Dokumentácia- Lokalizácia a polohovanie robota v prostredí

Úloha

Úlohou lokalizácie je povedať, kde v priestore sa mobilný robot nachádza. Najjednoduchší spôsob lokalizácie mobilného kolesového robota s diferenciálnym podvozkom je odometria. Odometria je typ relatívnej lokalizácie (určuje polohu voči predchádzajúcej polohe), ktorá prírastok polohy určuje na základe otočenia kolies. Keďže robot Kobuki má v sebe aj gyroskop môžete natočenie robota spresniť a vytvoriť takzvanú gyroodometriu.

Riadenie pohybu robota

Robot je vedený tak, aby sledoval sériu bodov x a y definovaných premennou qxr a qyr. Program berie dáta senzorov z encoderov a gyroskopu robota a vykonáva výpočty na aktualizáciu polohy robota (x, y), orientácie (rads) a požadovaného smeru (fi).

(Niektoré časti kódu nie sú kompletne zobrazené na obrázkoch lebo by sa sem nezmestili a dokumentácia by bola neprehľadná)

Princíp algoritmu:

- Počiatočný blok kódu vo vnútri bloku if(štart) nastavuje body, ktoré má robot sledovať. Do queue qxr a qyr sa načítajú súradnice x a y.

```
118     if(start){
119       /* qxr.push(5.00);
120        qyr.push(0.00);*/
121
122        qxr.push(0);
123        qxr.push(2.70);
124        qxr.push(2.70);
125        qxr.push(4.6);
126        qxr.push(4.6);
127
128        qyr.push(2.90);
129        qyr.push(2.90);
130        qyr.push(0.0);
131        qyr.push(0.0);
132        qyr.push(1.6);
133        start = false;
134     }
```

Obr. 1

- Ďalší blok kódu kontroluje náhle zmeny v údajoch z ľavého a pravého encodera robota. Ak je takáto zmena zistená, vykoná úpravu na predchádzajúce čítanie encodera na zvládnutie prevrátenia v encodery. Potom vypočíta vzdialenosť prejdenú každým kolesom od poslednej kontroly na základe rozdielu v počtoch kódovačov.

```
148     if(abs(previousEncoderLeft - robotdata.EncoderLeft) > 10000){
149         printf("\nLeft encoder pretec\n");
150         if(previousEncoderLeft > robotdata.EncoderLeft){
151             previousEncoderLeft -= 65535;
152         }else if(previousEncoderLeft < robotdata.EncoderLeft) previousEncoderLeft += 65535;
153 
154     }
155     if(abs(previousEncoderRight - robotdata.EncoderRight) > 10000){
156         printf("\nRight encoder pretec\n");
157         if(previousEncoderRight > robotdata.EncoderRight){
158             previousEncoderRight -= 65535;
159         }else if(previousEncoderRight < robotdata.EncoderRight) previousEncoderRight += 65535;
160 
161     }
162 
163     float rightWheel = tTM*(robotdata.EncoderRight - previousEncoderRight);
164     float leftWheel = tTM*(robotdata.EncoderLeft - previousEncoderLeft);
165 
```

Obr. 2

- V závislosti od toho, či sa robot pohybuje priamočiaro alebo nie, sa orientácia robota aktualizuje buď na základe údajov z gyroskopu alebo na základe rozdielu vzdialenosti prejdenej ľavým a pravým kolesom. Podobne ako v prednáške z úlohy 1.

```
177 
178         if(translation == 0){
179             rads = (robotdata.GyroAngle/100.0) * (pi1/180.0);
180             //if(rads < 0) rads += 6.283185;
181         }else {
182             rads += (rightWheel - leftWheel)/diameter;
183             if(rads > (6.283185/2)){
184                 rads = -(6.283185/2) + (rads-(6.283185/2));
185             }else if(rads < -(6.283185/2)){
186                 rads = (6.283185/2) + (rads + (6.283185/2));
187             }
188         }
189 
```

Obr. 3

- Poloha x a y robota sa potom aktualizuje na základe vzdialenosti prejdenej kolesami a orientácie robota.

```
198         if((rightWheel - leftWheel) != 0){
199             x += ((diameter*(rightWheel + leftWheel)) / (2.0*(rightWheel - leftWheel)))*(sin(rads) - sin(previousRads));
200             y -= ((diameter*(rightWheel + leftWheel)) / (2.0*(rightWheel - leftWheel)))*(cos(rads) - cos(previousRads));
201 
202         }else{
203             x += ((rightWheel + leftWheel)/2.0)*cos(rads);
204             y += ((rightWheel + leftWheel)/2.0)*sin(rads);
205 
206         }
```

Obr. 4

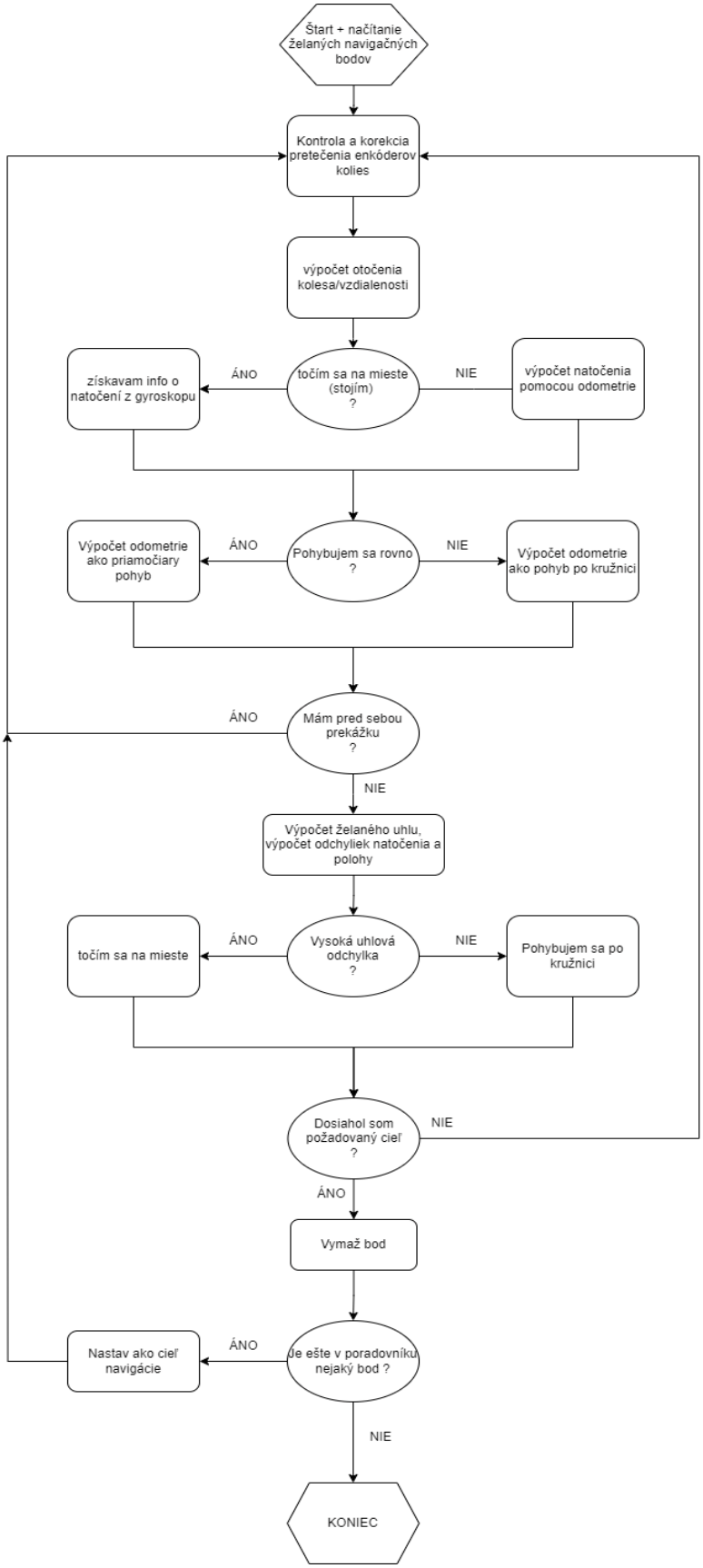
- V ďalšej časti kódu ide robot smerom k ďalšiemu bodu. Vypočíta sa požadovaný kurz smerom k bodu trasy (fi) a na základe rozdielu medzi týmto požadovaným kurzom a aktuálnym kurzom robota sa upraví rýchlosť robota. Ak je smerovanie robota posunuté o viac ako určitú hranicu, robot spomalí a upraví svoje smerovanie predtým, ako sa vráti do pohybu. Inač povedané, vykonáva kontrolu spätnej väzby PID algoritmom.

```
213         ///POLOHOVANIE
214         if(!mojRobot.stop){
215 
216             if(!qyr.empty()){
217                 yr = qyr.front();
218                 xr = qxr.front();
219             }
220             double finish = 0.1;
221             /*if (qyr.size() > 1){
222                 finish = 0.2;
223             }*/
```

Obr. 5

- Nakoniec, v poslednom bloku kódu je stav núdzového zastavenia, pri ktorom môže byť pohyb robota náhle zastavený.

Diagram Algoritmu



Obr. 6 Diagram algoritmu