Projekt

BPC-PRP

Robotické vozítko sledující čáru

Dominik Fuxa
Martin Bezecný
Petr Černocký
Miroslava Škutová

Obsah

Seznam obrázků

Seznam programů

1 Úvod

V této práci jsme měli za úkol navrhnout řídící program pro sledovače čáry. Tento sledovač byl realizován za pomocí simulátoru ROS (Robot Operating Systems).
Robot by si měl poradit s různými typy čar. Jak s klasickou rovnou čarou či obloukem, tak by měl zvládnout i přerušení jak čáry tak oblouků. Zároveň pro pozdější soutěž by měl zvládnout detekovat falešné („Slepé") čáry. Dále by se pak měl dokázat na základě pravidel záverečné soutěže rozhodnout, jakou cestou se vydat.
Pro celou programovou část projektu byl využit nástroj CLion. Prograomvá část je psána v jazyce C++ a skládá se ze tříd:
Comm - třída pro řešení komunikace
Drive - třída pro řešení ovládání robota
NMEA - třída pro dešifrování NMEA zpráv
Regulator - třída pro regulaci
Sensor - třída pro zpracování signálů ze sensorů
Hlavní supersmyčka programu se pak nachází v souboru main.cpp. Máme zde i zaimplementován stavový automat, který zatím obsahuje pouze možnosti, kdy jedeme po čáře a kdy není čára detekována. Později jej doplníme o další stavy pro správné projetí soutěžní dráhy.

2 Návrh robota

V tomto projektu se nebudeme zabývat návrhem reálného robota, ale pokusíme se navrhnout model, který se bude co nejvíce podobat reálnému návrhu. Pro ladění využijeme nástroje ROS. Obecné parametry robota byly pro realistický návrh ponechány v základním nastavení:

1. wheel_radius: 0.02
2. chassis_base: 0.12
3. width: 0.1
4. length: 0.1
5. height: 0.1

Tyto parametry slouží pro generování simulovaného modelu v ROSu. Kola jsou poháněna krokovými motorky s parametry:

1. Wheel: wheel_diameter * pi [m/turn]
2. Stepper motor: 200 [step/turn]
3. Motor driver: 32 [microstep/step]

Tyto parametry je nutné znát pro výpočet aktuální rychlosti motorků. Tento výpočet budeme provádět ve třídě Drive.

2.1 Senzory

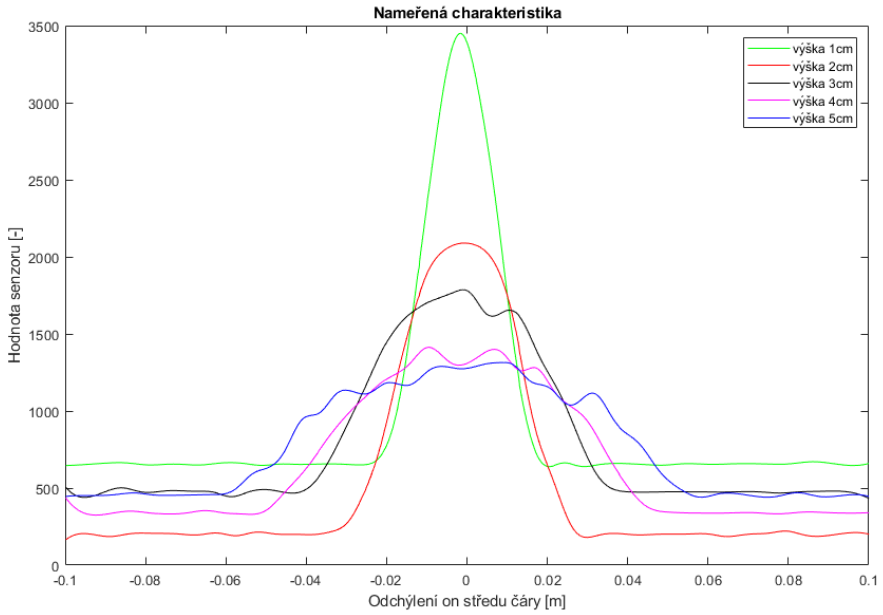
Pro regulaci jsme chtěli využít co nejméně senzorů. V prvním návrhu byly použity senzory dva, v diferenčním zapojení každý po jedné straně čáry. Kvůli malému rozsahu senzorů (snímače čáru zaznamenají až ze vzdálenosti 2cm od čáry) ale musely být velmi blízko u sebe, a protože robot nemůže tak prudce měnit rychlost, odchylka se nestihla vyregulovat. To jsme vyřešili přidáním třetího snímače "do kříže", který určoval skutečnou orientaci robota vůči čáře. V tomto zapojení už se podařilo navrhnout funkční regulátor.

V teoretickém návrhu je toto zapojení funkční. Kvůli velmi špatným snímačům (v klidové pozici se hodnota jednoho snímače liší až o 300, což je více než 10% jeho maximální hodnoty) je ale určení polohy robota vůči

čáře velmi nepřesné a proto i regulátor vykazoval velké odchylky a kmitání. Při dostatečně velkém zesílení schopném vyregulovat ostřejší zatáčky dochází k velkým zákmitům právě kvůli nepřesným snímačům.

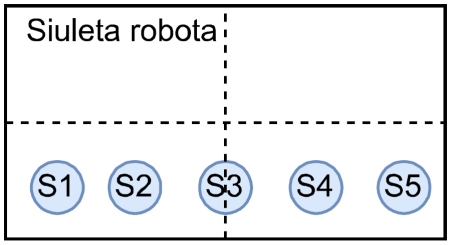
Proto bylo zvoleno řešení s čtyřmi snímači v řadě, umístěnými vepředu robota. Výhodou tohoto zapojení je možnost detekovat ostřejší zatáčky (jsou aktivní dva snímače boční snímače zároveň) a vhodně zvýšit výstupní deltu ze senzorů a tím i akční zásah.

Pro určení ideální výšky sensorů jsme vycházeli z naměřených charakteristik viz. 2.1. Z něj je patrné že senzory ve výšce 1cm mají mnohem větši citlivost než ostatní a také vetší odstup od šumu.



Obrázek 2.1: Naměřená chrakteristika čidel

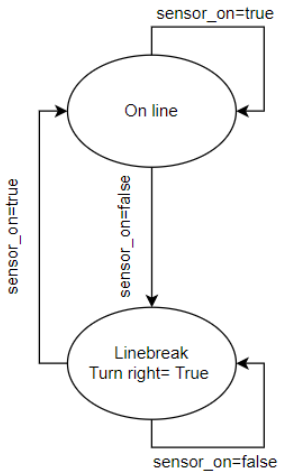
Po menších problémech při špatné detekci zda robot je na čáře nebo není, které bylo zapříčiněné vetšími mezerami mezi senzory, takže vznikal hluchý úsek, tak bylo nakonec vytvořeno řešení, které obsahovalo celkem 5 senzorů v řadě vedle sebe. Prostřední senzor tam je čistě jen od zaplnění hluchého místa a správné detekci kdy se nachází mimo čáru.



Obrázek 2.2: Umístění jednotlivých senzorů na robotu

2.2 Návrh algoritmu pro řízení

Jak již bylo zmíněno, celý projekt se skládá z několika dříd, přičemž každá obstarává nějakou funkci. Hlavní program je pak koncipován jako super-smyčka, která obsahuje i stavový automat s možnostmi, které mohou nastat. Viz obr.2.3

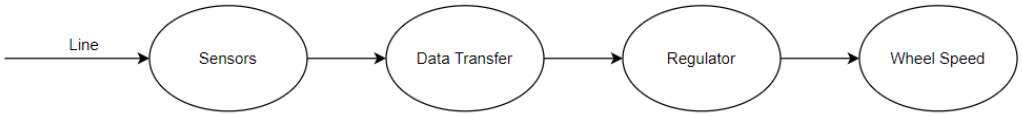


Obrázek 2.3: Stavový automat

Do budoucna zde budeme muset zaimplementovat možnost, že si budeme muset pamatovat přerušení čáry, jakmile dojedeme k rozvětvení budeme se muset rozhodnout, kterou větví se dát dál. Dle pravidel by měla být jednoduchší trasa vždy ta vpravo.

2.3 Způsob zpracování dat pro řízení

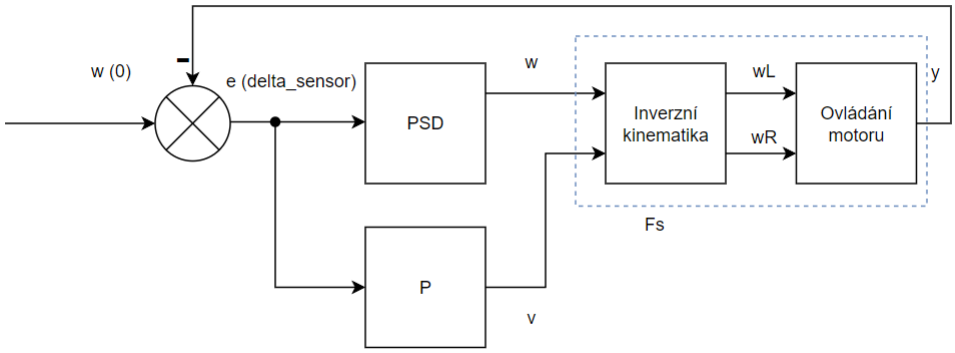
Ze simulace přijímáme data senzorů a ty pak zpracováváme podle blokového schématu uvedeného níže.



Obrázek 2.4: Blokové schéma zpracování dat

3 Návrh regulátoru

Pro ovládání robota byl využit PSD regulátor. Ten jsme realizovali ve třídě regulator. Celé regulační schéma vypadá dle obr.



Obrázek 3.1: Schéma regulační smyčky

Ze schématu vyplývá, že budeme regulovat jak úhel natočení, tak rychlost vozítka. Regulaci rychlosti jsme zatím ještě správně neimplementovali a tuto regulaci se pokusíme navrhnout pro vozítko na finálovou soutěž.

Pro návrh regulátoru jsme použili experimentální metody, kdy jsme postupně ladili parametry regulátoru.

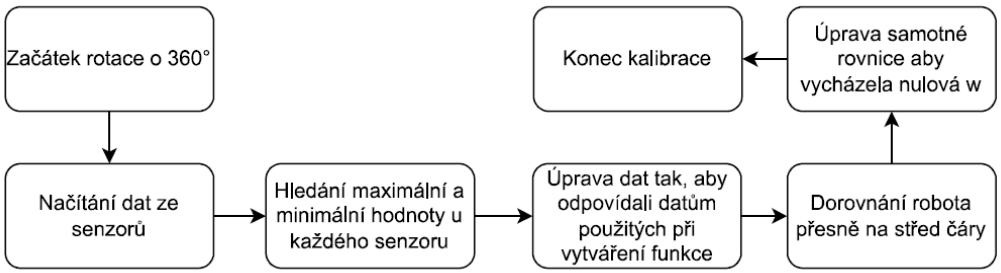
Listing 1: PSD regulátor

```
double Regulator::PSD(double delta_angle,
                      double velocity) {
    double sum = error.sum();
    double last_err = error.operator[](error.size()-1);
    error = error.shift(1);
    error.operator[](error.size()-1) = delta_angle;
    return PSD_K*(delta_angle+1/PSD_Ti*sum+PSD_Td*
                          (delta_angle-last_err));
}
```

4 Kalibrace senzorů

4.1 Teoretické zpracování

Jelikož nebyla kalibrace zabudována při zápočtovém zkoušení, tak je zde nastíněna zatím pouze teoreticky. Ale předpoklad kalibrace by byl že pro každý senzor naměříme maximální a minimální hodnotu. Tu následně porovnáme s daty, ze kterých jsme vytvářeli danou rovnici a následně poté s ohledem na to o kolik se budou lišit je budeme konstantou násobit aby se to nejlépe nové meření přiblížilo datům, ze kterých rovnice vycházela. Poté co se provede tento druh kalibrace nastane přesné narovnání na čáru. Toho by se dalo docílit díky prostřednímu senzoru, který je přesně umístěný na středu. Díky němu by jsme dokázali dokonale srovnat robota s čarou a poté následně modifikovat hodnoty rovnice tak, aby nám vycházela přesně nulová odchylka a robot pak nemusel ze startu hned vyregulovávat rovnou čáru. Díky tomu by se zabránilo hned prvnímu skoku, který by regulátor dostal ihned po samotném zapnutí.



Obrázek 4.1: Diagram představované kalibrace senzorů

4.2 Praktické zpracování

Praktické zpracování bude ještě dodělávano do finální zkoušky.

5 Detekce čáry a samotná jízda

5.1 Detekce čáry

Pro detekci čáry jsme si napsali vlastní proměnou s názvem **sensor_on**, která je umístěná ve tříde sensor, a je aktualizovaná pokaždě, kdy jsou čtena data ze senzoru.

Listing 2: Detekce čáry

```
 sensor_on[index] = val > 800;
```

Tento kus kódu je pak následně volán v main.cpp

Listing 3: Detekce přerušení čáry

```
bool l_break =  !sensor.sensor_on[0] &&
                !sensor.sensor_on[1] &&
                !sensor.sensor_on[2] &&
                !sensor.sensor_on[3] &&
                !sensor.sensor_on[4];
```

5.2 Jízda po čáře

Při jízdě po čáre dochází akorát k výpočtu rozdílu hodnot senzorů a tento rozdíl je následně posílán do regulátoru. Poté je výsledek regulace ukládán do bufferu. Následně je hned tato jedna hdnota psoílána do driveru, které řídí motory.

Listing 4: Výpočet diference senzorů

```
delta_sens = - sensor.sensor[0]
             + sensor.sensor[1]
             - 3.5*sensor.sensor[2]
             + 3.5*sensor.sensor[3];
diff = regulator.delta_to_angle(delta_sens);
state = 1;
```

Poté co je diference vypočtena je následně tato hodnota předána PSD regulátoru. Následně přichází na řadu samotné vyhodnocení a ukládání hodnot do bufferu a ovládání jednotlivých motorů.

Listing 5: Regulace a následně odesílání dat při jízde na čáře

```
w = regulator.PSD(diff, v);
buffer_w = buffer_w.shift(1);
buffer_w.operator[](buffer_w.size()-1) = w;
comm.Send(drive.BuildSpeed(v, w));
```

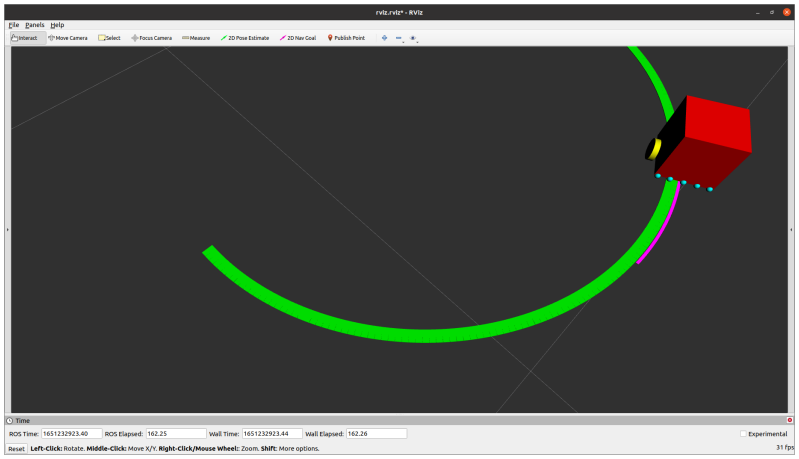
5.3 Jízda mimo čáru

Pro samotnou jízdu mimo čáru je využíván buffer, který je již zmíněn v kapitole 5.2. Pro dokonalé použití jsme se rozhodli posledních 5 měřených hodnot zanedbat a poté následně samotný průmer z těchto hodnot je využiván a konstantně držen do doby než senzory opět zaznamenají čáru.

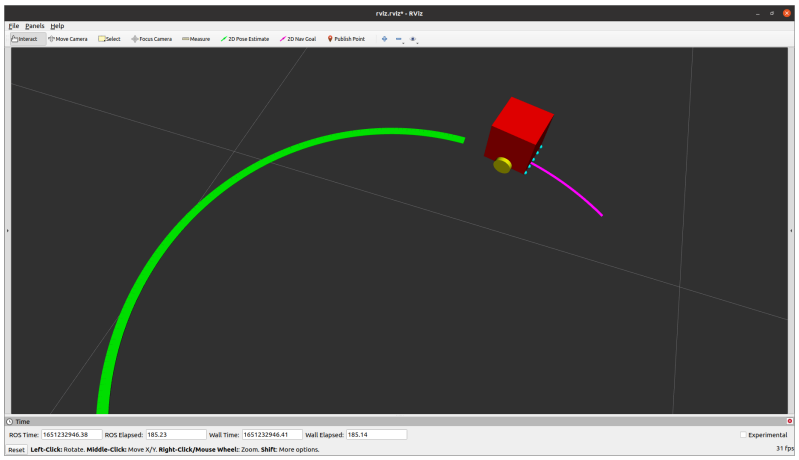
Listing 6: Výčet dat z buferru při jízde mimo čáru

```
double sum = 0;
for(int i = 0; i < buffer_w.size() - 5; i++)
    sum += buffer_w[i];
w = sum/(buffer_w.size()-5);
comm.Send(drive.BuildSpeed(v, w));
```

5.4 Ukázka jízdy



Obrázek 5.1: Jízda po čáře



Obrázek 5.2: Jízda mimo čáru

6 Závěr

V zadaném projektu jsme měli za úkol navrhnout řízení a umístění čidel na sledovači čar. Po zdlouhavém testování jsme se nakonec rozhodli pro umístění 5 čidel na přední straně vozítka s výškou 1 cm od čáry. Toto uspořádání nám umožňuje detekovat i ostřejší zatáčky a tím uzpůsobit akční zásah. Pro regulování jsme pak použili PSD regulátor, který jsme experimentální metody vyladili. Konstanty pak mají hodnoty:

1. K= 0.01
2. Ti = 1
3. Td = 0.1

Aktuálně se projekt nachází ve stavu, že náš sledovač čar zvládne projet elementární trasy. Zbývá nám však ještě doimplementovat kalibraci senzorů a ošetření možných stavů, které mohou nastat při finálové soutěži (např. rozdvojení trasy).

Celý projekt včetně zdrojových souborů se nachází v Git repozitáři.