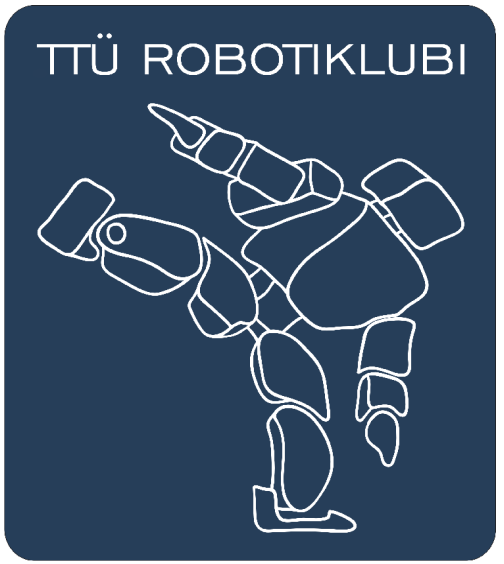
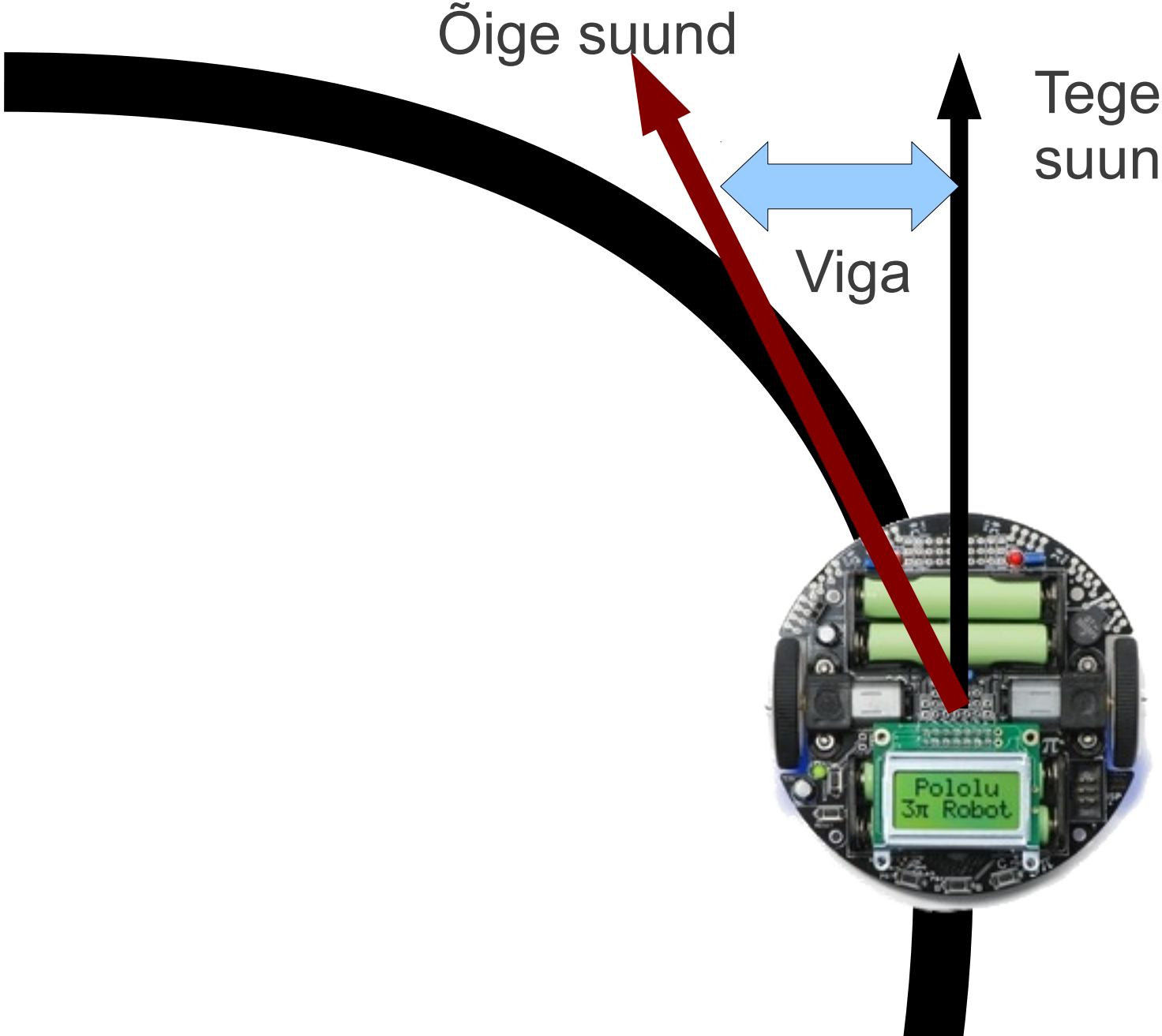
PID-kontroller

Teeme Ise 2011

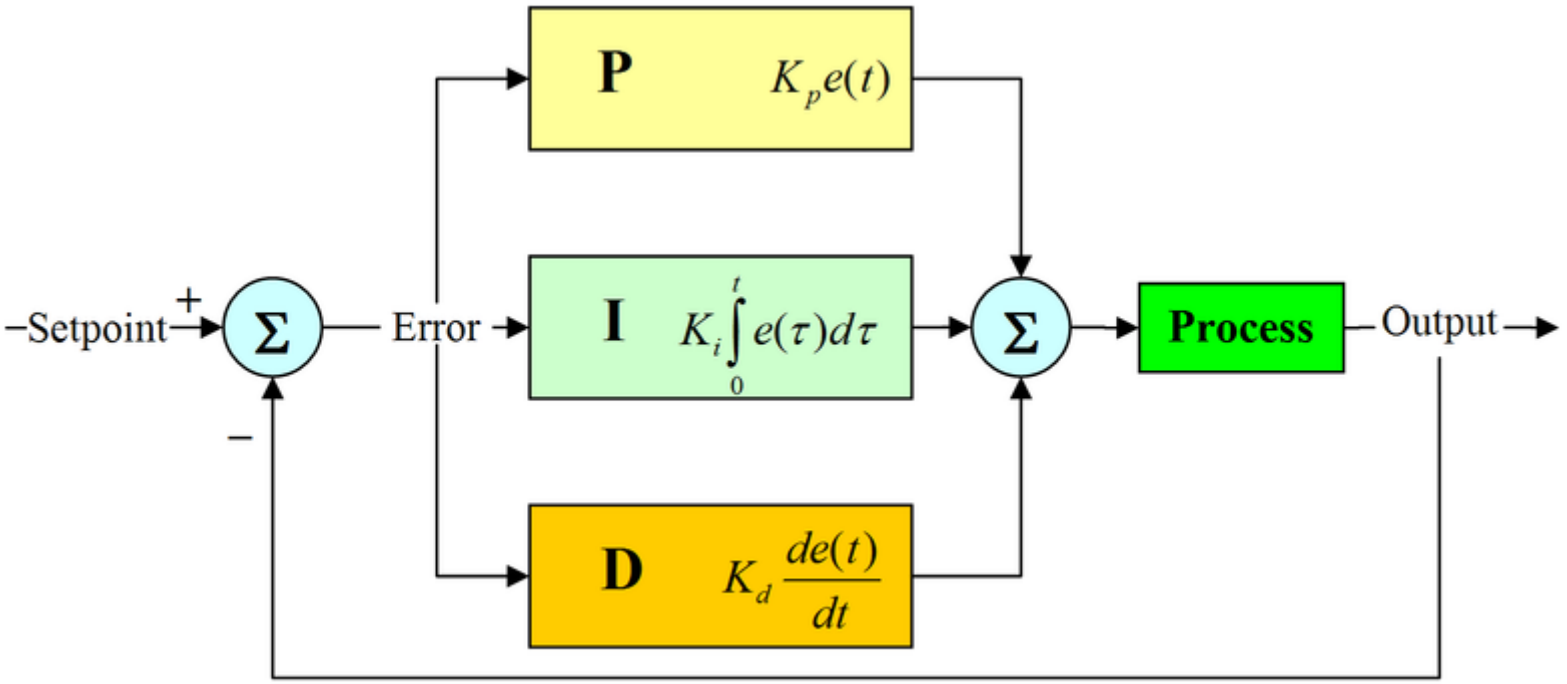


14.04.2011 Rasmus Raag

Kontroller vea minimeerimiseks



Plokkskeem

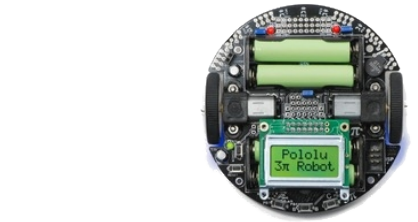


Allikas: http://en.wikipedia.org/wiki/File:Pid-feedback-nct-int-correct.png

P

- Proportsionaalne juhtimine: mida suurem on viga, seda suurem on parandustegur.

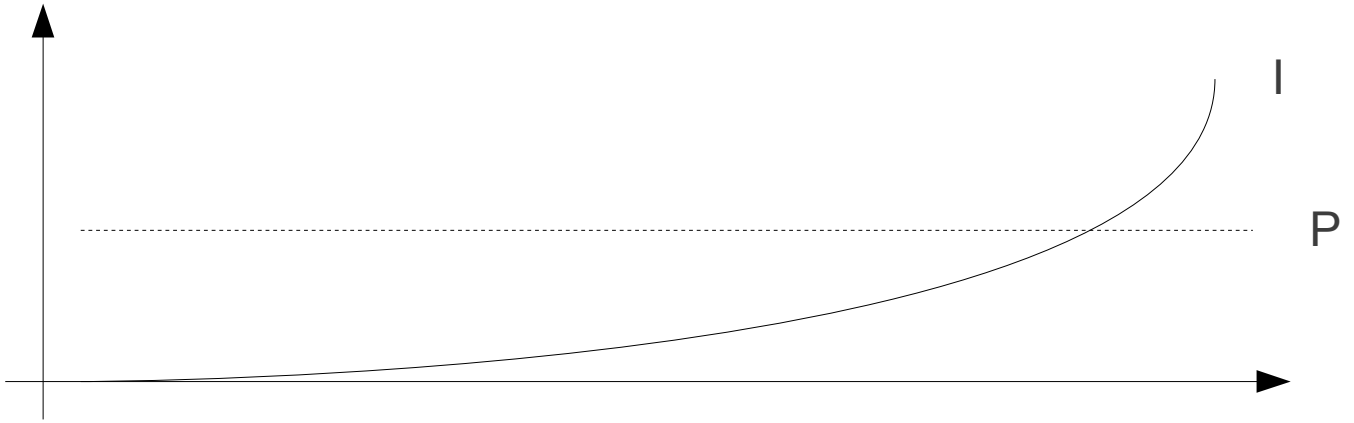
Liiga suur Kp



Liiga väike Kp

I

- Vea integreerimine: vea summeerimine pikema aja jooksul.
- Kui P ei suuda trajektoori piisavalt kiiresti parandada, muutub I järjest suuremaks, kuni lõpuks P ja I koostöös õige trajektoor saavutatakse.



D

- Derivatiivne komponent mõõdab vea muutumise kiirust.
- Vea kiirel kasvamisel on D positiivne ja võimendab parandustegurit.
- Vea kahanemisel on D negatiivne ja aeglustab vea parandamist.

Parameetrite leidmine proovimise teel

- Alusta ***Kp***-ga, pane ***Ki*** ja ***Kd*** nulliks.
- Suurenda ***Kp***-d, kuni tekib võnkuv trajektoor.
- Jaga ***Kp*** kahega.
- Suurenda ***Ki***-d
- Suurenda ***Kd***-d

Effects of *increasing* a parameter independently

Parameter	Rise time	Overshoot	Settling time	Steady-state error	Stability[3]
K_p	Decrease	Increase	Small change	Decrease	Degrade
K_i	Decrease[4]	Increase	Increase	Decrease significantly	Degrade
K_d	Minor decrease	Minor decrease	Minor decrease	No effect in theory	Improve if K_d small

Allikas: http://en.wikipedia.org/wiki/PID_controller

Kaherattaline robot

- Juhtimiseks kaks parameetrit: vasaku ja parema mootori kiirused.
- Saab esitada sõidukiiruse ja pööramiskiirusena:
  - vasak_kiirus = sõidukiirus + pööramiskiirus
  - parem_kiirus = sõidukiirus – pööramiskiirus
- Fikseerime sõidukiiruse
- Pööramiskiirus on PID-kontrolleri väljund

Pololu PID-kontroller

```
// Get the position of the line.  Note that we *must* provide
// the "sensors" argument to read_line() here, even though we
// are not interested in the individual sensor readings.
unsigned int position = read_line(sensors,IR_EMITTERS_ON);

// The "proportional" term should be 0 when we are on the line.
int proportional = ((int)position) - 2000;

// Compute the derivative (change) and integral (sum) of the
// position.
int derivative = proportional - last_proportional;
integral += proportional;

// Remember the last position.
last_proportional = proportional;
```

Pololu PID-kontroller

```
// Compute the difference between the two motor power settings,
// m1 - m2.  If this is a positive number the robot will turn
// to the right.  If it is a negative number, the robot will
// turn to the left, and the magnitude of the number determines
// the sharpness of the turn.
int power_difference = proportional/20 + integral/10000 + derivative*3/2;

// Compute the actual motor settings.  We never set either motor
// to a negative value.
const int max = 60;
if(power_difference > max)
    power_difference = max;
if(power_difference < -max)
    power_difference = -max;

if(power_difference < 0)
    set_motors(max+power_difference, max);
else
    set_motors(max, max-power_difference);
```