Regulátory

ING. MARTIN HLINOVSKÝ, PHD.

KATEDRA ŘÍDICÍ TECHNIKY, FEL ČVUT

Motivace

Regulace v každodenním životě, o které ne tak často přemýšlíme:

- Sprchování (nastavení správné teploty)
- Holení (úprava pohybu ruky podle pozorování v zrcadle)

Více technické:

- Termostatické kohouty na topení, nemusíme stále chodit a vypínat/zapínat topení

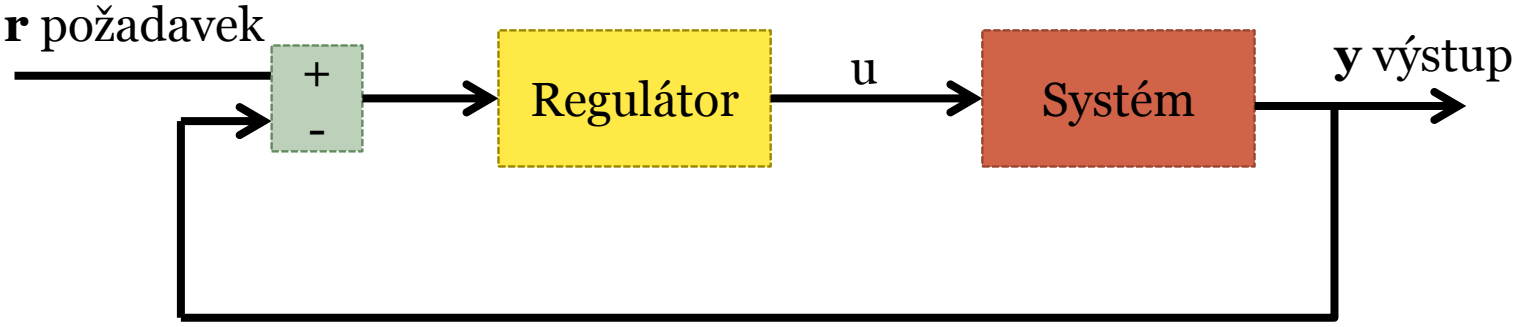
Motivace

Více technické/ robotické:

- řízení otáček motoru – při zátěži motoru, např. brzdění kola o nějaký předmět, by se bez zpětné vazby se nic nestalo
- Ale potřebujeme zvýšit napětí na vstupu motoru, aby podal vyšší výkon, překonalo se brzdění a otáčky zůstaly konstantní

Jak na to

• Známe požadovanou hodnotu **r** a zní odvodíme vstup do systému **u** (ot/min na napětí motoru)



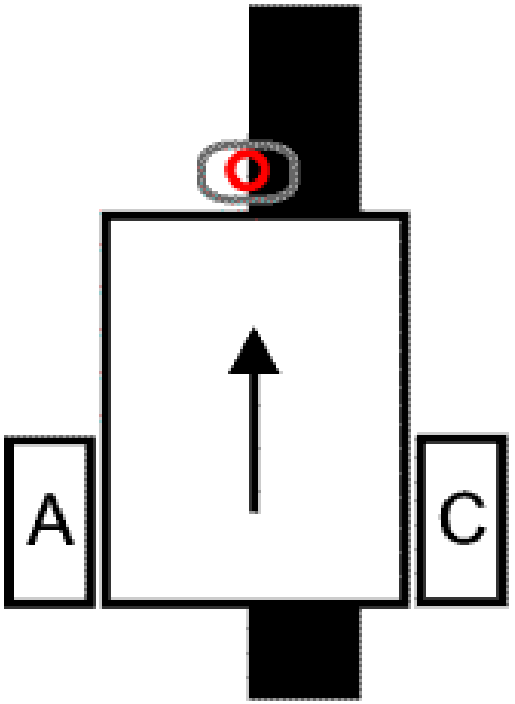
Zpětná vazba

• Zavedení jen zpětné vazby nestačí, jak zkombinovat požadovanou a skutečnou výstupní hodnotu?
• Zajímavá je pro nás jen odchylka, nechceme to celé změnit, ale jen trochu dorovnat výstup podle požadavku
• Jak z odchylky odvodit změnu na vstupu systému? => Vhodným nastavením konstant regulátoru – P, I, D

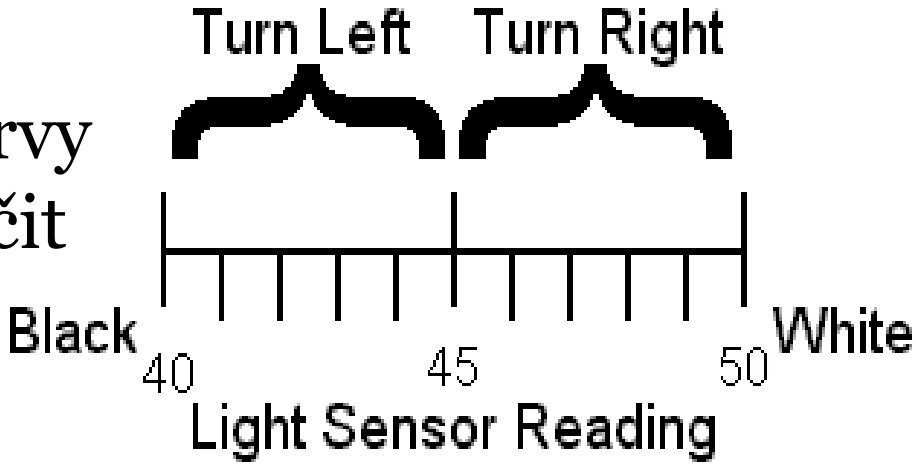
Robotický příklad

Sledování černé čáry:

- Diferenciální podvozek (2 kola a opěrný bod)
- Světelný senzor pro snímání barev
- Sledování hrany – víme, zda zatáčí doleva či doprava

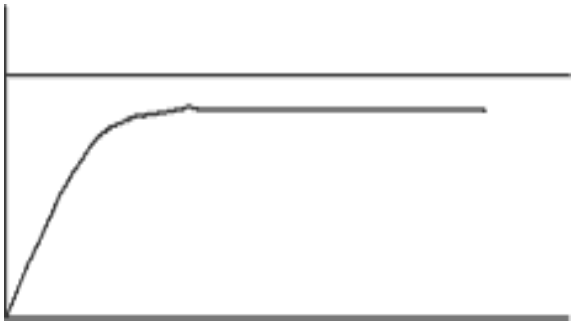


- Zjistíme hodnoty pro obě barvy a převedeme na potřebu zatočit
- Nutno změřit, typicky 40-50

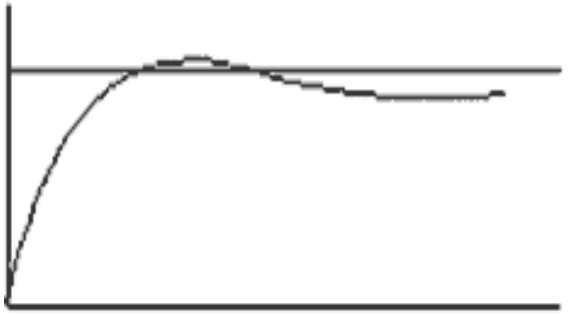


Proporcionální regulátor (P)

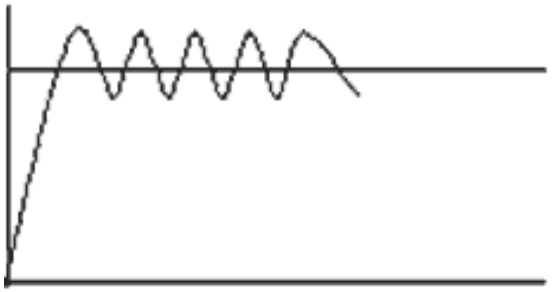
• Funguje podobně jako zesilovač - regulátor přenásobí odchylku výstupní hodnoty od požadované hodnoty danou konstantou



Malá konstanta



Dobře nastavená konstanta

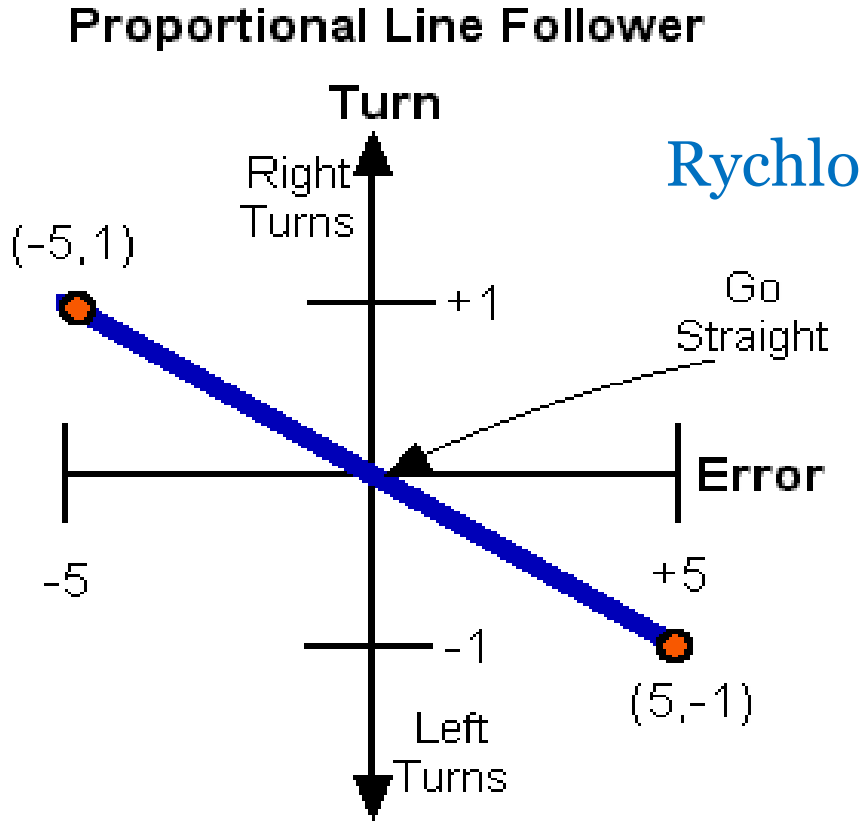


Přiliš vysoká konstanta

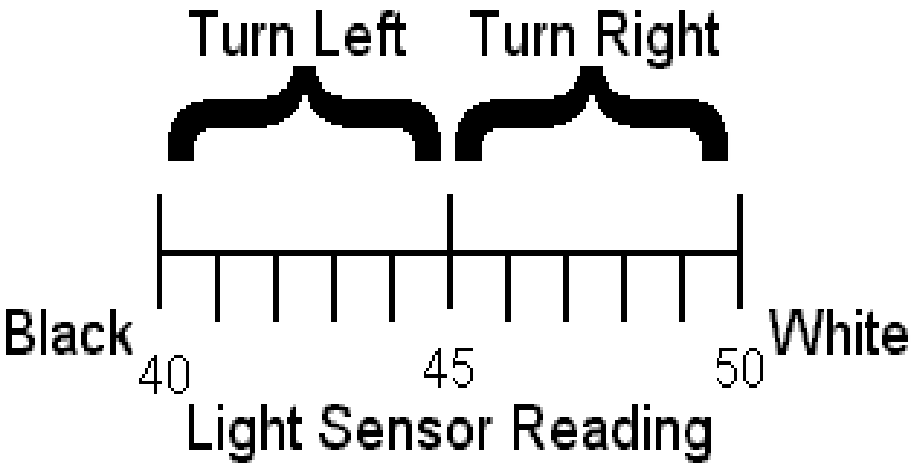
• Problém – nikdy nezískáme přesně požadovanou hodnotu kvůli odchylce tak malé, která po přenosábení konstantou nezpůsobí znatelnou změnu systému

Proporcionální regulátor (P)

- Můžeme využít např. při sledování čáry
- pokud jsme daleko od čáry uděláme velkou zatáčku, pokud blízko tak menší

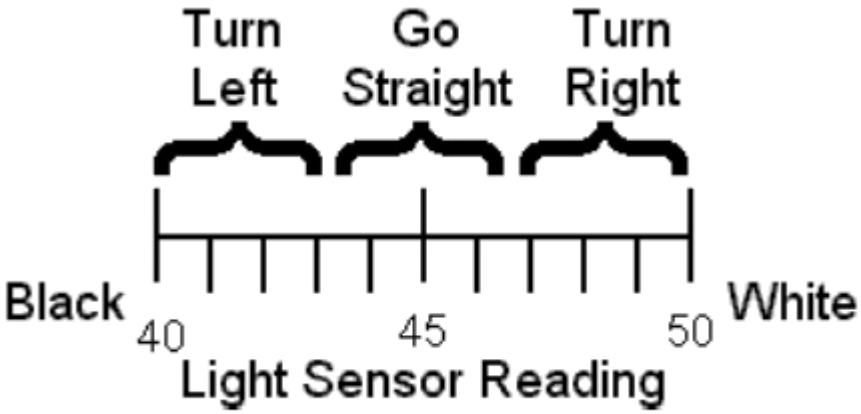


Rychlost_zataceni = Pkonstanta * odchylka



Proporcionální regulátor (P)

- Lepší řešení:
  - pokud jsme daleko od čáry uděláme velkou zatáčku, pokud blízko tak menší, ale hlavně, pokud jsme na rozhraní čáry (43 až 47), tak robot jede rovně



Pseudokód pro P-regulátor

```
//Inicializace proměných
Kp = 10  //konstanta regulátoru
zadana = 45 //žádaná hodnota, rozhraní černé a bílé
//Začátek smyčky
While(true)
  LightValue = vyčti_světelný_senzor () //Aktuální hodnota senzoru
  error = LightValue – zadana           //výpočet odchylky
  Turn = Kp * error                     //Výpočet konstanty pro zatáčení
  MOTOR A = + Turn                      //Odeslání hodnot do motoru
  MOTOR C = - Turn                      //Odeslání hodnot do motoru
//Konec smyčky
```

Teď robot bude prudce zatáčet, kola v protipohybu. Vylepšení – nastavení offsetové rychlosti, robot plynuleji zatáčí

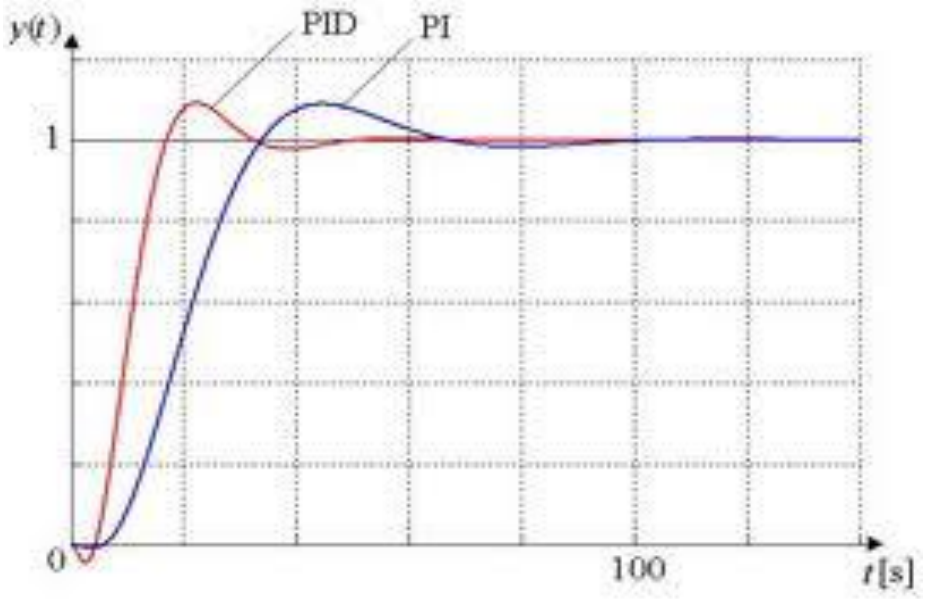
```
rychlost_offset = 50
MOTORA= rychlost_offset + Turn
MOTORC= rychlost_offset - Turn
```

Proporcionální a Integrační regulátor (PI)

- Pro lepší odezvu můžeme přidat Integrační složku, nasčítává odchylky integral = integral + error
- Řeší problém P regulátoru – i malá odchylka se nakonec projeví => zmizí trvalá regulační odchylka
- Například:
  - Při prvním odečtu senzoru je chyba 1, při dalším 2, při dalším zůstává stále dva, ale integral = 1+2+2 = 5
  - Výpočet zatočení: Turn = Kp*(error) + Ki*(integral)
  - Větší zásah, ale nebezpečí rokmitání

Proporcionální, Integrační a Derivační regulátor (PID)

- Pro rychlejší ustálení na požadované hodnotě přidáme Derivační složku a vznikne PID regulátor
- Složka P se snaží odstraňovat chyby aktuální, složka I chyby minulé a složka D chyby budoucí



Proporcionální, Integrační a Derivační regulátor (PID)

- Odstranění budoucí odchylky - budeme předpokládat, že její vývoj se nemění, tedy jak se změnila minulá odchylka na aktuální, tak se změní aktuální na budoucí
- Příklad: předchozí odchylka byla 2, aktuální je 5

  změna = 2 – 5 = -3, také označováno derivát Derivace = změna
- Budoucí odchylku můžeme odhadnout
- Budoucí_odchylka = aktuální_odch + aktuální_změna
- Tedy: Budoucí_odchylka= 2 + (-3) = -1

Pseudokód pro PID-regulátor

```
//Inicializace proměných
Kp = 10             //konstanta  P regulátoru
Ki  = 1             //konstanta I regulátoru
Kd  = 100           //konstanta D regulátoru
zadana = 45 //žádaná hodnota, rozhraní černé a bílé
rychlost_offset = 50       //rovnou uvažujeme vylepšení s plynulejší jízdou
//potřebné pomocné proměné
Integral  = 0               //proměná na sčítání odchylek
LastError  = 0              //proměná na uložení poslední odchylky
Derivate = 0                //uložení derivátu, změny odchylky ve dvou stavech
//Začátek smyčky
While(true)
  LightValue = vyčti_světelný_senzor ()          //Aktuální hodnota senzoru
  error = LightValue – zadana                    //výpočet odchylky
  Integral = Integral + error                    //sčítání odchylek
  Derivate = error – LastError                   //spočtení změny
  Turn = Kp * error+Ki*Integral + Kd*Derivate    //Výpočet konstanty pro zatáčení
  MOTOR A   = rychlost_offset +Turn              //Odeslání hodnot do motoru
  MOTOR C   = rychlost_offset – Turn              //Odeslání hodnot do motoru
  LastError = error                              //uložení aktuální odchylky
//Konec smyčky
```

Jak nastavit a odladit PID konstanty

Nejdříve ladíme konstantu KP

- Pokud $K_P = K_C = 300$, $P_C = 0{,}8$ a $dT = 0{,}014$ pak využijeme následující tabulku pro první odhad konstant a pak dolaďujeme

Ziegler–Nichols method giving *K'* values (loop times considered to be constant and equal to dT)			
Control Type	**Kp**	**Ki'**	**Kd'**
P	$0.50K_c$	0	0
PI	$0.45K_c$	$1.2K_p dT / P_c$	0
PID	$0.60K_c$	$2K_p dT / P_c$	$K_p P_c / (8dT)$

Kp = (0.60)(Kc) = (0.60)(300) = 180
Ki = 2(**Kp**)(**dT**) / (**Pc**) = 2(180)(0.014) / (0.8) = 6.3 (which is rounded to 6)
Kd = (**Kp**)(**Pc**) / ((8)(**dT**)) = (180)(0.8) / ((8)(0.014)) = 1286

```
#define Kp 180
#define offset 50
#define HiSpeed 85
#define Ki 20
#define Kd 1200
#define LowSpeed 20

int actual, error, turn, powerB, powerC,
integral, derivative, lasterror;

task main()
{
SetSensorLight(IN_3);
integral = 0;
derivative = 0;
lasterror = 0;

do
{
  OnFwd(OUT_BC, LowSpeed);
}
while (Sensor(IN_3) > offset);

PlayTone(220,500);

while (true)
    {
    actual = Sensor(IN_3);
    error = offset - actual;
    derivative = error - lasterror;
    turn = (Kp * error)+ (Ki * integral) + (Kd *
derivative);
    turn = turn / 100;
    powerB = HiSpeed - turn;
    powerC = HiSpeed + turn;
    if (powerB < 0)
    {
      powerB = 0;
    }
    if (powerC > 100)
    {
      powerC = 100;
    }
    OnFwd(OUT_C,powerC);
    OnFwd(OUT_B,powerB);
    integral =+ error;
    lasterror = error;
    }
}
```

Jak změny v chování robota způsobí zvýšení hodnot parametrů Kp, Ki a Kd?

Effects of increasing parameters				
Parameter	**Rise time**	**Overshoot**	**Settling time**	**Error at equilibrium**
K_p	Decrease	Increase	Small change	Decrease
K_i	Decrease	Increase	Increase	Eliminate
K_d	Indefinite (small decrease or increase)	Decrease	Decrease	None

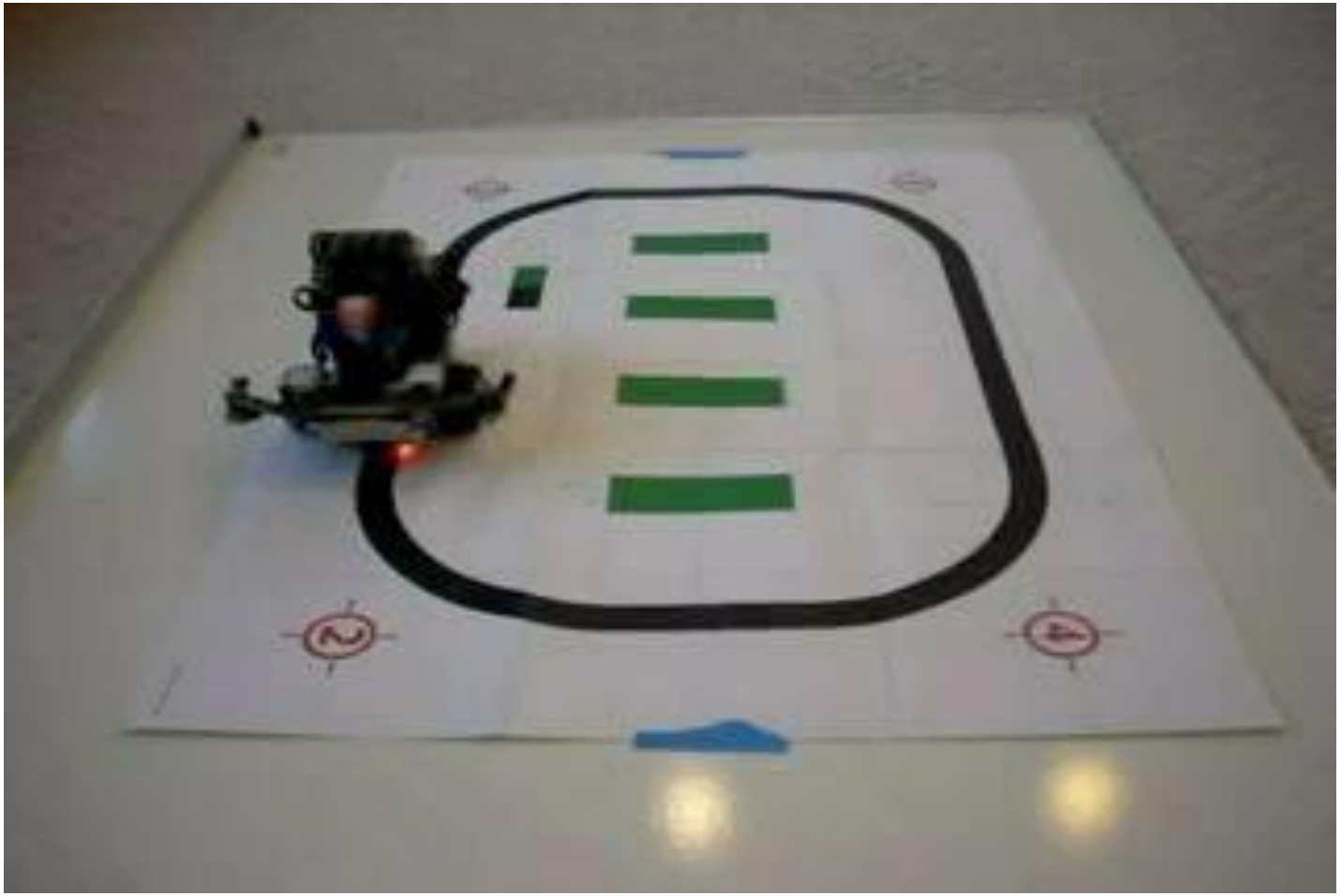
Rise time = doba náběhu

Overshoot = překmit, přesah

Setting time = doba ustálení

Error at equilibrium = chyba v rovnováze

A už nic nebrání v plynulé jízdě....



Použitá literatura

- http://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html
- http://www.techbricks.nl/My-NXT-projects/feed/rss.html

Poděkování

- Bc. Monika Svědirohová
- Ing. Lenka Mudrová

Děkuji za pozornost....