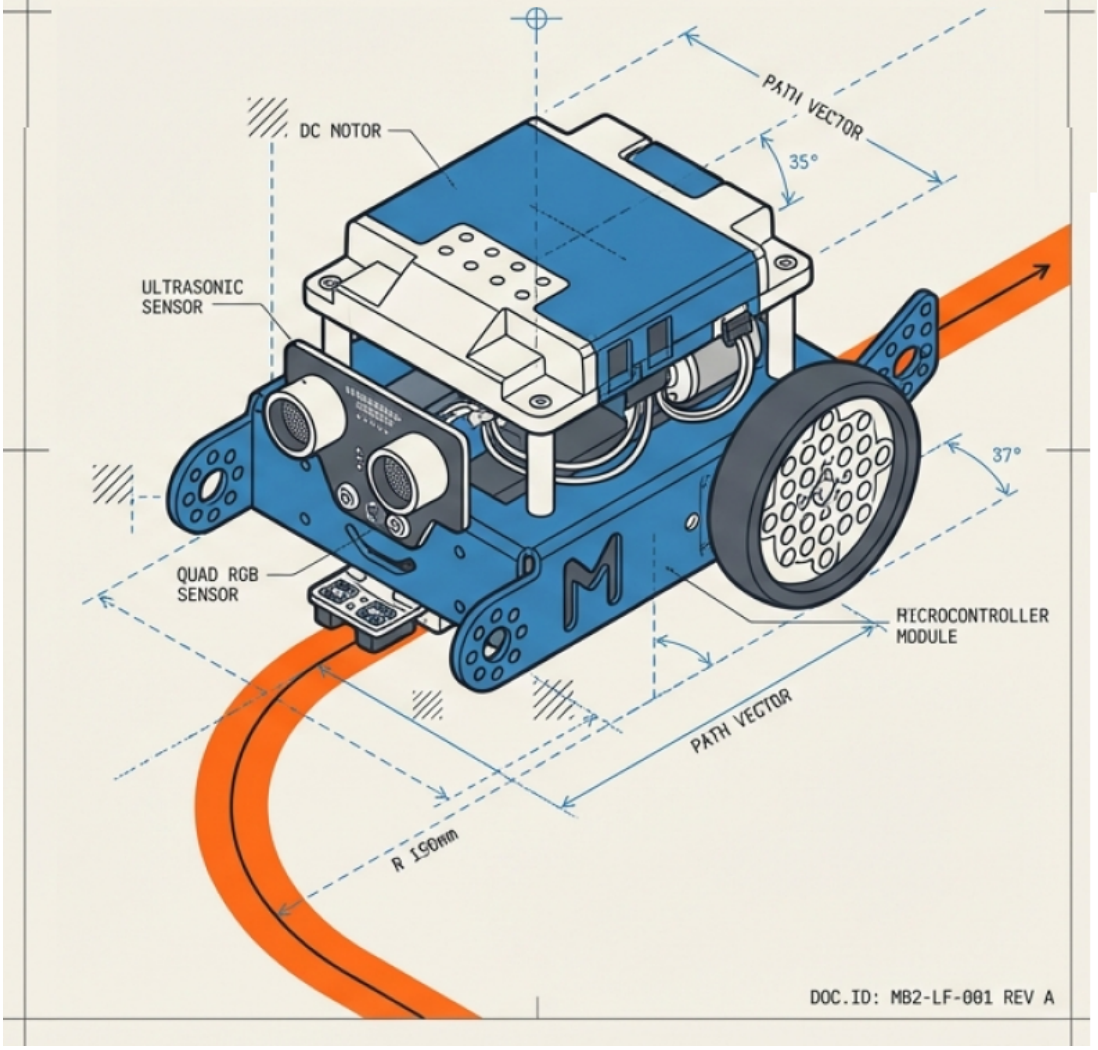
Mastering Line Following

From basic binary logic to mathematical proportional control for the mBot2.



1 Demo

Ako robot dokáže sledovať čiaru najlepšie vyskúšate vstavaným programom číslo 2, ktorý je v robotovi mBot2 už od výroby.

Ako spustiť vstavané programy:

1. Zapnite robota.
2. Vstúpte do menu: Ak nie ste v hlavnom menu, stlačte tlačidlo Home pod joystickom.
3. Pohybom joysticku nadol prejdite na možnosť Switch Program a potvrďte stlačením joysticku.
4. Položte robota na čiaru.
5. Vyberte požadovaný program (Program 2) a stlačte tlačidlo B na spustenie.

Ak robot nesleduje čiaru dobre, treba senzor skalibrovať.

2 Kalibrácia

Kalibrovať senzor sa dá najrýchlejšie bez počítača. Budeš potrebovať rovné miesto s bežným osvetlením pri ktorom budeš aj naďalej pracovať a biely papier s čiernou čiarou, napr. plagát k stavebnici mBot2.

Postup:

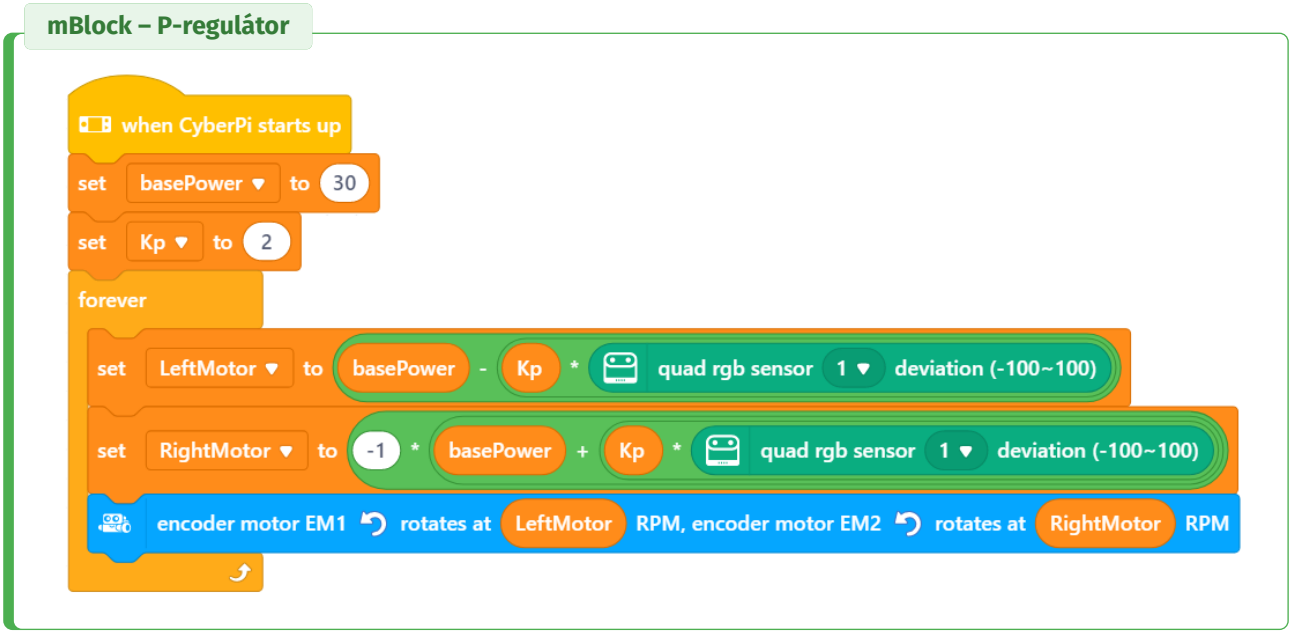
1. Polož robota na **bielu plochu** papiera.
2. Skontroluj, či je senzor vo výške **12–13 mm** nad podložkou.
3. **Dvakrát rýchlo stlač** tlačidlo na senzore.
4. LED diódy začnú blikať rôznymi farbami – kalibrácia prebieha.
5. Pomaly posúvaj robota tak, aby senzor **prešiel ponad čiernu čiaru aj späť** na bielu.
6. Keď LED diódy prestanú blikať, kalibrácia je hotová.

Takto ste senzor nakalibrovali tak, aby rozlišoval čiaru od pozadia.

3 Linefollower

Takto vyzerá program na riadenie pohybu po čiare, ktorý meria odchýlku robota od stredu čiary. Uvádzame program v blokovom jazyku aj v Pythone.

mBlock – P-regulátor



Python – P-regulátor

```
import cyberpi, mbuild

basePower = 30
Kp = 0.5
LeftMotor = 0
RightMotor = 0

while True:
    LeftMotor = (basePower - Kp * mbuild.quad_rgb_sensor.get_offset_track(1))
    RightMotor = -1 * ((basePower + Kp * mbuild.quad_rgb_sensor.get_offset_track
        (1)))
    mbot2.drive_speed(LeftMotor, RightMotor)
```

Ak chcete naozaj porozumieť celej problematike a využiť tento algoritmus vo svojom projekte, odporúčame prejsť podrobne cez všetky nasledujúce pracovné listy.

Poznámka

Tip: pri práci s robotom môže byť dosť otravné stále pripájať a odpájať robota k USB kábliku. Odporúčame použiť Bluetooth kľúč a spojenie nadviazať bezdrôtovo. Makeblock ponúka špeciálny USB 2.0 Bluetooth Adapter, Bluetooth Dongle for PC Connectivity.

Meno: _____ **Trieda:** _____ **Dátum:** _____

4 Opis senzora

Quad RGB senzor je štvorkanálový senzor na detekciu farieb alebo čiary, ktorý sa nachádza na robote mBot2. Štyri senzory sú označené ako L1, L2, R1 a R2 (L znamená ľavú stranu, R pravú stranu). Senzor detekuje hodnoty odrazenej farby a vnútorne porovnávajú kombináciu týchto hodnôt s rôznymi prednastavenými farbami. Senzor dokáže detekovať 6 rôznych farieb plus čiernu a bielu. Vďaka senzoru môže robot mBot2 sledovať čiaru na podlahe a reagovať na rôzne farebné značky. Senzor je umiestnený na prednej strane robota. Na spodnej strane uvidíte štyri senzory. Malé tlačidlo na hornej strane senzora slúži na kalibráciu.

Pri programovaní robota mBot2 budete používať tento senzor na sledovanie farieb na ceste. Program môžete nastaviť tak, aby mBot2 napríklad zastavil na červenej a odbočil vľavo pri zelenej.

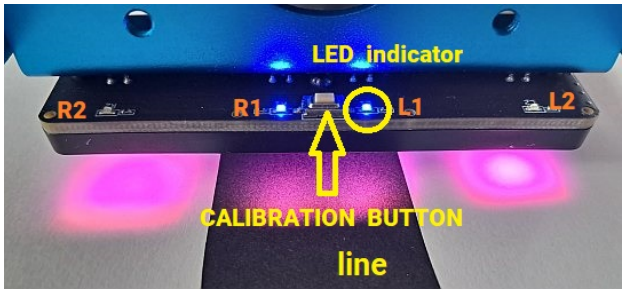
Quad RGB senzor má **4 snímače** číslované 1–4 pri pohľade zhora:

port 4 (L2) port 3 (L1) port 2 (R1) port 1 (R2)

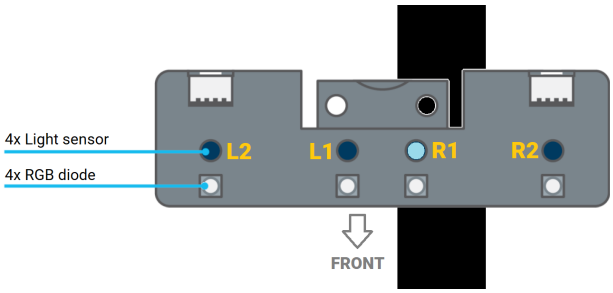
● ● ● ●

Na senzore je aj **tlačidlo** a **farebné LED diódy** (fill lights), ktoré osvetľujú podložku viditeľným svetlom.

4.1. Fyzické usporiadanie



Obr. 1. Senzor na robotovi, modrá LED indikuje čiaru



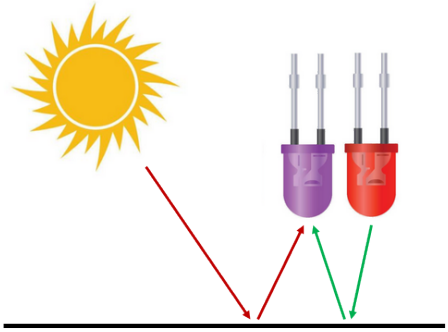
Obr. 2. Pohľad zdola – označenie snímačov

5 Prečo kalibrácia?

Tento RGB senzor meria intenzitu odrazeného svetla. Problém je však v tom, že **čierna** a **biela** nie sú striktne definované hodnoty, pretože v skutočnosti závisia od:

- **osvetlenia miestnosti** (slnečný deň vs. zamračené počasie),
- **materiálu podložky a druhu farby** (matný papier vs. lesklá fólia),
- **výšky senzora** nad podložkou,
- **farby osvetľovacích LED** na senzore.

Na obrázku vpravo vidno, že okrem odrazu svetla ktoré chceme merať (zelené šipky) senzor ruší okolité svetlo (červené šípky) zo slnka alebo lámp.



Obr. 3. Vplyv osvetlenia na hodnoty senzora

Poznámka

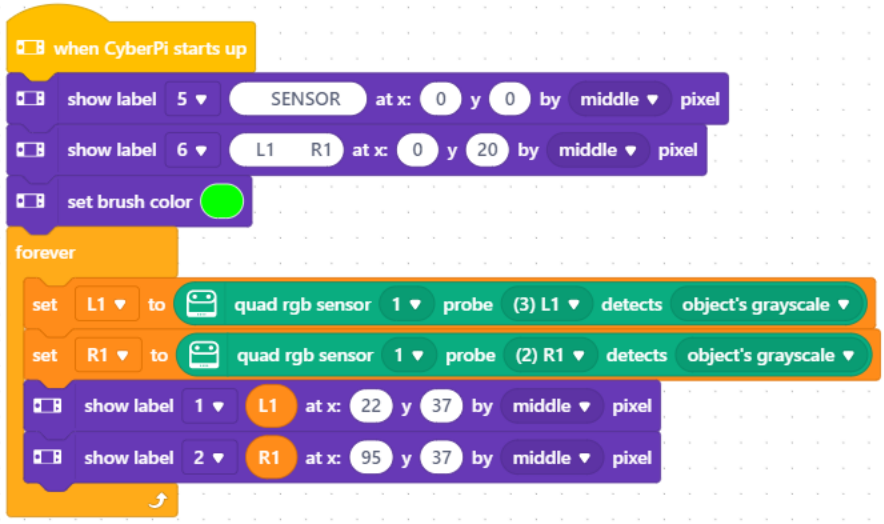
Bez kalibrácie môže senzor v jednej triede fungovať perfektne a v inej miestnosti zlyhať – aj keď kód je identický.

Robot fungoval dobre na hodine, ale doma na rovnakom programe čiaru nesledoval. Aké 2 príčiny by to mohli spôsobiť?

6 Porovnanie pred a po kalibrácii

Ak sa ponáhľate, túto časť môžete preskočiť. Ak však chcete celému procesu naozaj dobre porozumieť, je dobré sledovať, ako sa hodnoty pred a po kalibrácii odlišujú. Skúste preto nahrať do robota nasledovný program a spravte jednoduché meranie pred kalibráciou. Na displeji zobrazíme veľkosť odrazeného svetla, ktoré zmerajú dva prostredné senzory L1 a R1.

Blockly – experiment so senzorom



Python – experiment so senzorom

```
import cyberpi, mbuild

L1 = 0
R1 = 0

cyberpi.display.show_label("      SENSOR   ",16,0,0,1)
cyberpi.display.show_label("   L1       R1",16,0,20,2)
cyberpi.display.set_brush(255, 255, 0)
while True:
  L1 = mbuild.quad_rgb_sensor.get_gray("L1",1)
  R1 = mbuild.quad_rgb_sensor.get_gray("R1",1)
  cyberpi.display.show_label(L1,16,22,37,3)
  cyberpi.display.show_label(R1,16,95,37,4)
```

6.1. Experiment

Nastavte robota nad biely papier a nad čiaru a zapíšte si hodnoty do nasledovnej tabuľky.

	Pred kalibráciou		Po kalibrácii	
	L1	R1	L1	R1
Hodnota na bielej ploche				
Hodnota na čiernej čiare				
Rozdiel hodnôt biela–čierna				
Robot sleduje čiaru?				

Vráťte sa k tabuľke po **kalibrácii** (alebo merajte pri inom osvetlení) a zaznamenajte nové hodnoty, potom ich porovnajte.

7 Postup kalibrácie – metóda A (tlačidlo)

Toto je najrýchlejší spôsob – bez počítača. Budeš potrebovať rovné miesto s bežným osvetlením pri ktorom budeš aj naďalej pracovať a biely papier s čiernou čiarou, napr. plagát k stavebnici mBot2.

Postup:

1. Polož robota na **bielu plochu** papiera.
2. Skontroluj, či je senzor vo výške **12–13 mm** nad podložkou.
3. **Dvakrát rýchlo stlač** tlačidlo na senzore.
4. LED diódy začnú blikať rôznymi farbami – kalibrácia prebieha.
5. Pomaly posúvaj robota tak, aby senzor **prešiel ponad čiernu čiaru aj späť** na bielu.
6. Keď LED diódy prestanú blikať, kalibrácia je hotová.

Takto ste senzor nakalibrovali tak, aby rozlišoval čiaru od pozadia.

Pozor

Nekalibruj pri priamom slnečnom svetle ani pri veľmi slabom osvetlení – výsledky budú nepresné.

8 Postup kalibrácie – metóda B (program)

Kalibráciu možno spustiť aj z tvojho programu. Je to užitočné, ak chceš kalibráciu opakovať automaticky alebo overiť cez mBlock.

Python – spustenie kalibrácie

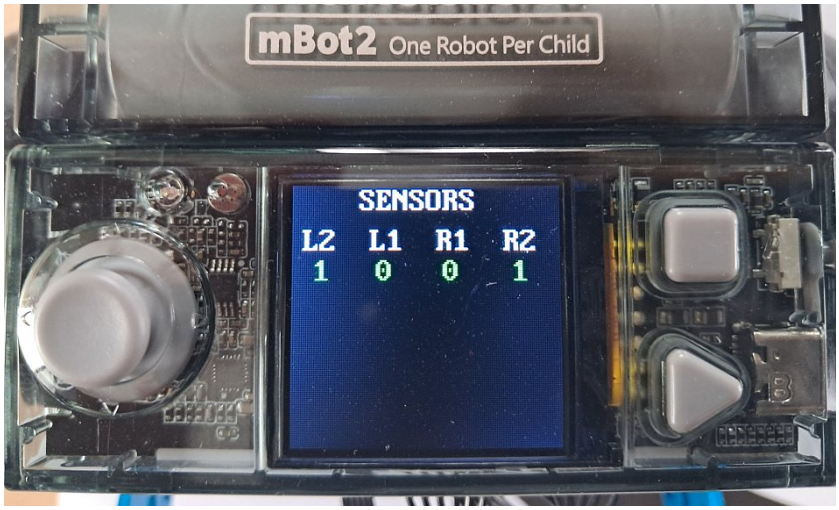
```
import cyberpi, mbuild

mbuild.quad_rgb_sensor.adjust(1)
cyberpi.display.show_label("Hotovo", 12, "center", index= 0)
```

9 Overenie kalibrácie – experiment

Po kalibrácii si môžeme overiť, že senzor správne rozlišuje čiernu a bielu. Nahraj tento program a prechádzaj robotom pomaly nad čiarou. Tentoraz na displeji nebudeme zobrazovať intenzitu odrazeného svetla, ale len hodnoty 0 (biela) a 1 (čierna) z každého zo štyroch senzorov.

Python – overenie kalibrácie

```
import cyberpi, mbuild

cyberpi.display.show_label("  RGB SENSOR\n\n L2  L1  R1  R2\n", 16, 0, 0, index =
    0)
cyberpi.display.set_brush(255, 255, 0)

while True:
    L2 = mbuild.quad_rgb_sensor.is_line("L2",1)
    L1 = mbuild.quad_rgb_sensor.is_line("L1",1)
    R2 = mbuild.quad_rgb_sensor.is_line("R2",1)
    R1 = mbuild.quad_rgb_sensor.is_line("R1",1)

    vL1 = 1 if L1 else 0
    vL2 = 1 if L2 else 0
    vR1 = 1 if R1 else 0
    vR2 = 1 if R2 else 0

    cpi.display.show_label(vL2, 16, 10, 54, index = 1)
    cpi.display.show_label(vL1, 16, 45, 54, index = 2)
    cpi.display.show_label(vR1, 16, 75, 54, index = 3)
    cpi.display.show_label(vR2, 16, 110,54, index = 4)
```



Obr. 4. Pohľad na displej ak je robot na čiare

Prečo je dôležité kalibrovať senzor na tom istom mieste a pri tom istom osvetlení, kde bude robot aj jazdiť?

9.1. Meranie – tabuľka overenia

Posúvaj robota a zaznamenaj, čo displej zobrazuje:

Poloha robota	L2	L1	R1	R2	Správne?
Celý robot na bielej	0	0	0	0	✔
Čiara pod L1 a R1 (stred)	0	1	1	0	
Čiara len pod L1					
Čiara len pod R1					
Celý robot na čiernej					
Iné:					
Iné:					

Ak senzor na čiernej čiare ukazuje hodnotu 0 namiesto 1, čo to znamená a čo treba urobiť?"

10 Záver – pravidlá správnej kalibrácie

Čím väčší je rozdiel medzi hodnotou na bielej a hodnotou na čiernej, tým lepšia je kalibrácia. Prečo?

Doplň chýbajúce slová:

1. Kalibruj vždy pri _____ osvetlení, aké bude počas jazdy.
2. Výška senzora nad podložkou má byť _____ až _____ mm.
3. Ak sa osvetlenie v miestnosti výrazne zmení, treba kalibráciu _____.
4. Úspešnú kalibráciu overíme tak, že na bielej ploche vidíme hodnotu _____ a na čiernej _____.

Poznámka

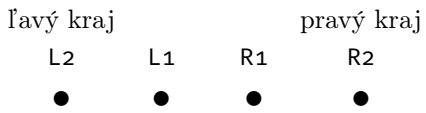
Kalibráciu je dobré zopakovať vždy, keď premiestnite robota do inej miestnosti alebo sa výrazne zmení denné svetlo.

Meno: _____ **Trieda:** _____ **Dátum:** _____

11 Zopakovanie – senzor čiary

V predchádzajúcej hodine sme spoznali Quad RGB senzor a naučili sme sa ho kalibrovať. Dnes ho použijeme na prvý skutočný algoritmus sledovania čiary.

Pripomeňme si usporiadanie snímačov pri pohľade zhora:



Poznámka

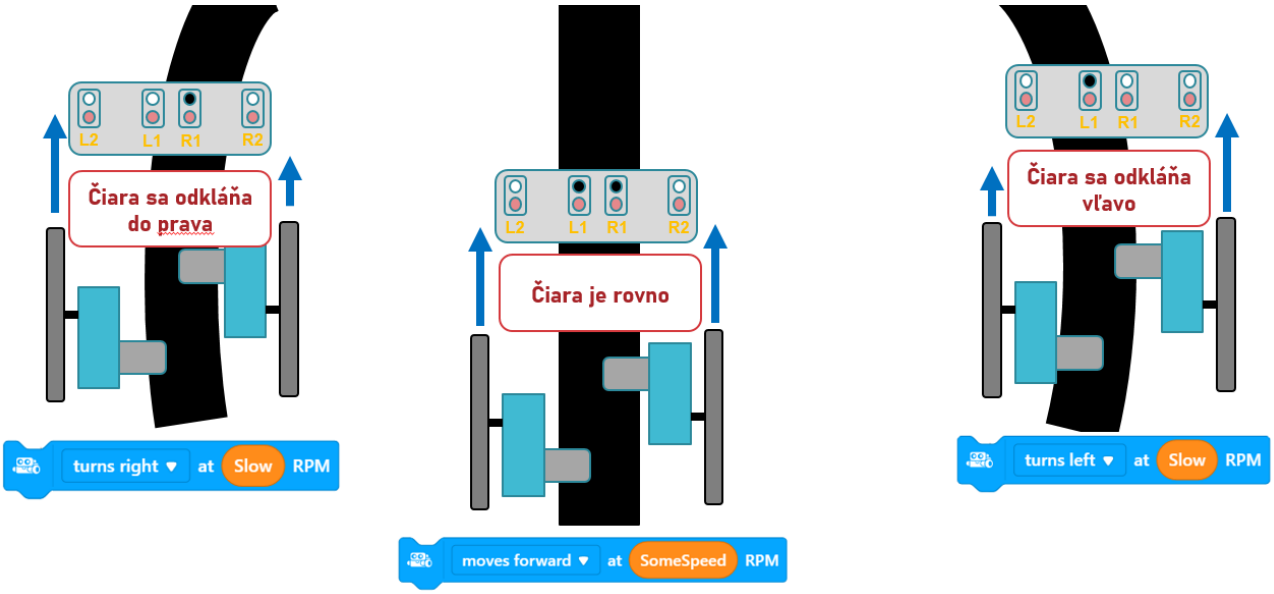
Dnes budeme pracovať len s dvoma **prostrednými** snímačmi – **L1** a **R1**. Krajné snímače L2 a R2 využijeme až neskôr.

Každý snímač vracia jednu z dvoch hodnôt:

Hodnota	Význam
`True` (1)	snímač vidí **čiernu čiaru**
`False` (0)	snímač vidí **bielu plochu**

12 Bang-Bang algoritmus

Bang-Bang je najjednoduchší spôsob riadenia – robot nereaguje plynule, ale prepína medzi pevne danými akciami podľa toho, čo práve vidí senzor. Žiadna matematika, žiadne výpočty – len séria podmienok `if/elif`.



Obr. 5. Najjednoduchší algoritmus pre navigáciu po čiare

12.1. Tabuľka stavov

Kombinácia hodnôt L1 a R1 určuje, čo robot urobí (pozri aj obr. ??):

L1	R1	Stav	Situácia	Akcia robota
1	1	3	čiara uprostred	choď rovno
1	0	2	čiara viac vľavo	zatoč vľavo
0	1	1	čiara viac vpravo	zatoč vpravo
0	0	0	čiara stratená	zastav

12.2. Pseudokód

Na základe tabuľky stavov, si najprv zapíšeme algoritmus slovami (poriadne si ho premyslite):

Pseudokód

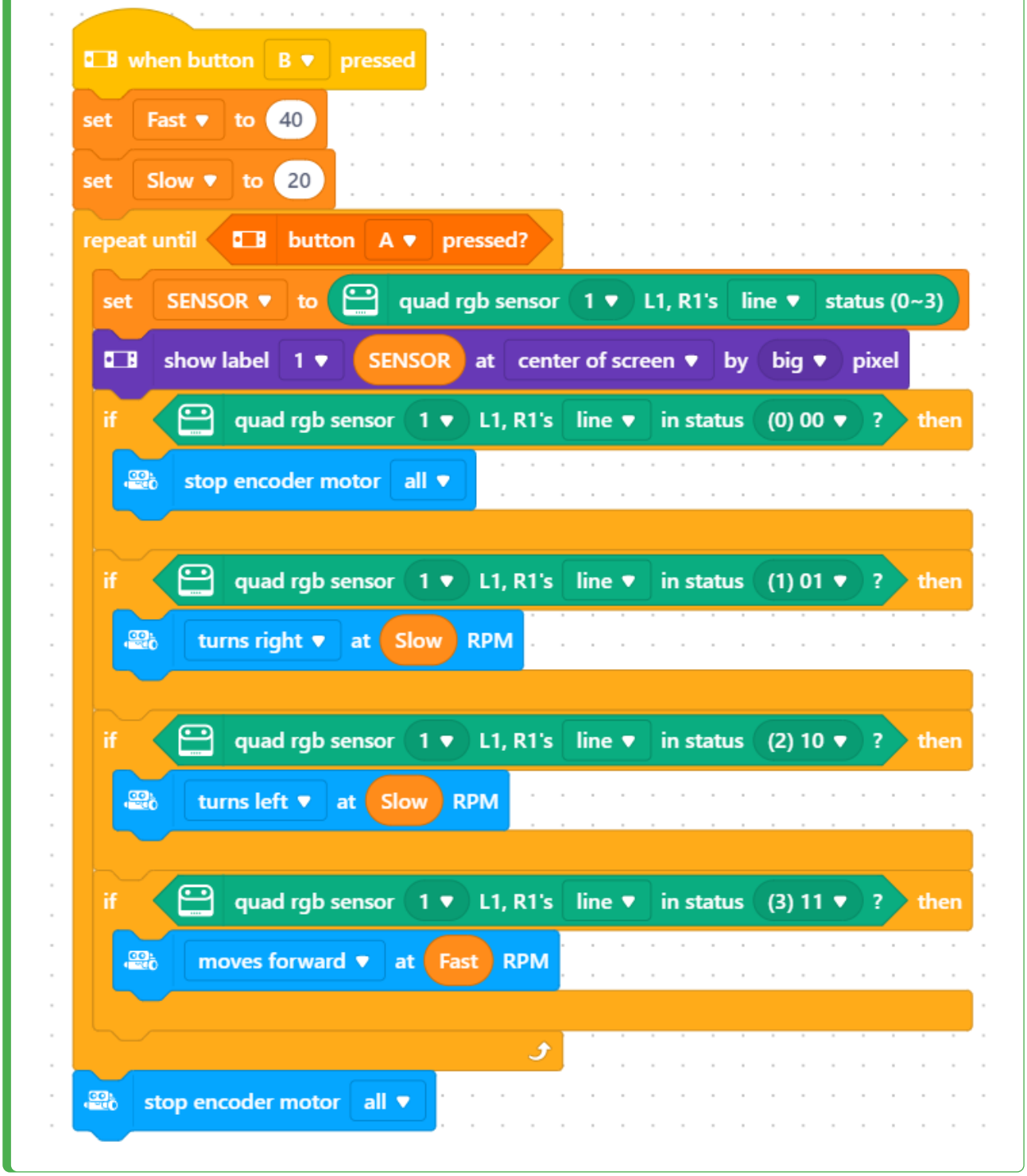
```
opakuj stále:
  L1 = hodnota ľavého stredného snímača
  R1 = hodnota pravého stredného snímača

  ak L1=1 a R1=1: choď rovno
  inak ak L1=1 a R1=0: zatoč vľavo
  inak ak L1=0 a R1=1: zatoč vpravo
  inak: zastav
```

Nakresli vývojový diagram tohto algoritmu. Nezabudni na: START, slučku OPAKUJ STÁLE a všetky štyri vetvy.

13 Program

13.1. Blockly

Blockly – Bang-Bang, 2 snímače



13.2. Python

Python – Bang-Bang, 2 snímače

```python
import event, time, cyberpi, mbuild, mbot2

SlowSpeed = 0
FastSpeed = 0
Sensor = 0

@event.is_press('b')
def is_btn_press():
    global SlowSpeed, FastSpeed, Sensor
    FastSpeed = 40
    SlowSpeed = 20

    while not cyberpi.controller.is_press('a'):
      Sensor = mbuild.quad_rgb_sensor.get_line_sta("middle", 1)
      cyberpi.display.show_label(Sensor, 48, "center", index= 0)

      if mbuild.quad_rgb_sensor.get_line_sta("middle", 1) == 3:
        mbot2.forward(FastSpeed)

      if mbuild.quad_rgb_sensor.get_line_sta("middle", 1) == 2:
        mbot2.turn_left(SlowSpeed)

      if mbuild.quad_rgb_sensor.get_line_sta("middle", 1) == 1:
        mbot2.turn_right(SlowSpeed)

      if mbuild.quad_rgb_sensor.get_line_sta("middle", 1) == 0:
        mbot2.EM_stop("ALL")
```

Poznámka k rýchlostiam

V programe používame dve rôzne rýchlosti pre pohyb vpred (FastSpeed) a pre otáčanie robota smerom ku čiare (SlowSpeed).

14 Testovanie a ladenie

14.1. Pred spustením – predpovedaj

Bude robot sledovať čiaru lepšie pri väčšej alebo menšej rýchlosti otáčania (SlowSpeed)? Prečo?

Čo sa stane, ak nastavíš obe rýchlosti na 100?

14.2. Tabuľka meraní

Vyskúšaj rôzne kombinácie rýchlostí a zaznamenaj výsledky. Ohodnoť každú kombináciu známkou 1-5 ako v škole:

Rýchlosť rovno	Rýchlosť zatáčania	Výsledok (1–5)	Poznámka
FastSpeed	SlowSpeed		
40	20		
50	30		
70	40		
10	50		
vlastné: _____	vlastné: _____		

15 Zamyslenie

Aká je hlavná nevýhoda Bang-Bang algoritmu? Popíš situáciu, kde by robot zlyhal.

Robot pri sledovaní čiary "cikcakuje" – osciluje zo strany na stranu. Navrhni spôsob ako to zlepšiť bez zmeny algoritmu (teda len zmenou čísel v programe).

Predstav si, že máš k dispozícii všetky 4 snímače (L2, L1, R1, R2). Aké nové situácie by si vedel rozlíšiť oproti dnešnej 2-snímačovej verzii? Vymenuj aspoň 2.

Bonusová výzva

Uprav program tak, aby robot pri strate čiary (stav 0, 0) nezastal, ale otočil sa na mieste a hľadal čiaru ďalej. Napíš pseudokód tohto riešenia.

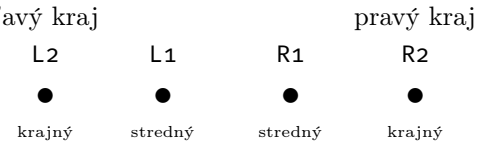
Meno: _____ **Trieda:** _____ **Dátum:** _____

16 Zopakovanie a motivácia

V predchádzajúcej hodine sme naprogramovali robota pomocou Bang-Bang algoritmu s dvoma prostrednými snímačmi L1 a R1. Robot fungoval, ale mal jednu zásadnú slabinu.

Čo sa stalo, keď robot prišiel na ostrý oblúk alebo keď čiara "ušla" obom snímačom naraz? Ako sa robot zachoval?

Dnes zapojíme aj **krajné snímače L2 a R2**. Robot tak bude mať väčší "záber" – uvidí čiaru skôr, než ju stratí, a dokáže reagovať na väčšiu odchýlku.



17 Reálne stavy senzora

So štyrmi snímačmi existuje teoreticky $2^4 = 16$ rôznych kombinácií. Nie všetky však nastanú pri normálnej jazde po súvislej čiare.

Zamysli sa a doplň chýbajúce akcie v poslednom stĺpci:

Status	L2	L1	R1	R2	Situácia	Akcia robota
0	0	0	0	0	žiadna čiara / koniec	zastav
1	0	0	0	1	čiara výrazne vpravo	_____
3	0	0	1	1	čiara mierne vpravo	_____
6	1	1	1	1	čiara uprostred	choď rovno
12	1	1	0	0	čiara mierne vľavo	_____
8	1	0	0	0	čiara výrazne vľavo	_____
15	1	1	1	1	všetko čierne	_____

Prečo stav `L2=1, L1=0, R1=0, R2=1` (krajné snímače vidia čiaru, prostredné nie) nemôže pri normálnej jazde po jednej súvislej čiare nastať?

Vymysli 2 ďalšie stavy, ktoré sú technicky možné, ale pri normálnej jazde po jednej čiare by nastali len výnimočne. Vysvetli kedy:

Stav _____: _____

Stav _____: _____

18 Pseudokód a vývojový diagram

18.1. Pseudokód

Pseudokód

```
opakuj stále:
  L2, L1, R1, R2 = hodnoty všetkých štyroch snímačov

  ak L2=1 a L1=1 a R1=1 a R2=1: choď rovno
  inak ak L2=1 a L1=1 a R1=1 a R2=0: zatoč jemne vľavo
  inak ak L2=1 a L1=0 a R1=0 a R2=0: zatoč ostro vľavo
  inak ak L2=0 a L1=1 a R1=1 a R2=0: zatoč jemne vpravo
  inak ak L2=0 a L1=0 a R1=0 a R2=1: zatoč ostro vpravo
  inak: zastav
```

Nakresli vývojový diagram tohto algoritmu. Použi šesťuholníky pre podmienky a obdĺžniky pre akcie.

19 Program

Python – Bang-Bang, 4 snímače

```python
import event, time, cyberpi, mbuild, mbot2

while True:
    cyberpi.display.show_label(mbuild.quad_rgb_sensor.get_line_sta("all", 1), 24,
        "center", index= 0)
    if mbuild.quad_rgb_sensor.get_line_sta("all", 1) == 6:
        mbot2.forward(50)

    if mbuild.quad_rgb_sensor.get_line_sta("all", 1) == 8:
        mbot2.turn_left(50)

    if mbuild.quad_rgb_sensor.get_line_sta("all", 1) == 12:
        mbot2.turn_left(30)

    if mbuild.quad_rgb_sensor.get_line_sta("all", 1) == 3:
        mbot2.turn_right(30)

    if mbuild.quad_rgb_sensor.get_line_sta("all", 1) == 1:
        mbot2.turn_right(50)

    if mbuild.quad_rgb_sensor.get_line_sta("all", 1) == 0:
        mbot2.EM_stop("ALL")
```

20 Porovnanie s 2-snímačovou verziou

20.1. Pred testovaním – predpovedaj

V čom očakávaš zlepšenie oproti 2-snímačovej verzii? Kde naopak neočakávaš rozdiel?

20.2. Tabuľka porovnania

Vyskúšaj obe verzie na rovnakej trati a zaznamenaj:

	2 snímače (ws02)	4 snímače (ws03)
Správanie na priamej čiare		
Správanie na miernom oblúku		
Správanie na ostrom oblúku		
Správanie pri strate čiary		
Celkové hodnotenie (1–5)		

Zodpovedala realita tvojim predpovediam? Čo ťa prekvapilo?

21 Zamyslenie

Bang-Bang algoritmus so 4 snímačmi má viac stavov, ale stále len "skokovú" reakciu. Čo by sa zmenilo, keby sme namiesto dvoch rýchlostí (napr. 10 a 50) mohli nastaviť rýchlosť plynule podľa toho, ako ďaleko je čiara od stredu?

Meno: _____ **Trieda:** _____ **Dátum:** _____

22 Čo je zlé na Bang-Bang?

V predchádzajúcich hodinách sme naprogramovali robota pomocou Bang-Bang algoritmu. Ak si sledoval pohyb robota pozorne, mohol si si všimnúť jeden nepríjemný jav.

Ako sa volá jav, keď robot striedavo prestriela čiaru na obe strany a "cikcakuje"? Čo je jeho príčina?

Poznámka

Bang-Bang nevie, o koľko je čiara vzdialená – vie len *ktorým smerom*. Dnes to napravíme: naučíme robota reagovať **silnejšie**, keď je čiara ďaleko, a **slabšie**, keď je blízko stredu.

23 Časť A – Experiment: digitálny senzor

23.1. Príprava

Robot je vypnutý. Polož ho na trať tak, aby čiara prechádzala priamo pod snímačmi L1 a R1 (uprostred). Budeš ho ručne pomaly posúvať kolmo na čiaru doľava a doprava a zapisovať hodnoty snímačov.

Nahraj do robota pomocný program, ktorý sme používali pri overení kalibrácie (pozri str. XX).

23.2. Meranie 1 – digitálne hodnoty

Posúvaj robota pomaly od krajnej ľavej polohy po krajnú pravú. Pri každej polohe zastav a zaznamenaj hodnoty L1 a R1:

Poloha robota	L1	R1	error = L1 − R1
Čiara úplne vľavo (pod L2)			
Čiara pod L1			
Čiara uprostred (L1 aj R1)			
Čiara pod R1			
Čiara úplne vpravo (pod R2)			

Aké rôzne hodnoty môže `error = L1 - R1` nadobudnúť pri digitálnom senzore? Vymenuj všetky možné hodnoty.

Koľko rôznych hodnôt erroru si dostal? Je to veľa alebo málo, ak chceme plynulú reguláciu? Prečo?

Ak by sme rýchlosť motorov vypočítali ako `rýchlosť = 50 + Kp × error`, koľko rôznych rýchlostí by motor mohol mať (pre ľubovoľné Kp)? Čo to znamená pre plynulosť jazdy?

Aha moment

P-regulátor s **digitálnym** senzorom je v skutočnosti len inak zapísaný Bang-Bang – má rovnaké obmedzenie. Potrebujeme senzor, ktorý dáva **viac informácií**.

24 Časť B – Experiment: analógový senzor

Analógový senzor nemeria len "čierna/biela", ale **intenzitu odrazeného svetla** – číslo od 0 do 100.

Hodnota	Význam
blízko 0	minimum odrazeného svetla = **čierna čiara**
blízko 100	maximum odrazeného svetla = **biela**

Zmeň program na analógové čítanie:

Python – zobrazenie analógových hodnôt

```python
import cyberpi, mbuild

cyberpi.display.show_label("L1      R1", 16, 0, 0, index=0)
cyberpi.display.set_brush(255, 255, 0)

while True:
    L1 = mbuild.quad_rgb_sensor.get_gray("L1", 1)
    R1 = mbuild.quad_rgb_sensor.get_gray("R1", 1)

    cyberpi.display.show_label(L1, 16, 22, 37, index=1)
    cyberpi.display.show_label(R1, 16, 95, 37, index=2)
```

24.1. Meranie 2 – analógové hodnoty

Posúvaj robota rovnako ako pri Meraní 1, tentokrát zapisuj analógové hodnoty (0–100) zo snímačov L1 a R1. Senzor dokáže aj priamo odmerať odchýlku od stredu čiary, túto hodnotu (Dev) tiež zapíš do tabuľky:

Poloha robota	L1	R1	error = R1 − L1	Dev
Čiara úplne vľavo (pod L2)				
Čiara medzi L2 a L1				
Čiara pod L1				
Čiara uprostred				
Čiara pod R1				
Čiara medzi R1 a R2				
Čiara úplne vpravo (pod R2)				

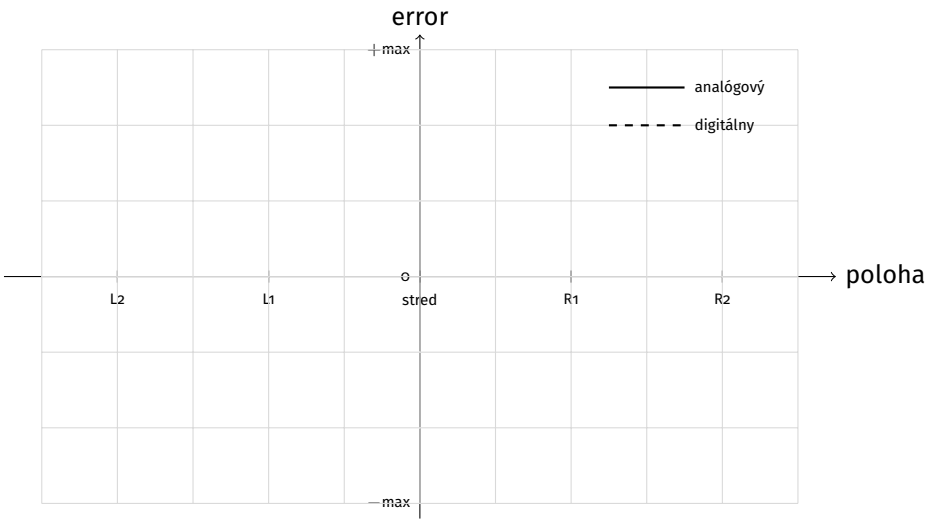
Aký je minimálna a maximálna odchýlka (error), ktorú si nameral?

Min. error: _____ **Max. error:** _____

Porovnaj počet rôznych hodnôt odchýlky (error) pri digitálnom a analógovom senzore. Ktorý je vhodnejší pre plynulú reguláciu a prečo?

24.2. Graf

Nakresli do grafu závislosť **odchýlky** (error) **od polohy robota** pre oba senzory. Použi namerané hodnoty.



Aký tvar má krivka analógového senzora? Pripomína ti niečo z matematiky?

25 P-regulátor

Teraz, keď vieme, že analógový senzor dáva plynulé hodnoty odchýlky, môžeme zostrojiť tzv. **proporcionálny regulátor (P-regulátor)**.

25.1. Princíp

Myšlienka je jednoduchá: **čím väčšia odchýlka, tým väčšia korekcia.**

$$\begin{aligned} \texttt{korekcia} &= K_p \times \texttt{error} \\ \texttt{ľavý motor} &= \texttt{základná\_rýchlosť} - \texttt{korekcia} \\ \texttt{pravý motor} &= \texttt{základná\_rýchlosť} + \texttt{korekcia} \end{aligned}$$

K_p (proporcionálna konštanta) určuje, ako silno robot reaguje na odchýlku. Musíme ju experimentálne naladiť.

Čo sa stane, keď `error = 0` (čiara presne uprostred)?

korekcia = _____ **ľavý motor =** _____ **pravý motor =** _____

Čiara je výrazne vľavo, `error = +100`, základná rýchlosť = 50, K_p = 0.3. Vypočítaj:

korekcia = _____ **ľavý motor =** _____ **pravý motor =** _____

Zatáča robot správnym smerom? _____

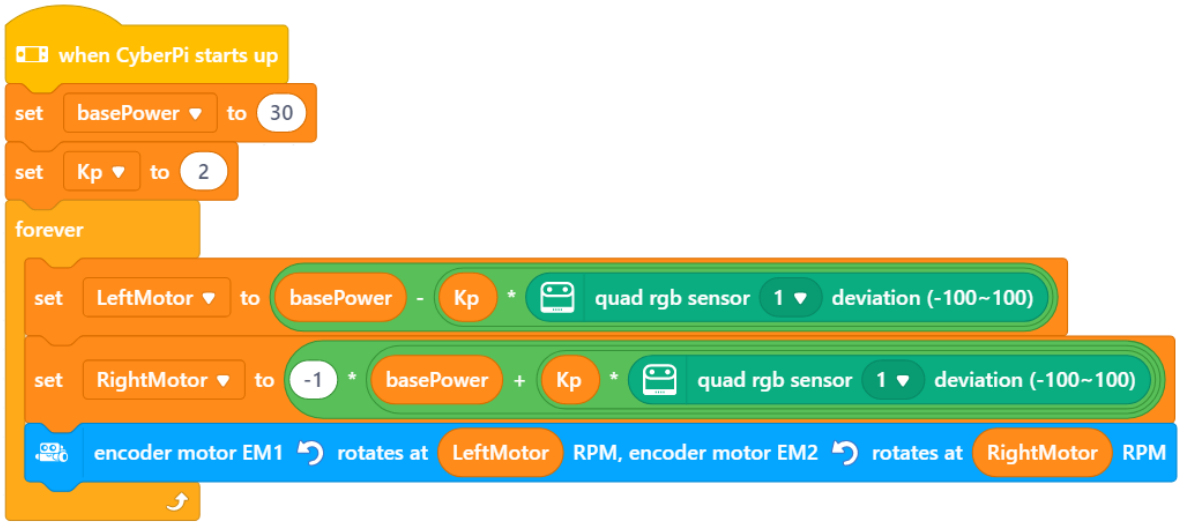
Ak zvýšime K_p z 0.3 na 1.5, čo sa stane so správaním robota? Predpovedaj:

25.2. Program

Python – P-regulátor

```python
import event, time, cyberpi, mbuild, mbot2


basePower = 30
Kp = 0.5
LeftMotor = 0
RightMotor = 0

while True:
    LeftMotor = (basePower - Kp * mbuild.quad_rgb_sensor.get_offset_track(1))
    RightMotor = -1 * ((basePower + Kp * mbuild.quad_rgb_sensor.get_offset_track
        (1)))
    mbot2.drive_speed(LeftMotor, RightMotor)
```

Blockly – P-regulátor



26 Ladenie K_p

Vyskúšaj rôzne hodnoty K_p a zaznamenaj správanie:

K_p	Správanie na rovinke	Správanie v zákrutách	Hodnotenie (1–5)
0.1			
0.3			
0.5			
1.5			
2.0			
vlastné: _____			

Nájdi hodnotu K_p, pri ktorej robot jazdí najlepšie. Má aj P-regulátor nejakú slabinu – správa sa robot stále dokonale?

27 Záver a porovnanie

Ohodnoť každý algoritmus známkou 1 (najlepší) až 5 (najhorší) na základe vlastného testovania:

	Bang-Bang (2)	Bang-Bang (4)	P-regulátor
Priamočiara čiara			
Mierny oblúk			
Ostrý oblúk			
Plynulosť pohybu			
Náročnosť programu			

Aká je najdôležitejšia výhoda P-regulátora oproti Bang-Bang?

Ďalší krok

P-regulátor reaguje len na **aktuálnu** odchýlku, nie na to, ako rýchlo sa odchýlka mení. Ak robot "prestriela" čiaru a odchýlka rastie rýchlo, P-regulátor to nevie včas zachytiť. To rieši **PD-regulátor** – pridáme derivačnú zložku, ktorá reaguje na *zmenu* erroru. Ale to je téma na ďalšiu hodinu!