r.rotate_by_angle(360, 300)

    # A teď se otočíme
    motor_l.rotate_by_angle(360, 300, 0)
    motor_r.rotate_by_angle(-360, 300)
# A celé to zopakujeme 4x
```

6. SPIRÁLA

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

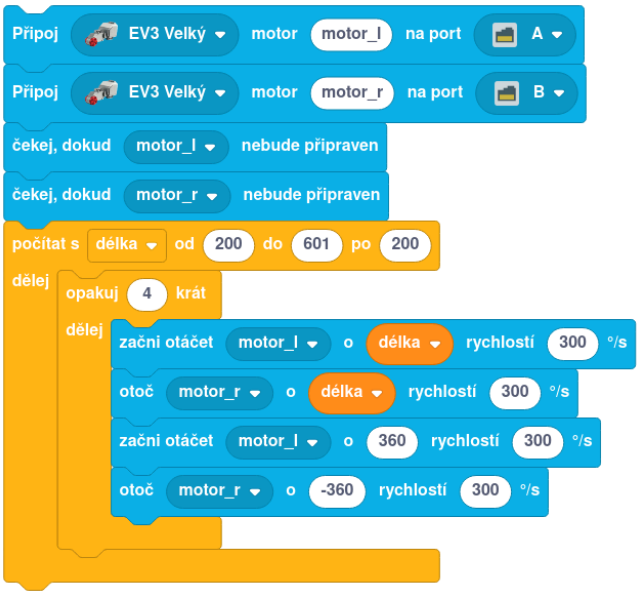
Naprogramovat robota tak, aby se točil do spirály. Použijte vnořený *for cyklus*.

💡 NÁPOVĚDA

Vnější *for cyklus* můžeme napsat třeba jako `for length in range(200, 600, 200):`, budeme tak mít proměnnou délku, která se bude prodlužovat o 200 každou iteraci.

POSTUP ŘEŠENÍ

```python
import brian.motors as motors

motor_l =
        motors.EV3LargeMotor(motors.MotorPor
        t.A)
motor_l.wait_until_ready()

motor_r =
        motors.EV3LargeMotor(motors.MotorPor
        t.B)
motor_r.wait_until_ready()

for length in range(200, 600, 200): # Horní
        limit musí být o jedna větší než
        požadovaná poslední hodnota v cyklu,
        rozmysli si proč
    for _ in range(4):
        motor_l.rotate_by_angle(length, 300,
        0)
        motor_r.rotate_by_angle(length, 300)

        motor_l.rotate_by_angle(360, 300, 0)
        motor_r.rotate_by_angle(-360, 300)
```



1. DETEKCE STISKU

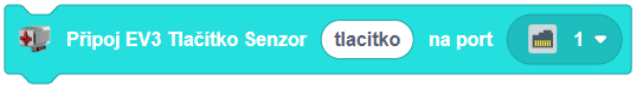
TLAČÍTKO

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, poté si ho musíme pojmenovat a nakonec počkat až se připraví. Stejně jak u motoru.



⚠ UPOZORNĚNÍ

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

```
import brian.sensors as sensors # Oznámí, že
        budeš používat senzory

button =
        sensors.EV3.TouchSensorEV3(sensors.S
        ensorPort.S3) # Řekne, kam jste
        senzor připojil/a a pojmenuje ho

button.wait_until_ready() # Zkontroluje
        jestli je senzor připojen a
        připraven
```

ÚKOL

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

💡 NÁPOVĚDA

Použijte funkci `wait_for_press()`

💡 NÁPOVĚDA

Použijte funkci blok `čekej než se __ zmáčkne`

POSTUP ŘEŠENÍ

```python
import brian.sensors as sensors

button =
        sensors.EV3.TouchSensorEV3(sensors.S
        ensorPort.S1)
button.wait_until_ready()

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

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.

```
import brian.sensors as sensors
import brian.motors as motors

button = sensors.EV3.TouchSensorEV3(sensors.SensorPort.S1)

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.

```
import brian.sensors as sensors
import brian.motors as motors

button = sensors.EV3.TouchSensorEV3(sensors.SensorPort.S1)

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 nedojde
```

📝 **VYZKOUŠEJTE SI**

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

📝 **VYZKOUŠEJTE SI**

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

ÚKOL

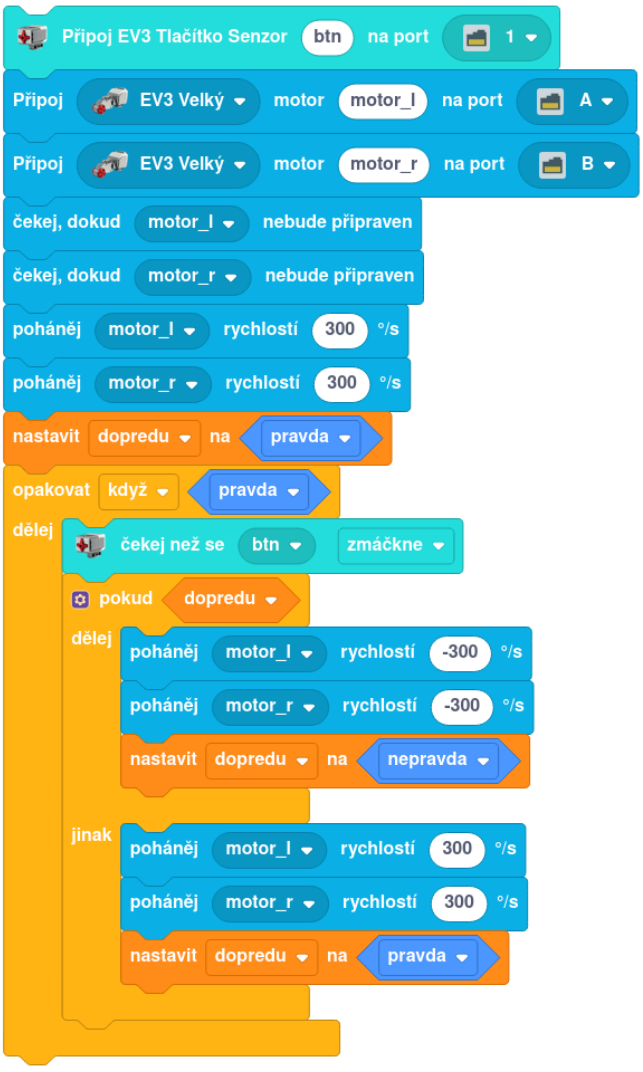
Rozjeďte se dopředu a na zmáčknutí tlačítka otočte směr jízdy.

NÁPOVĚDA

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

POSTUP ŘEŠENÍ

```
import brian.sensors as sensors
import brian.motors as motors

button =
        sensors.EV3.TouchSensorEV3(sensors.S
        ensorPort.S1)

motor_l =
        motors.EV3LargeMotor(motors.MotorPor
        t.A)
motor_l.wait_until_ready()
motor_r =
        motors.EV3LargeMotor(motors.MotorPor
        t.B)
motor_r.wait_until_ready()

motor_l.run_at_speed(300)
motor_r.run_at_speed(300)
forward = True

while (True):
    button.wait_for_press()
    if forward:
        motor_l.run_at_speed(-300) # Jeďte
         dozadu
        motor_r.run_at_speed(-300)
        forward = False
    else:
        motor_l.run_at_speed(300) # Jeďte
         dopředu
        motor_r.run_at_speed(300)
        forward = True
```



TIP

Použijte funkci `wait_for_press()` (v BriVisu `čekej než se ___ zmáčkne`)

3. ZPOMALIT NEBO ZRYCHLIT

ÚKOL

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

⚠️ **POZOR**

Rychlost motorů je omezená a u velkého motoru by neměla jít níže než `0` nebo výše než `1050 deg/s`. U středních motorů je maximum `1560 deg/s`

⚠️ **POZOR**

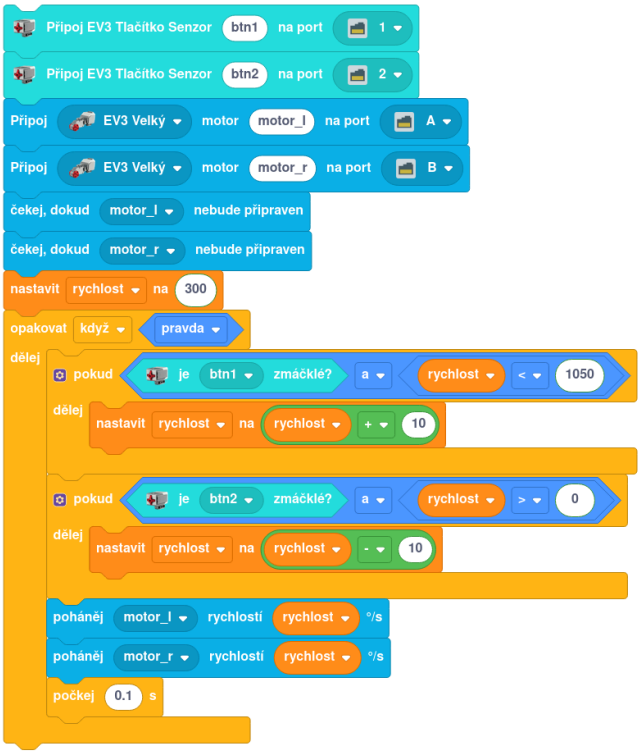
Ve smyčce použijte `sleep(0.1)`, aby se jedno stisknutí tlačítka nezaznamenalo jako několik naráz. Nezapomeňte na začátku programu přidat `from time import sleep`

💡 **NÁPOVĚDA**

Budete potřebovat proměnnou, která si bude pamatovat rychlost jízdy.

POSTUP ŘEŠENÍ

```python
import brian.sensors as sensors
import brian.motors as motors
from time import sleep

# Velkými písmeny si hned na začátku určím
        konstanty (neměnné hodnoty)
LOWER_LIMIT = 0
UPPER_LIMIT = 1050 # Pro střední motor je to
        1560 deg/s

button1 =
        sensors.EV3.TouchSensorEV3(sensors.S
        ensorPort.S1)
button2 =
        sensors.EV3.TouchSensorEV3(sensors.S
        ensorPort.S2)

motor_l =
        motors.EV3LargeMotor(motors.MotorPor
        t.A)
motor_l.wait_until_ready()
motor_r =
        motors.EV3LargeMotor(motors.MotorPor
        t.B)
motor_r.wait_until_ready()

speed = 300
while (True):
    if button1.is_pressed() and speed <
        UPPER_LIMIT:
        speed += 10 # Přičtu 10 k rychlosti
    if button2.is_pressed() and speed >
        LOWER_LIMIT:
        speed -= 10 # Odečtu 10 od rychlosti
    motor_l.run_at_speed(speed) # Rozjedu
        motory novou rychlostí
    motor_r.run_at_speed(speed)
    sleep(0.1)
```



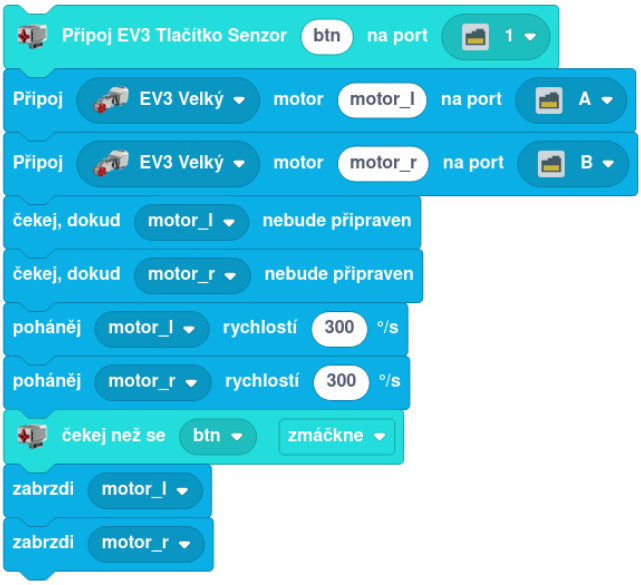
4. ZASTAVIT, STÁT!

ÚKOL

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

POSTUP ŘEŠENÍ

```python
import brian.sensors as sensors
import brian.motors as motors

button =
        sensors.EV3.TouchSensorEV3(sensors.S
        ensorPort.S1)

motor_l =
        motors.EV3LargeMotor(motors.MotorPor
        t.A)
motor_l.wait_until_ready()
motor_r =
        motors.EV3LargeMotor(motors.MotorPor
        t.B)
motor_r.wait_until_ready()

motor_l.run_at_speed(300)
motor_r.run_at_speed(300)

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

motor_l.brake()
motor_r.brake()
```



ULTRASONICKÝ SENZOR

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

Ten pomocí zvuku a jeho zpětného odražení dokáže měřit vzdálenost.

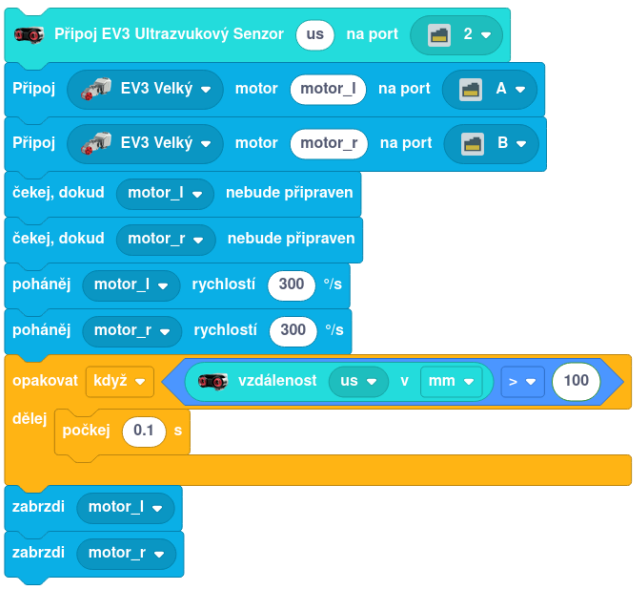
> ⚠️ **POZOR**
>
> Protože senzor používá zvuk, nemusí dávat ideální výsledky při odrážení zvuku od látky.

ÚKOL

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

POSTUP ŘEŠENÍ

```python
import brian.sensors as sensors
import brian.motors as motors
from time import sleep

ultrasonic =
        sensors.EV3.UltrasonicSensorEV3(sens
        ors.SensorPort.S2)

motor_l =
        motors.EV3LargeMotor(motors.MotorPor
        t.A)
motor_l.wait_until_ready()
motor_r =
        motors.EV3LargeMotor(motors.MotorPor
        t.B)
motor_r.wait_until_ready()

motor_l.run_at_speed(300)
motor_r.run_at_speed(300)

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

motor_l.brake() # A poté zastavte.
motor_r.brake()
```



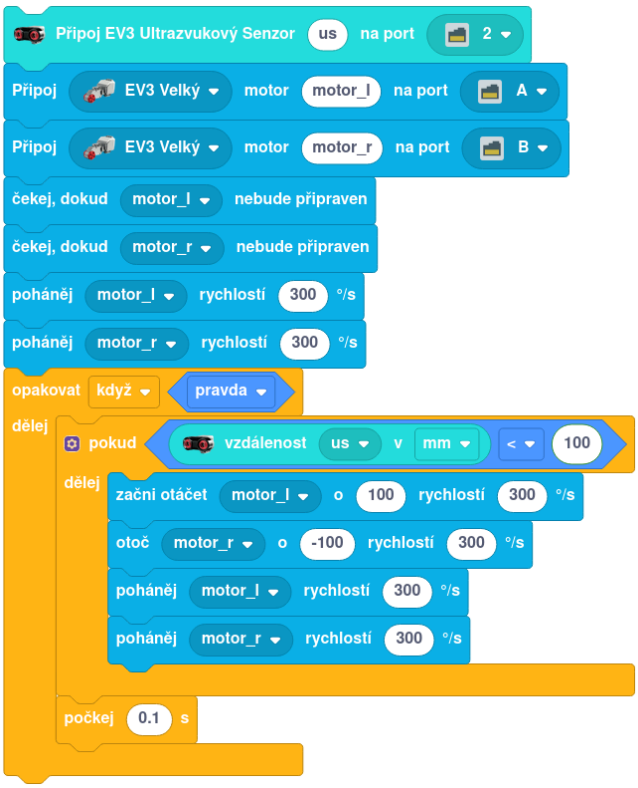
6. NENABOUREJ 2

ÚKOL

Rozjeďte se dopředu a měřte vzdálenost před robotem. Když je vzdálenost menší než `100 mm` (10 cm) pootočte robota a objeďte překážku.

POSTUP ŘEŠENÍ

```python
import brian.sensors as sensors
import brian.motors as motors
from time import sleep

ultrasonic =
        sensors.EV3.UltrasonicSensorEV3(sens
        ors.SensorPort.S2)

motor_l =
        motors.EV3LargeMotor(motors.MotorPor
        t.A)
motor_l.wait_until_ready()
motor_r =
        motors.EV3LargeMotor(motors.MotorPor
        t.B)
motor_r.wait_until_ready()

motor_l.run_at_speed(300)
motor_r.run_at_speed(300)

while True:
    if ultrasonic.distance_mm() < 100:
        motor_l.rotate_by_angle(100, 300, 0)
        motor_r.rotate_by_angle(-100, 300)

        motor_l.run_at_speed(300)
        motor_r.run_at_speed(300)
    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 odráží světlo víc, matné méně.

💡 **TIP**

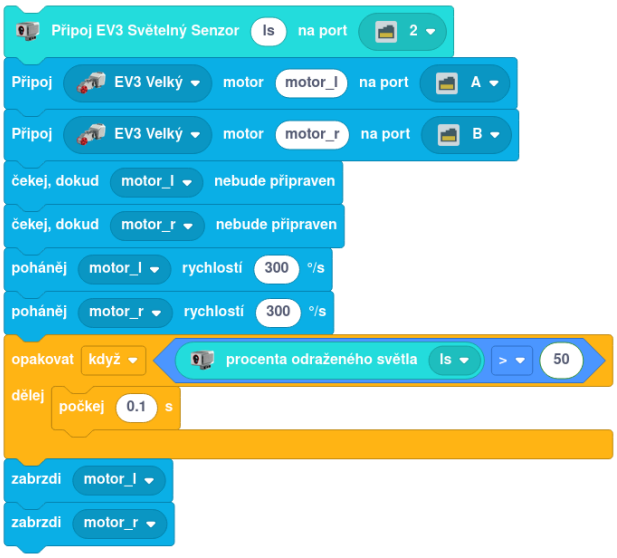
Umístěte světelný senzor co nejblíž k podlaze, aby nepropustilo moc světla bočními cestami.

ÚKOL

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

POSTUP ŘEŠENÍ

```python
import brian.sensors as sensors
import brian.motors as motors
from time import sleep

color =
        sensors.EV3.ColorSensorEV3(sensors.S
        ensorPort.S2)

motor_l =
        motors.EV3LargeMotor(motors.MotorPor
        t.A)
motor_l.wait_until_ready()
motor_r =
        motors.EV3LargeMotor(motors.MotorPor
        t.B)
motor_r.wait_until_ready()

motor_l.run_at_speed(300)
motor_r.run_at_speed(300)

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

motor_l.brake() # A poté zastavte.
motor_r.brake()
```



8. GYRO SENZOR

GYROSKOPICKÝ SENZOR

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

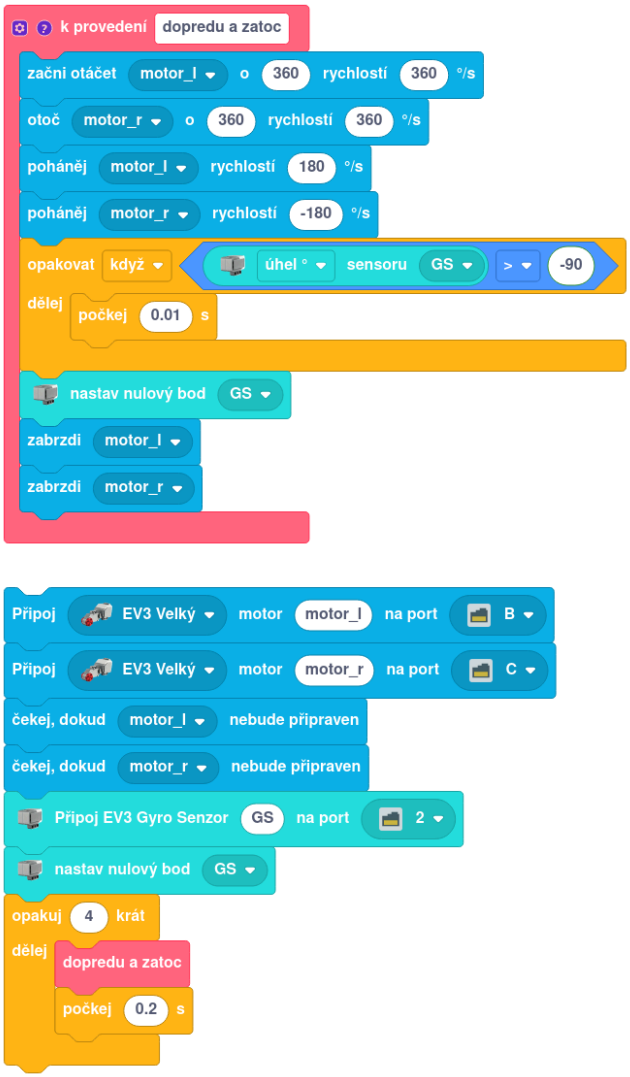
> ⚠️ **VAROVÁNÍ**
>
> **Pokud `GyroSensor`, tak je třeba zkalibrovat. V `Demo programs` -> `Sensor view` -> `Port X` -> `Modes` (tlačítko vpravo) -> Na konci seznamu `Reboot (fix drift)`**
> **Při kalibraci se s Brianem nesmí hýbat, jinak se na na gyru začne nasčítávat úhel.**

ÚKOL

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

POSTUP ŘEŠENÍ

```python
import brian.sensors as sensors
import brian.motors as motors
from time import sleep

gyro =
        sensors.EV3.GyroSensorEV3(sensors.Se
        nsorPort.S1)
gyro.wait_until_ready()

motor_l =
        motors.EV3LargeMotor(motors.MotorPor
        t.B)
motor_r =
        motors.EV3LargeMotor(motors.MotorPor
        t.C)
motor_l.wait_until_ready()
motor_r.wait_until_ready()


def forward_and_rotate():
    motor_l.rotate_by_angle(360, 360, 0)
    motor_r.rotate_by_angle(360, 360)
    motor_l.run_at_speed(180)
    motor_r.run_at_speed(-180)
    while gyro.angle() > -90:
        sleep(0.01)
    gyro.set_zero_point()
    motor_l.brake()
    motor_r.brake()


for _ in range(4):
    forward_and_rotate()
    sleep(0.2)
```



1. SLEDOVÁNÍ ČERNÉ ČÁRY

ÚKOL

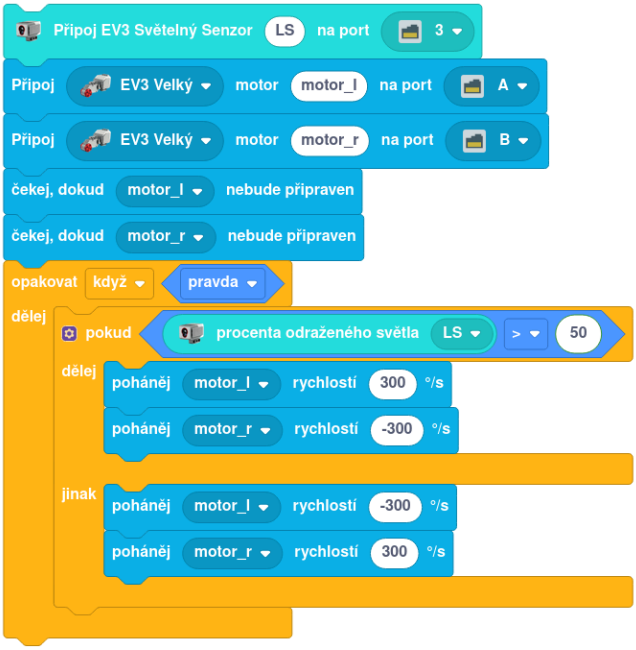
Naprogramujte robota, co bude sledovat černou čáru.

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.

POSTUP ŘEŠENÍ

```python
import brian.sensors as sensors
import brian.motors as motors

color =
        sensors.EV3.ColorSensorEV3(sensors.S
        ensorPort.S3)

motor_l =
        motors.EV3LargeMotor(motors.MotorPor
        t.A)
motor_l.wait_until_ready()
motor_r =
        motors.EV3LargeMotor(motors.MotorPor
        t.B)
motor_r.wait_until_ready()

while True:
    if color.reflected_value() > 50: # Vidím
        bílou
        motor_l.run_at_speed(300)
        motor_r.run_at_speed(-300) # Zatáčím
        doleva
    else: # Vidím černou
        motor_l.run_at_speed(-300)
        motor_r.run_at_speed(300) # Zatáčím
        doprava
```



2. SLEDOVÁNÍ ČERNÉ ČÁRY 2

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**: `chyba = aktuální_hodnota - středová_hodnota`.
4. **Vynásobte chybu zesílením (Kp)**: `oprava = Kp * chyba`.
5. **Upravte rychlost motorů**:
   - levý motor = základní_rychlost – oprava
   - pravý motor = základní_rychlost + oprava

Úkolem bude vyzkoušet různé hodnoty `Kp`, dokud robot nepojede plynule.

📝 VYZKOUŠEJTE SI

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

💡 NÁPOVĚDA

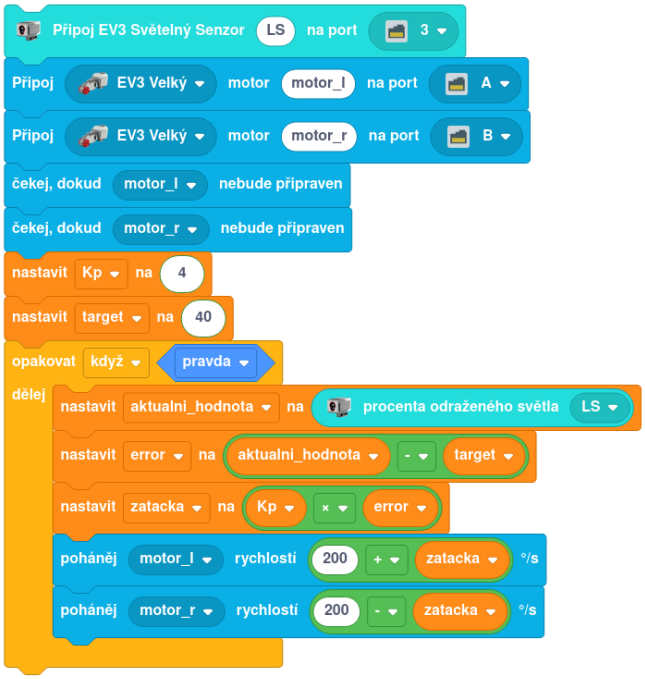
Středovou hodnotu senzoru můžete naměřit po zapojení barevného senzoru v sekci `Demo programs` -> `Sensor view`.

ÚKOL

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

POSTUP ŘEŠENÍ

```python
import brian.sensors as sensors
import brian.motors as motors

color =
        sensors.EV3.ColorSensorEV3(sensors.S
        ensorPort.S3)

motor_l =
        motors.EV3LargeMotor(motors.MotorPor
        t.A)
motor_l.wait_until_ready()
motor_r =
        motors.EV3LargeMotor(motors.MotorPor
        t.B)
motor_r.wait_until_ready()

Kp = 4 # Otestovaná hodnota
WHITE = 80 # Změřené hodnoty
BLACK = 0
target = ( WHITE + BLACK ) / 2

while True:
    lightValue = color.reflected_value()
    error = lightValue - target
    turn = Kp * error
    motor_l.run_at_speed(200 + turn)
    motor_r.run_at_speed(200 - turn)
```



3. KNOB NA BRIANOVI

ÚKOL

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

> ⚠️ **WARNING**
>
> Tato funkce ještě není podporována aplikací BriVis.

POSTUP ŘEŠENÍ

```python
import brian.uicontrol as ui

listener = ui.UiEventsListener()

value = 0

while not listener.knob.last_button_event.is_pressed:
    value += listener.knob.last_button_event.turn_delta # přičte hodnotu otočení knobu k value
    print(value)

print("hodnota:", value)
```

6. SIMON SAYS

ÚKOL

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

ŘEŠENÍ 1/3

```python
import brian.uicontrol as ui
import brian.audio as audio
import brian.settings as settings
import random
from time import sleep

listener = ui.UiEventsListener()
settings.enable_os_button_sounds(False)
ui.set_button_led(listener.knob, ui.LedButtonAnimation.OFF, ui.LedColor(0, 0, 0))

TONE_DURATION = 500

buttons = [
    listener.top_left_button,
    listener.top_right_button,
    listener.bottom_left_button,
    listener.bottom_right_button,
]

button_tones = [1046.50, 1318.51, 1568.00, 2093.00]  # C6, E6, G6, C7
button_colors = [
    ui.LedColor(0, 0, 255),
    ui.LedColor(0, 255, 0),
    ui.LedColor(255, 0, 0),
    ui.LedColor(255, 255, 0),
]
def play_button_sequence(button_sequence):
    set_color_to_buttons(ui.LedButtonAnimation.OFF, ui.LedColor(0, 0, 0))
    for button_id in button_sequence:
        ui.set_button_led(
            buttons[button_id], ui.LedButtonAnimation.SELECTABLE, button_colors[button_id]
        )
        audio.play_tone(button_tones[button_id], TONE_DURATION)
        sleep(0.5)
        ui.set_button_led(
            buttons[button_id], ui.LedButtonAnimation.OFF, ui.LedColor(0, 0, 0)
        )
        sleep(0.2)

def set_color_to_buttons(animation, color=ui.LedColor(255, 255, 255)):
    for button_id in range(4):
        ui.set_button_led(buttons[button_id], animation, color)
```

ŘEŠENÍ 2/3

```
def game_over():
    for button_id in range(4):
        ui.set_button_led(
            buttons[button_id], ui.LedButtonAnimation.SELECTABLE, ui.LedColor(255, 0, 0)
        )
        ui.set_button_led(
            listener.knob, ui.LedButtonAnimation.SELECTABLE, ui.LedColor(255, 0, 0)
        )
    audio.play_tone(440, TONE_DURATION)  # A4
    sleep(4)



random.seed()
score = 0
print("PAMATOVACI HRA!")
print("Sleduj barvy a zopakuj je!")
print("Zmackni libovolne tlacitko pro start hry")
listener.any_button_incl_knob.wait_for_press()
sleep(1)
button_sequence = []
over = False

while not over:
    button_sequence.append(random.randrange(0, 4))
    play_button_sequence(button_sequence)
    print("")
    print("Ted jsi na rade ty!")
    set_color_to_buttons(ui.LedButtonAnimation.SELECTABLE)
    for expected_button in button_sequence:
        listener.any_button.wait_for_press()
        set_color_to_buttons(ui.LedButtonAnimation.SELECTABLE)
        buttons_event = listener.buttons_event_since_last()
        pressed_button = -1

        if buttons_event.top_left.just_pressed:
            pressed_button = 0
        elif buttons_event.top_right.just_pressed:
            pressed_button = 1
        elif buttons_event.bottom_left.just_pressed:
            pressed_button = 2
        elif buttons_event.bottom_right.just_pressed:
            pressed_button = 3

        if not pressed_button == -1:
            audio.play_tone(button_tones[pressed_button], TONE_DURATION)
            ui.set_button_led(
                buttons[pressed_button],
                ui.LedButtonAnimation.SELECTABLE,
                button_colors[pressed_button],
            )

        if not pressed_button == expected_button:
            over = True
            break
```

ŘEŠENÍ 3/3

```
    if not over:
        sleep(TONE_DURATION / 1000)
        score += 1
        set_color_to_buttons(ui.LedButtonAnimation.OFF)
        print("")
        print("Ted ti ukazu co dal")
        sleep(1)

print("")
print("KONEC HRY!")
print(f"Zapamatoval sis {score} barev!")

for flash in range(4):
    for button_id in range(4):
        ui.set_button_led(
            buttons[button_id], ui.LedButtonAnimation.SELECTED, ui.LedColor(255, 0, 0)
        )
    ui.set_button_led(
        listener.knob, ui.LedButtonAnimation.SELECTED, ui.LedColor(255, 0, 0)
    )
    audio.play_tone(300, 200)
    sleep(0.2)

    set_color_to_buttons(ui.LedButtonAnimation.OFF, ui.LedColor(0, 0, 0))
    ui.set_button_led(listener.knob, ui.LedButtonAnimation.OFF)
    sleep(0.2)

print(f"Tvoje skore je: {score}")
```

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**
>
> Začni jen s **P** (Ki=0, Kd=0).
> `Kp` moc velké → kmity.
> `Ki` moc velké → pomalé „plavání".
> `Kd` moc velké → trhaná jízda.

S PID regulací jede robot plynuleji a chytřeji.

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.

3. BLUDIŠTĚ

ÚKOL

Naprogramujte robota, který najde cestu z bludiště z kostek.

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.