Robotické vozítko sledující čáru

Dominik Fuxa
Miroslava Škutová
Petr Černocký
Martin Bezecný

Úvod

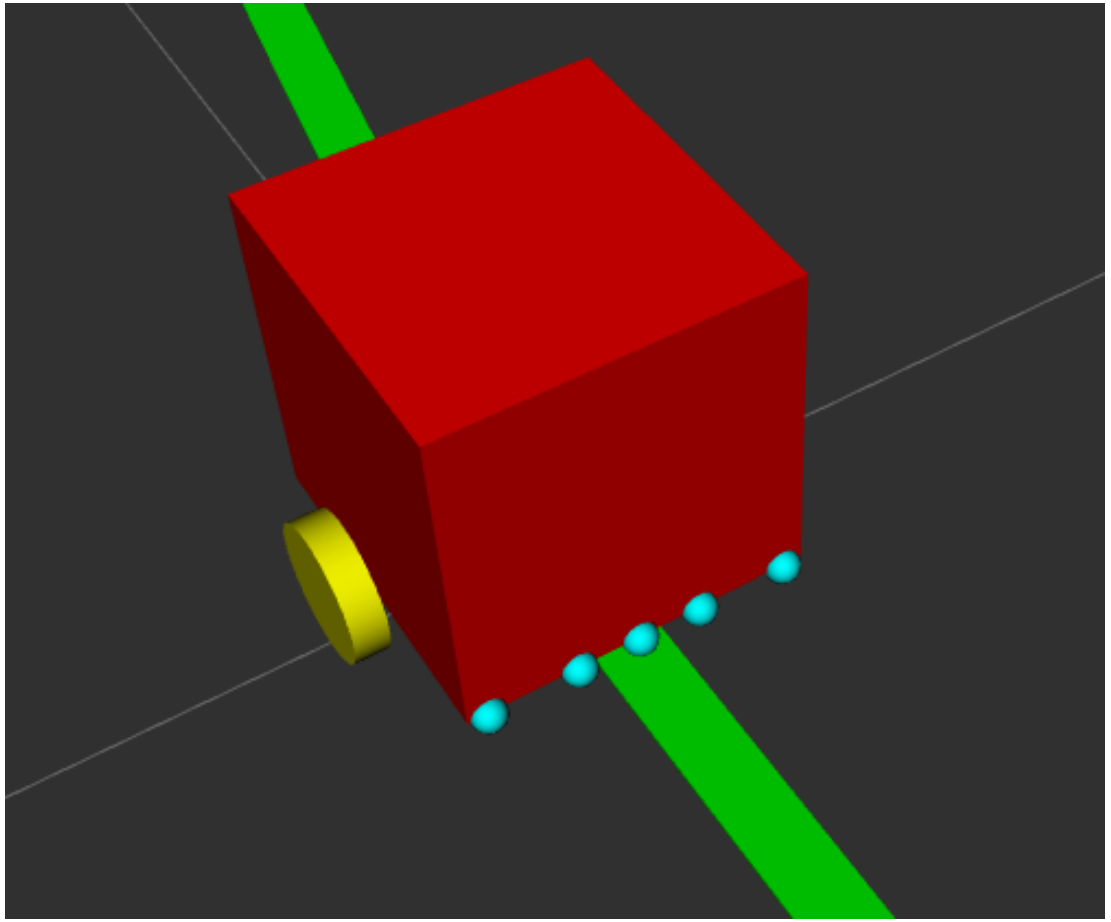
Celý projekt se skládá z několika tříd

Regulace za pomocí PSD regulátoru

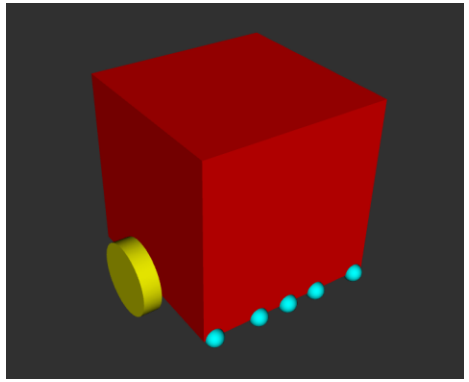
Hlavní ovládací kód formou stavového automatu

Tělo robota

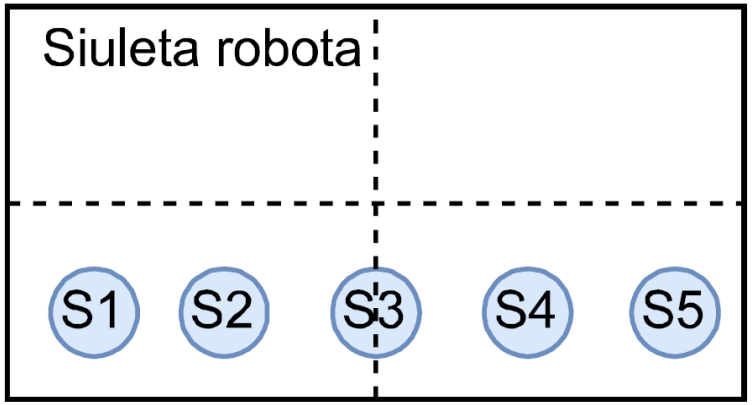
- Velikost: 10x10x10 cm
- Velikost kol: 2cm
- Chassis base: 12cm
- Maximální rychlost: 10cm/s

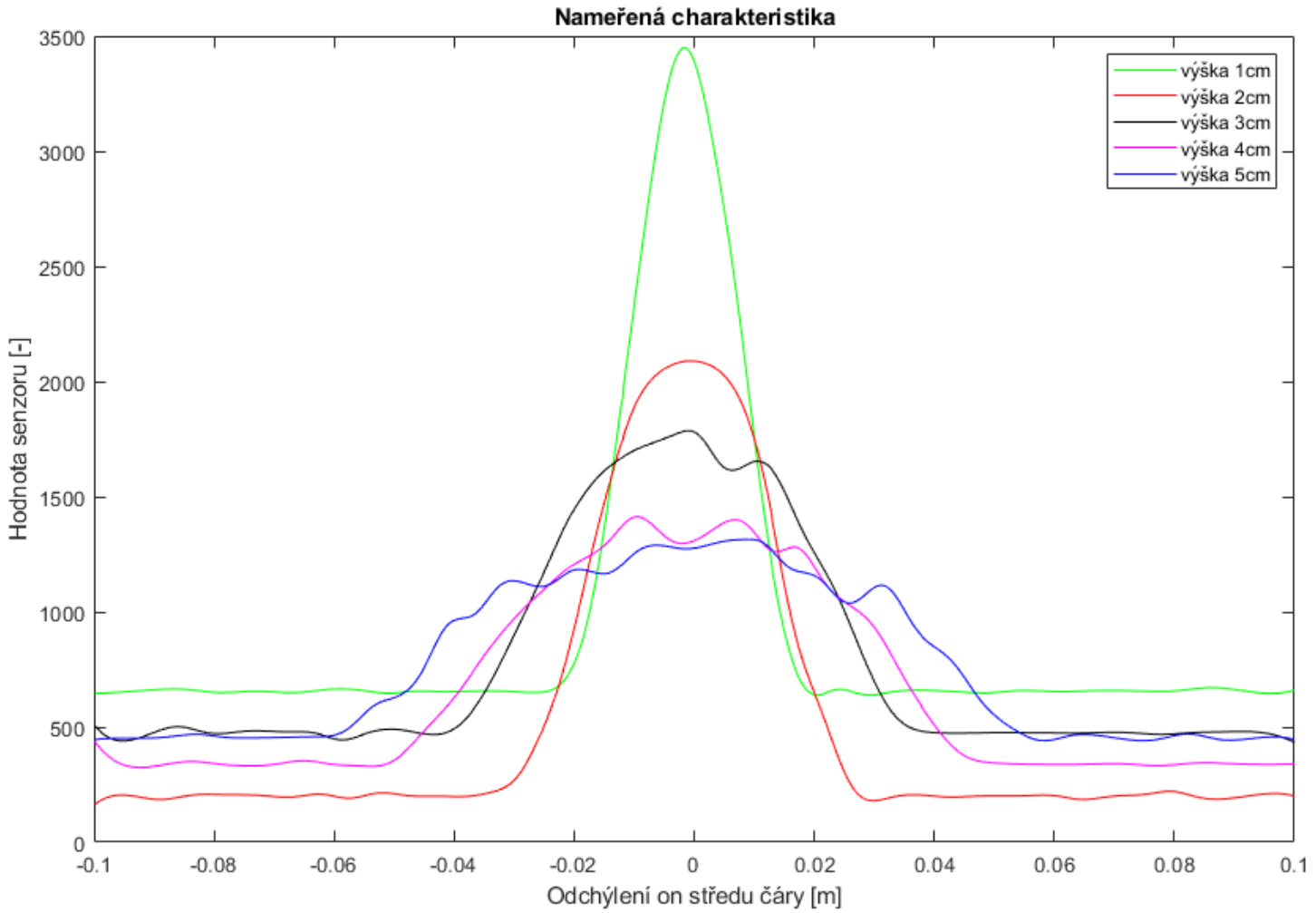


Senzory

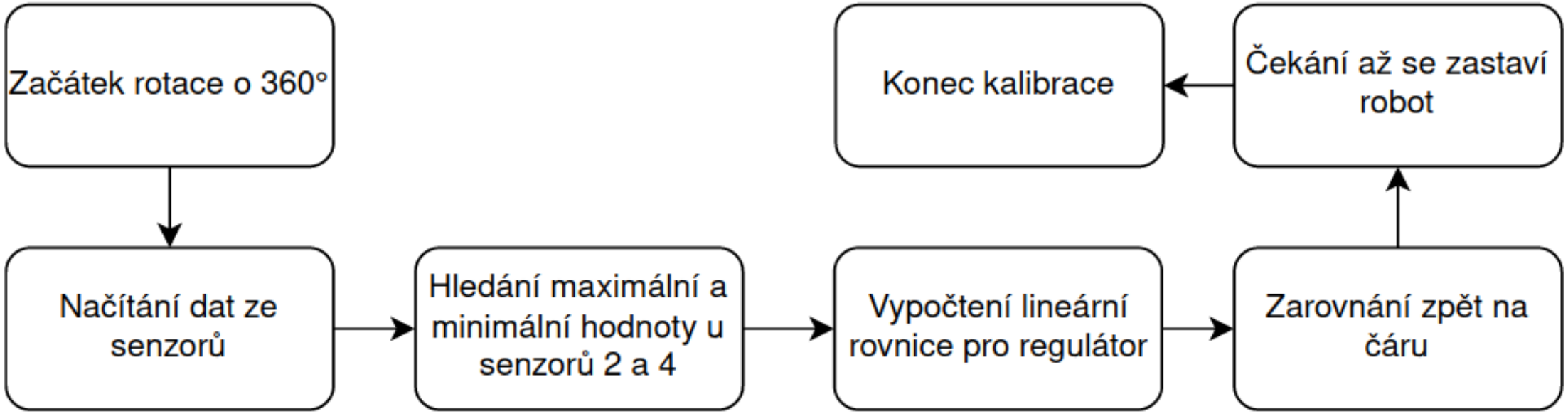


- Umístění
- Počet a jejich rozmístění
- Diferenčně zapojené





Kalibrace senzorů



Ukázka kódu pro kalibraci

```
if(abs( x: drive.trace_length) < 0.36 ){
    if(abs( x: drive.trace_length) < 2)
        comm.Send( m: drive.BuildSpeed( v: 0,  w: -80));
    else
        comm.Send( m: drive.BuildSpeed( v: 0,  w: -5));
}
else{
    comm.Send( m: drive.BuildSpeed( v: 0,  w: 0));
    regulator.a_line = 0.36/(sensor.sensor_max[1]+sensor.sensor_max[3]);
    regulator.b_line = 0.18 - sensor.sensor_max[3]*regulator.a_line;
    kalibrace = drive.delta_left == 0;
    if(kalibrace == 0 && vypis){
        std::cout << "Sensory max hodnoty:" << sensor.sensor_max[1]<< ":" << sensor.sensor_max[3] << std::endl;
        std::cout << "y = " << regulator.a_line<< "*x + " << regulator.b_line << std::endl;
        vypis = false;
    }
```

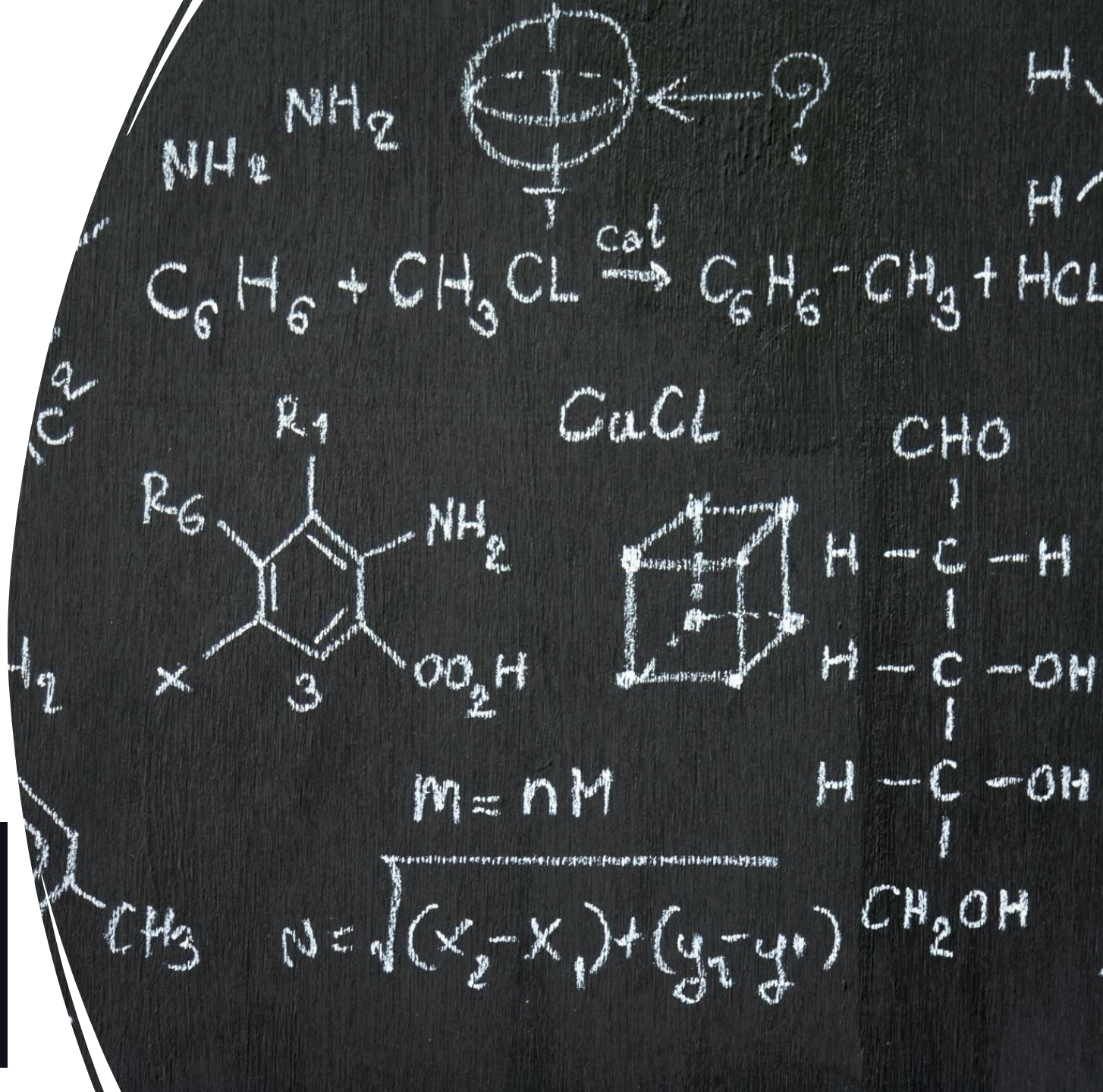
PSD Regulátor

- Realizován v třídě Regulator.cpp
- Vyladěn experimentální metodou
- Konstanty: Ti=0,08

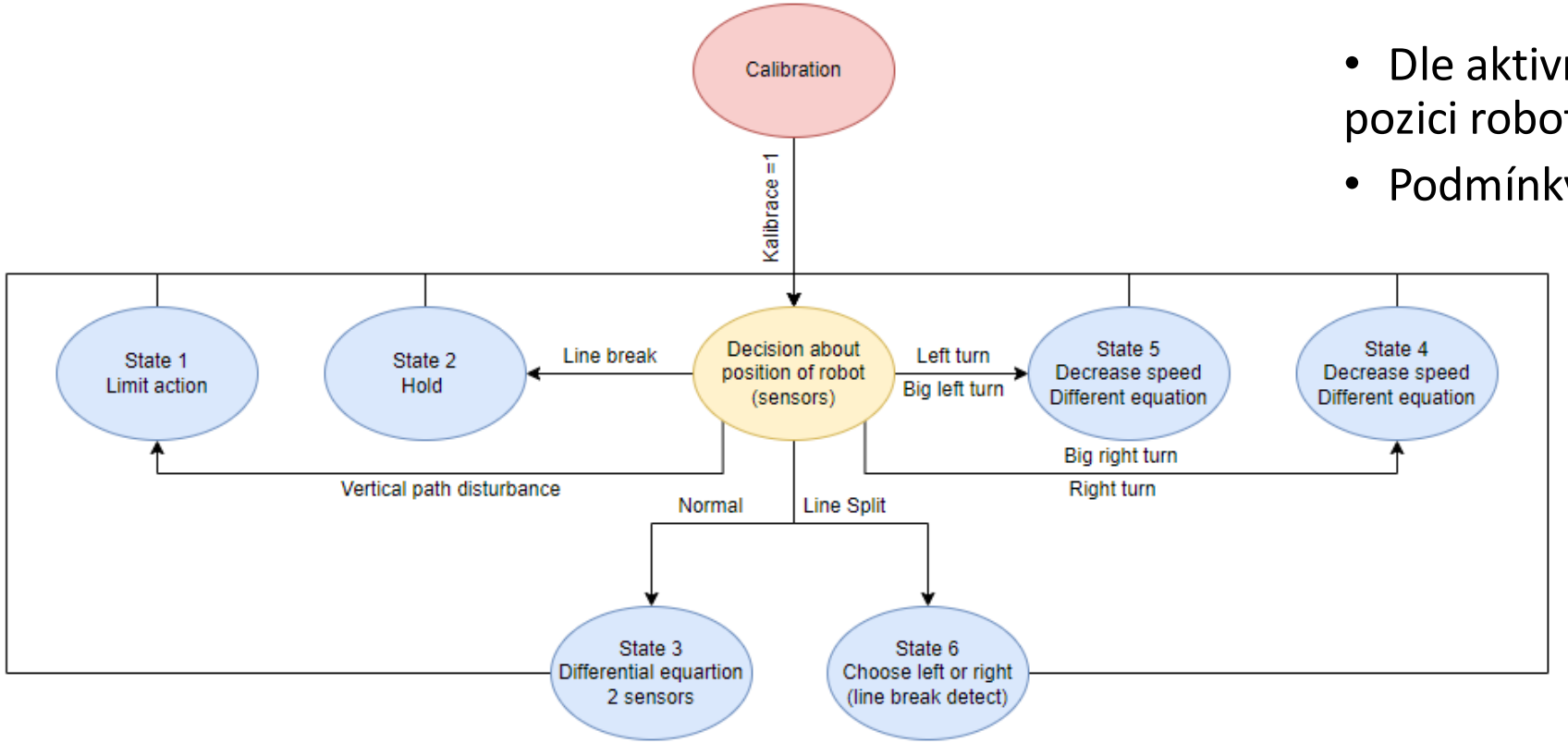
  Td=5

  K=0,65

```
double Regulator::PSD(double delta_angle, double velocity) {
    double sum = error.sum();
    double last_err = error.operator[]( i: error.size()-1);
    error = error.shift( n: 1);
    error.operator[]( i: error.size()-1) = delta_angle;

    return PSD_K*(delta_angle+1/PSD_Ti*sum+PSD_Td*(delta_angle-last_err));
```



Ovládání za pomocí stavového automatu



- Dle aktivních čidel rozlišujeme aktuální pozici robota a podle ní reagujeme
- Podmínky jsou vytvořeny pomocí masky

Ukázka masky

```
if(sensor.SensorState( mask_on: 0,  mask_off: 31)) {
    state = 2;

    line_break_bool = false; // false vypne omezení bočních senzorů při přerušení

    line_break_detect_bool = true;
    line_break_detect = -1;
}else if(sensor.SensorState( mask_on: 31,  mask_off: 0)){
    state = 1;
}else if(sensor.SensorState( mask_on: 24,  mask_off: 3) || sensor.SensorState( mask_on: 3,  mask_off: 24)){
    state = 4;
}else if(sensor.SensorState( mask_on: 16,  mask_off: 7) || sensor.SensorState( mask_on: 1,  mask_off: 28)){
    if (!line_break_bool)
        state = 5;
}else if(sensor.SensorState( mask_on: 27,  mask_off: 4)){
    state = 6;
}else if(sensor.SensorState( mask_on: 14,  mask_off: 0) || sensor.SensorState( mask_on: 4,  mask_off: 0)){
    state = 3;
    line_break_bool = false;
}else{
    //ugh
    state = 3;
}
```

```
* 5 SENSORS
* ROBOT INDEXING : R 4 3 2 1 0 L
* MASKING        : 4 3 2 1 0
*
* POSSIBLE MODES
* if
*  1 1 1 1 1        vertical path disturbance          STATE 1
*  0 0 0 0 0        line break                         STATE 2
* else if
*  X 1 1 1 X        normal mode, 2 differial sensors   STATE 3
*  1 1 X 0 0        left turn                          STATE 4
*  1 X 0 0 0        big left turn                      STATE 5
*  0 0 X 1 1        right turn                         STATE 4
*  0 0 0 X 1        big right turn                     STATE 5
*  1 1 0 1 1        path split                         STATE 6
* else
*  X X X X X        the world will burn
```

Nastavitelné parametry a regulace rychlosti

- Regulace rychlosti – P regulátor
- Engine brake - pomocná brzda

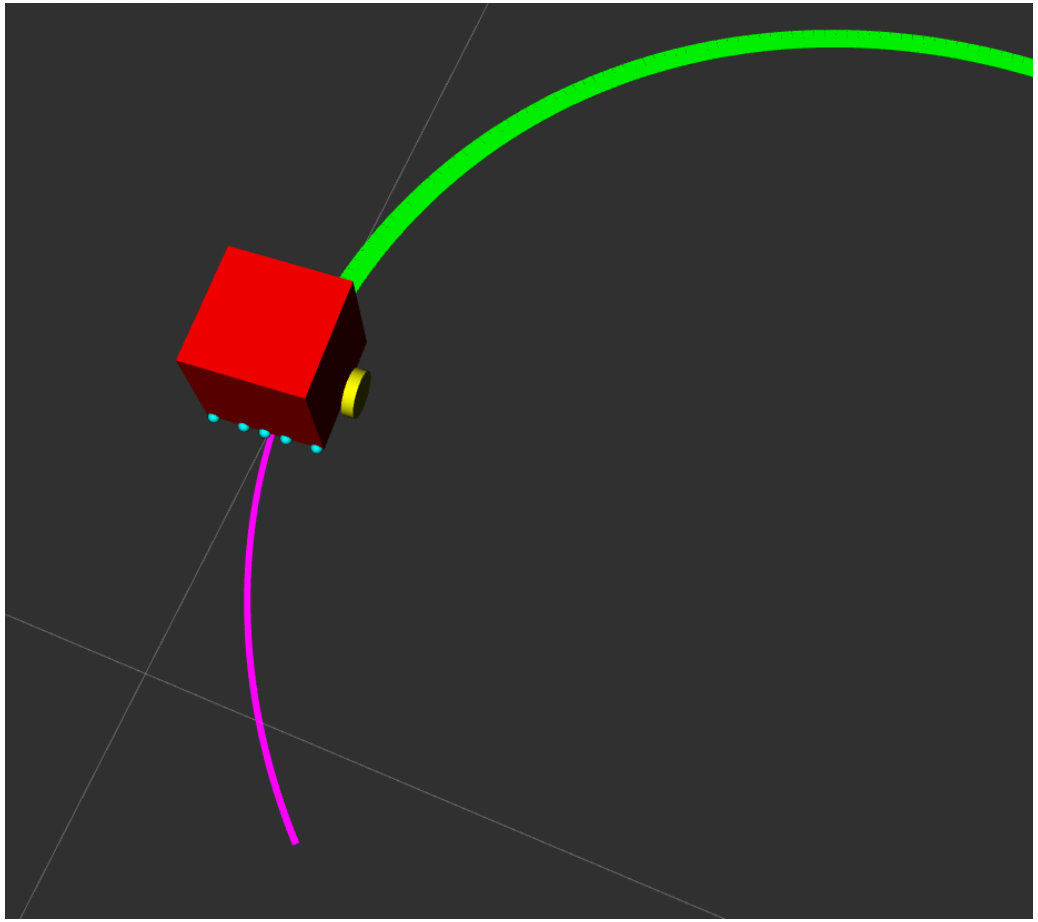
```
w = w < -w_max ? -w_max : w > w_max ? w_max : w;


v = v_max-abs( x: w*v_max/(w_max-engine_brake));
if (v < 0)
    v = 0;
```

```
// sett parametrs
double v_max = 0.10; // max robot speed at straight line
double w_max = 40; // max w
double samples_off = 10; // delte last samples from buffer_w
double engine_brake = 5; // increase variable for better engine brake
double line_break_detect_distance = 0.6; // distance between line break and branch
```

Jízda po přerušené čáře

- Použito pole
- Průměr pole dá vysledné W
- Možnost odebrání posledních X prvků z pole

```
// line break
double sum = 0;
for(int i = 0; i < buffer_w.size() - samples_off; i++)
    sum += buffer_w[i];
w = sum/(buffer_w.size()-samples_off);
break;
```



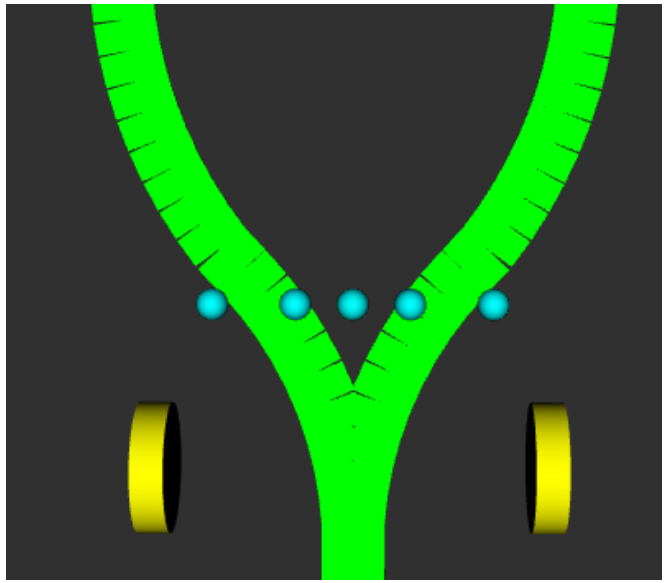
Detekce rozbočení

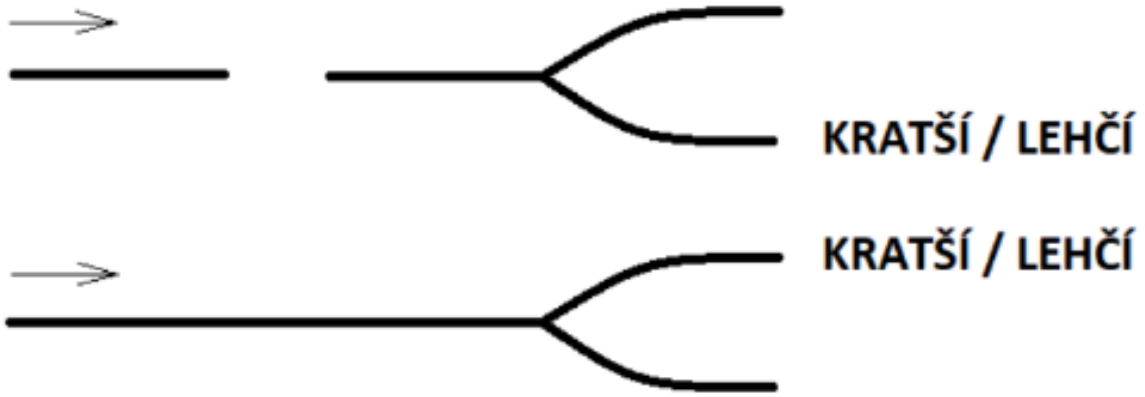
- Rozhodování na základě vzdálenosti
- Vzdálenost se dá libovolně měnit

```
if (line_break_detect < 0){
    if (line_break_detect_bool) {
        delta_distance = drive.trace_length;
        distance = 0;
        line_break_detect_bool = false;
    }
    distance = drive.trace_length - delta_distance;
    if (distance > line_break_detect_distance){
        line_break_detect = 1;
    }
```

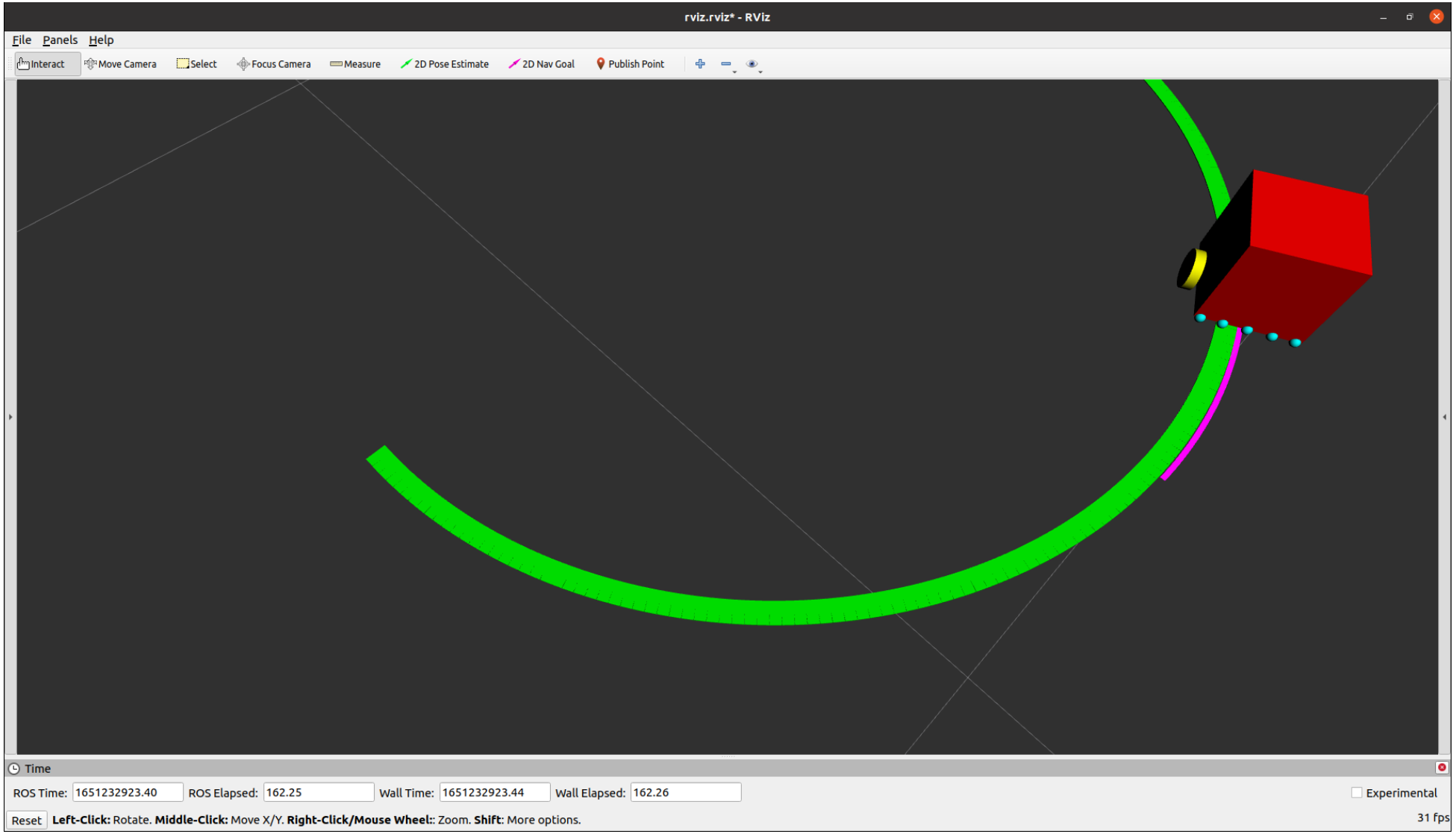
```
case 6:
    // line split
    delta_sens = -3000*(line_break_detect);  // linebreak ⟹ snažší při detekci pravá
            //- sensor.sensor[4]
            //+ sensor.sensor[3];

    diff = regulator.delta_to_angle( delta: delta_sens);
    w = regulator.PSD( delta_angle: diff,  velocity: v);

    buffer_w = buffer_w.shift( n: 1);
    buffer_w.operator[]( i: buffer_w.size()-1) = w;

    if(sensor.sensor_max[1]*0.9 < sensor.sensor[1])
        state = 1;
    break;
default:
    break;
```

```
if(sensor.SensorState( mask_on: 0,  mask_off: 31)) {
    state = 2;
    line_break_bool = false; // false vypne omezení bočních senzorů při přerušení
    line_break_detect_bool = true;
    line_break_detect = -1;
```





Ukázka regulace do zatáčky



Děkujeme za pozornost