Izdelava mobilnega robota z razvojnim sistemom roboPIC (12)

Avtor: Silvan Bucik
Forum: http://www.svet-el.si/phpBB2/index.php

Pretekli mesec smo pričeli s študijo gibanja robota vzdolž ravne stene. Spoznali smo vlogi nastavitev stranskih senzorjev ter odprli vprašanje hitrosti jemanja podatkov s senzorjev. Počasi ugotavljamo, da so vsi ti parametri, vključno z dimenzijami robota in hitrostjo potovanja, med seboj tesno povezani in jih ne moremo obravnavati ločeno. V tokratnem branju pa nas čakajo vožnje skozi ovinke ...

Rešitve »domačih nalog«

Za začetek si poglejmo rešitvi nalog iz 130. številke.

Napišite program, ki bo izvajal testiranje vseh šestih senzorjev.
Naloga ni nič posebnega, le dobra vaja za osvežitev znanja.

```
// DEKLARACIJE IN DEFINICIJE
#define BITNUM(adr,bit) ((unsigned)(&adr)*8+(bit))
#define                 frekvenca_kHz 1000

bit IRL_0 @ BITNUM(PORTD,0);
bit IRL_1 @ BITNUM(PORTD,7);
bit IRL_2 @ BITNUM(PORTD,5);
bit IRL_3 @ BITNUM(PORTD,4);
bit IRL_4 @ BITNUM(PORTD,3);
bit IRL_5 @ BITNUM(PORTD,2);

bit SEN_0 @ BITNUM(PORTA,2);
bit SEN_1 @ BITNUM(PORTA,4);
bit SEN_2 @ BITNUM(PORTA,5);
bit SEN_3 @ BITNUM(PORTE,0);
bit SEN_4 @ BITNUM(PORTE,1);
bit SEN_5 @ BITNUM(PORTE,2);

int frq=frekvenca_kHz;
char senzor_stat;


// INICIALIZACIJA

void main (void)

{
TRISA = 0B11111111;     // PORTA = vhod, senzorji
ADCON1 = 0x07;          // PORTA definiramo kot digitalna vhodno-
                           izhodna vrata
TRISB = 0B11000000;     // RB0-RB5 so izhodi (led diode)
TRISC = 0B11111111;     // PORTC = vhod, ga ne uporabljamo
TRISD = 0B01000010;     // RD0, RD7, RD5, RD4, RD3, RD2 so izhodi
                           (IR led diode)
TRISE = 0B00000111;     // PORTE = vhod, senzorji
PORTD = 0B00000000;     // vse IR led diode so ugasnjene
PORTB = 0B11111111;     // vse led diode na PORTB-ju so ugasnjene
```

```
//GLAVNI PROGRAM

for (;;)
{
senzor_stat = 0B00000000;

IRL_0 = 1;              // vklopimo IR led diodo
                           wait_mili(1);
                        // počakamo, da se izhod senzorja ustali
if (SEN_0==0)
{
senzor_stat=(1<<0)|senzor_stat;
                        // če senzor vidi, se na ustrezno mesto
}                       // v spremenljivki senzor postavi logična 1
IRL_0 = 0;              // izklopimo IR led diodo
IRL_1 = 1;
wait_mili(1);
if (SEN_1==0)
{
senzor_stat=(1<<1)|senzor_stat;
}
IRL_1 = 0;
IRL_2 = 1;
wait_mili(1);
if (SEN_2==0)
{
senzor_stat=(1<<2)|senzor_stat;
}
IRL_2 = 0;
IRL_3 = 1;
wait_mili(1);
if (SEN_3==0)
{                       senzor_stat=(1<<3)|senzor_stat;
}
IRL_3 = 0;
IRL_4 = 1;
wait_mili(1);
if (SEN_4==0)
{
senzor_stat=(1<<4)|senzor_stat;
}
IRL_4 = 0;

IRL_5 = 1;
```

```
wait_mili(1);
if (SEN_5==0)
{
senzor_stat=(1<<5)|senzor_stat;
}
IRL_5 = 0;
PORTB = ~senzor_stat;     // iz registra senzor_stat se prenesejo
                          // negirane vrednosti na PORTB, kjer led
                             diode indicirajo stanje senzorjev
                          // POZOR: led dioda sveti ob aktivni
                             logični 0
}
}
```

Spremenite program iz prejšnje naloge tako, da se bo izvajalo testiranje le treh senzorjev, in sicer senzorja 1, senzorja 3 in senzorja 5. Do rešitve boste prišli že z ustrezno nastavitvijo registra TRISD.

Ideja je odlična. Ohranili bomo formo programa, s to razliko, da v register TRISD postavimo logične 0 le na tista mesta, katera omogočajo delovanje izbranih IR led diod: TRISD7 (IRL_1), TRISD4 (IRL_3), TRISD2 (IRL_5). Razmišljanje pa je takšno: če želimo onemogočiti delovanje IR led diod, potem je čisto dovolj, da njihove nadzorne priključke v TRIS registru določimo kot vhode. Program, ki smo ga napisali v prejšnjih vrsticah, pa lahko ohranimo nespremenjen.

```
......;
TRISB = ......;
TRISC = ......;
TRISD = 0B01101011;       // RD7, RD4, RD2 so izhodi
TRISE = ......;
......;
                          // ves ostali program ostane nespremenjen
```

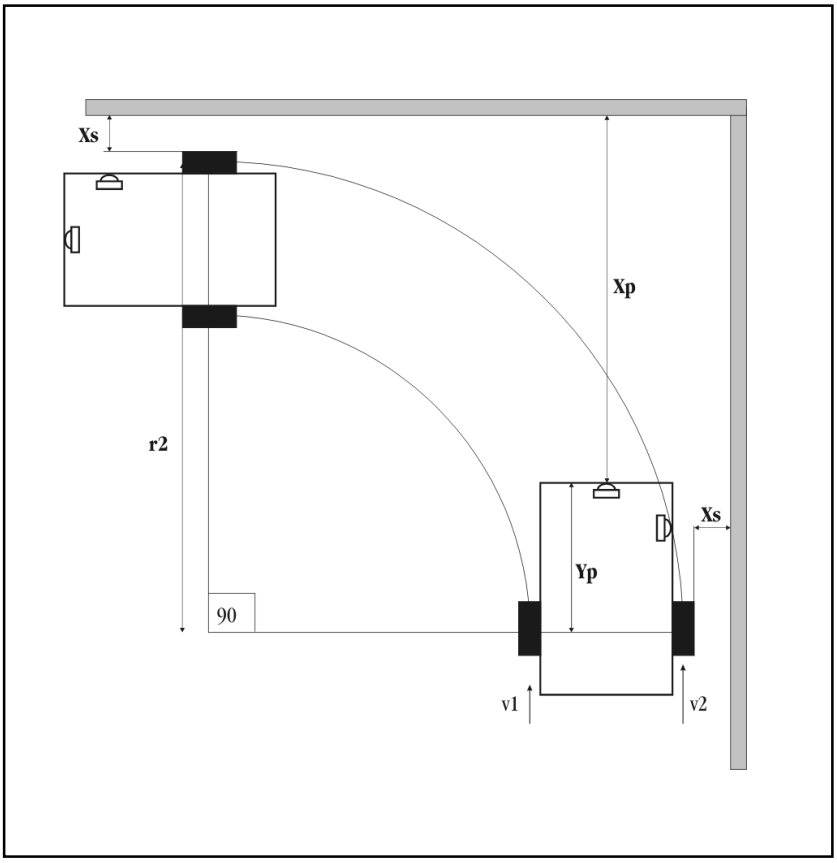
Dopolnite program tako, da boste upoštevali vse možne kombinacije stanj treh senzorjev. Napišite program za aplikacijo, ki za upravljanje robota uporablja servomotor.

Branje senzorjev smo si že pogledali, sedaj pa si oglejmo, kako bi robota poganjali s kombinacijo DC motorja kot pogona in servomotorja, ki bi nam služil kot krmilo. DC motor poganjamo ves čas z maksimalno močjo (moč lahko tudi zmanjšamo z uporabo PWM generatorja), spreminjali bomo le pozicijo krmila.

```
/*
Posamezni biti v registru senzor_stat pomenijo:
- bit 0: desni kratki senzor
- bit 1: desni dolgi senzor
- bit 2: sredinski - prednji senzor
*/

int frq=frekvenca_kHz;
char senzor_stat;

bit mot1a @ BITNUM(PORTC,0),
   mot1b @ BITNUM(PORTC,3),
   mot1_en @ BITNUM(PORTC,2);
// INICIALIZACIJA
void main (void)
{
TRISA = 0B11111111;       // PORTA = vhod, ga ne uporabljamo
ADCON1 = 0x07;            // PORTA definiramo kot digitalna vhodno-
                             izhodna vrata
TRISB = 0B11111111;       // PORTB = vhod, ga ne uporabljamo
TRISC = 0B11110000;       // RC0 - RC6 so izhodi
TRISD = 0B11111111;       // PORTD = vhod, ga ne uporabljamo
TRISE = 0B00000111;       // PORTE = vhod, ga ne uporabljamo

mot1_en = 1;              // omogoceno delovanje int. vezja L293

//GLAVNI PROGRAM
switch (senzor_stat)
{
case 0B00000000: servo2_out(102);
                          // ROBOT na desno
 mot1a=1;                 // DC motor naprej
 mot1b=0;
 break;
case 0B00000010: servo2_out(77);
                          // ROBOT naprej
 mot1a=1;                 // DC motor naprej
 mot1b=0;
 break;
case 0B00000011: servo2_out(51);
                          // ROBOT na levo
 mot1a=1;                 // DC motor naprej
 mot1b=0;
 break;
case 0B00000100: servo2_out(102);
                          // ROBOT na desno
 mot1a=1;                 // DC motor naprej
 mot1b=0;
 break;
case 0B00000110: servo2_out(51);
                          // ROBOT na levo
 mot1a=1;                 // DC motor naprej
 mot1b=0;
 break;
case 0B00000111: servo2_out(51);
                          // ROBOT na levo
 mot1a=1;                 // DC motor naprej
 mot1b=0;
 break;
default:
 mot1a=0;                 // ROBOT stop
 mot1b=0;                 // DC motor stop
 break;
}
stop:
goto stop;
}
```

Vožnja v ovinek ob sledenju stene

Ovinek ob sočasnem sledenju stene običajno robotom ne predstavlja večjih preglavic, vsaj pri nizkih hitrostih ne. Ko pa tekmovalne hitrosti postanejo večje, je potrebno upoštevati tudi možnost zdrsa koles. Kakorkoli že, »umetniških likov« z ostrimi zavoji ni pametno izvajati. Pri študiji vožnje skozi ovinek ob istočasnem sledenju stene bomo iskali idealni radij loka zavijanja (r2). Ta naj bi bil čim daljši, saj bo takrat ovinek manj oster (centrifugalna sila bo manjša in manjša bo možnost zdrsa koles), hitrost vožnje pa bo



Slika 1: Vožnja dvokolesnika v ovinek ob sočasnem sledenju steni

Ob upoštevanju enačbe

$$r2 = \frac{v2 \cdot Tv}{\Delta\varphi}$$

pridemo do končnega rezultata

$$Xp = \frac{v2 \cdot Tv}{\Delta\varphi} + Xs - Yp$$

Hitrost drugega kolesa v1 dobimo iz enačbe:

$$\Delta\varphi = \frac{(v2 - v1) \cdot Tv}{A}$$

ter pridemo do končnega zaključka:

$$v1 = v2 - \frac{\Delta\varphi \cdot A}{Tv}$$

Kot vidimo vodljivost robota v ovinku ni odvisna le od nastavitev senzorjev, ampak tudi od hitrosti vožnje, kar je nekako tudi razumljivo.

lahko večja. Pomembno vlogo pri vodenju skozi ovinek nosi prednji senzor, katerega nastavitev določa trenutek pričetka zavijanja.

Dvokolesnik

Idealni radij loka je tisti, ki bo po izpeljanem ovinku pripeljal robota na optimalno razdaljo od stene (Xs).
Takrat bo veljalo:

$$r2 + Xs = Yp + Xp$$

(Xs predstavlja srednjo oddaljenost od bočne stene)

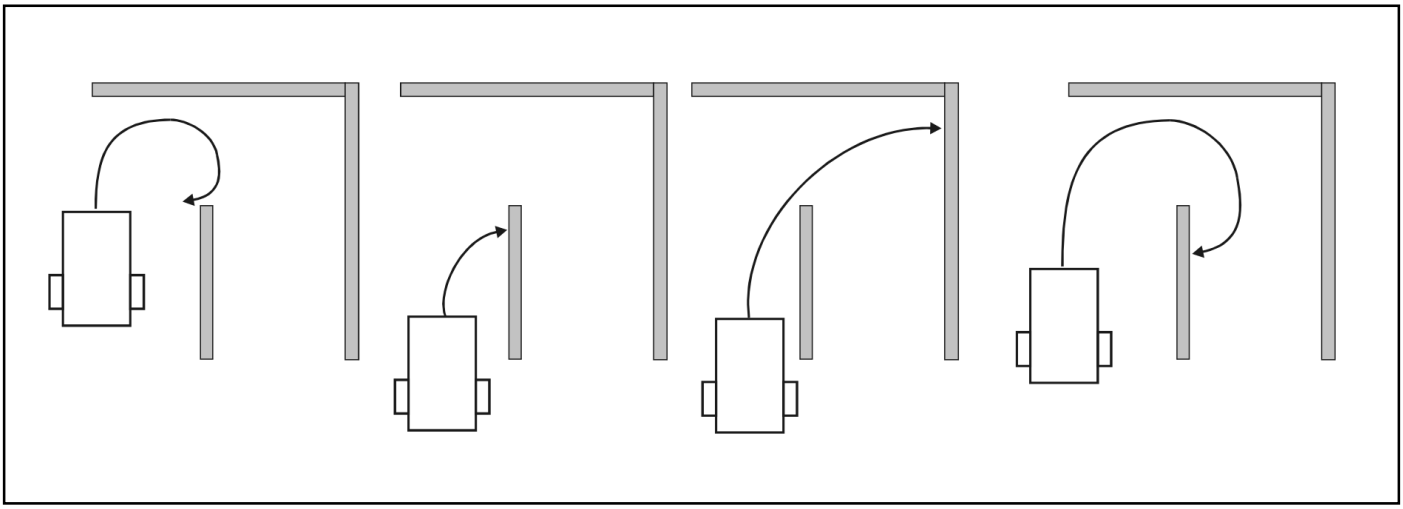
Enokolesnik

Poglejmo si še vožnjo enokolesnika. Za izhodišče privzamemo že znano ugotovitev:

$$r2 + Xs = Yp + Xp$$

Upoštevajoč enačbo:

$$tg\,a = \frac{m}{r2 - \frac{A}{2}}$$



Slika 2: Nevarnosti pri vožnji robota skozi ovinek ob nepopolnem sledenju steni

IR Electronic d.o.o.
Ziherlova ulica 2,
1000 Ljubljana, Slovenija
Tel.: 1.2835604,
Fax: 1.2835605
Internet: www.irelectronic.si
E-mail: info@irelectronic.si

Telit Communication

Prvič vse v eni enoti

Strokovnjaki za vmesnike M2M so uspeli GSM/GPRS in GPS združiti v enem modulu

Prva stroškovno optimirana rešitev s širokimi možnostmi uporabe

GM862-GPS Modem

- **SiRF** Powered**
- **20-Channel High Sensitivity GPS Receiver**
- **Pin to Pin backward compatible**
- **GSM Quad Band**
- **RoHS Compliant**
- **On Board SIM Holder**
- **Embedded TCP/IP Stack**
- **GPRS Class 10**
- **PYTHON***
- **Script Interpreter**
- **Embedded FTP and SMTP Client**
- **Extended Temperature Range**
- **Extended RF Sensitivity**

Telit kot eden prvih ponudnikov ponuja module, ki združujejo tehnologije GSM/GPRS in GPS v izredno majhnem ohišju. Rešitev podjetja Telit pomeni, da lahko zdaj aplikacije, ki so do zdaj zahtevale več modulov, vse funkcije uporabljajo z enim samim modulom. To zmanjša stroške in čas pri integraciji modula ter proizvodnji končnih naprav. Tehnologija GPS tako postaja privlačna tudi za velikoserijske specializirane aplikacije, na primer na trgu zabavne elektronike.

- **GPS žepne velikosti**
- **Široko območje uporabe**
- **Moduli za različne zahteve**
- **Navigacija v realnem času, tudi v težkih pogojih**
- **Izboljšana zmogljivost s tehnologijo SiRFstarIII**

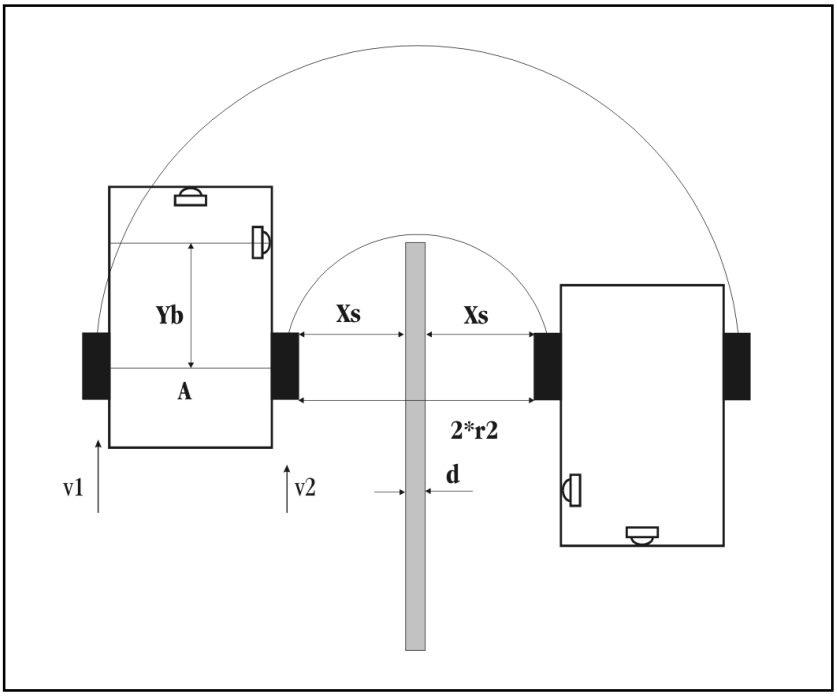
GE863-GPS Embedded

Telit GM862-GPSGE863-GPS Embedded

- **SiRF** Powered**
- **BGA Package**
- **20-Channel GPS Receiver**
- **High Sensitivity**
- **GSM Quad Band**
- **RoHS Compliant**
- **Embedded TCP/IP Stack**
- **GPRS Class 10**
- **PYTHON***
- **Script Interpreter**
- **Embedded FTP and SMTP Client**
- **Extended Temperature Range**
- **Extended RF SensitivityGE863-GPS Embedded**

Več informacij:
Bojan Zaletelj FSE
(Prodaja 040 88 66 27)

Armin Čatak DFAE
(Tehnična podpora 040 88 66 31)



Slika 3: Idealno izpeljan ovinek

pridemo do naslednjega rezultata:

$$r2 = Xs + \frac{d}{2}$$

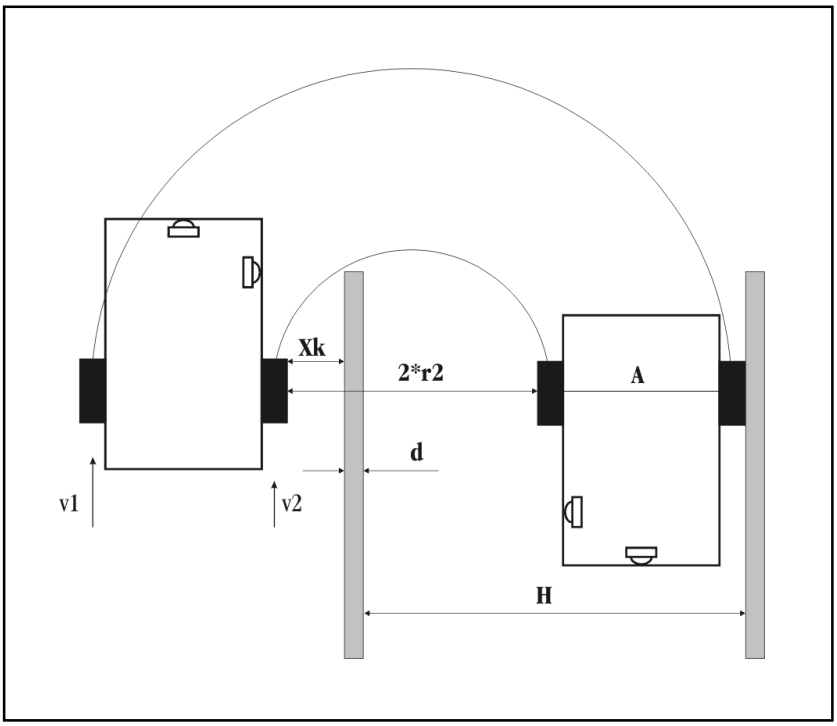
Če si dobro ogledamo dobljeni rezultat, opazimo, da na vožnjo skozi ovinek ne vplivata ne hitrost pogonskega motorja niti hitrost vzorčenja. Edina prava spremenljivka je poleg dimenzijskih značilnosti (m, A) in nastavitev senzorjev (Xs, Yb) le kot zasuka servomotorja (a). Servomotor opravlja podobno funkcijo kot jo ima volan pri avtomobilu. Z njunim zasukom direktno vplivamo na krivuljo zavijanja in je tako praktično vseeno s kakšno hitrostjo se robot vozi skozi ovinek (le zdrsniti ne sme). Res pa je tudi, kar vozniki dobro vedo, da so avtomobili na prednji pogon bolje vodljivi kot ostali na zadnji pogon.

Ovinek ob nepopolnem sledenju stene - slepo zavijanje

Pri vožnji skozi ovinek ob nepopolnem sledenju stene robot za kratek čas izgubi stik s stransko steno. Pri vožnji skozi ovinek te vrste gre za neko vrsto varanja. V situaciji, ko robot izgubi steno izpred oči, prične iskati stransko steno, misleč, da se je od nje že preveč oddaljil. Tako se počasi pripelje okoli stranice, ne da bi robot »vedel«, da je s tem manevrom uspel prevoziti ovinek. V tem trenutku pa je robot najbolj ranljiv. Pri izvajanju manevra je zelo pomembno, s kakšno krivuljo se vožnja skozi ovinek izvede. Sedaj robot pred seboj nima stene, ki mu je bila v oporo pri izbiri primernega trenutka pričetka zavijanja. V prejšnjem načinu je robot med vožnjo skozi ovinek lahko ves čas »tipal« steno. Ovinek te vrste pa mora robot »na slepo« izpeljati.

Na tem segmentu je potrebno precej časa nameniti umerjanju senzorjev in preizkušanju. Od konstruktorja zahteva veliko mero potrpežljivosti, predvsem pa vztrajnosti. Čeprav se problem slepega zavijanja rešuje v glavnem po načelu poskusa in napake, pa lahko kljub temu o tem spregovorimo nekaj besed.

Idealno izpeljavo zavoja prikazuje gornja slika. Robot prične in zaključi zavoj na optimalni - srednji oddaljenosti od stene (Xs). Radij loka tu je odvisen od nastavitve bočnega senzorja:



Slika 4: Vožnja skozi ovinek ob upoštevanju širine hodnika

$$r2 = Xs + \frac{d}{2}$$

Pravzaprav nas radij loka krivulje ne zanima neposredno. Za nas sta gotovo bolj zanimiva vzročna parametra, kot sta hitrosti obeh motorjev (v2 in v1)

$$v2 = \frac{r2 \cdot \Delta\varphi}{Tv}$$

$$v1 = \frac{(r2 + A) \cdot \Delta\varphi}{Tv}$$

pri enokolesniku pa kot zasuka servomotorja (a)

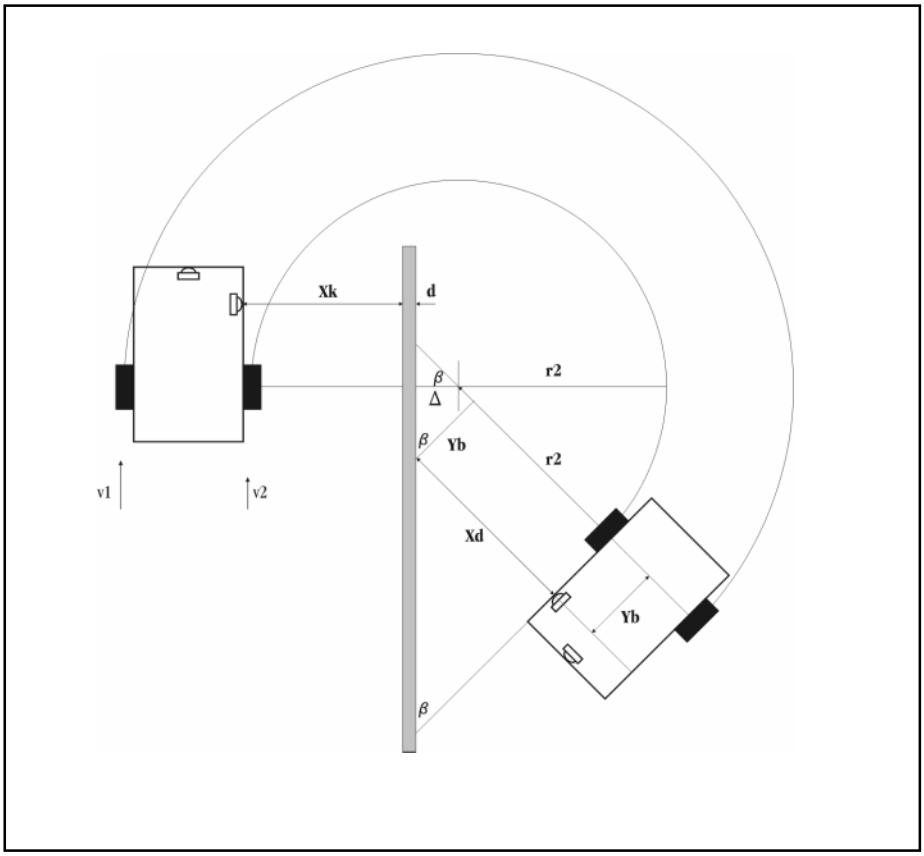
$$tga = \frac{m}{r2 + \frac{A}{2}}$$

Ti rezultati veljajo le pri optimalni izpeljavi zavoja. Optimalna izpeljava pomeni, da robot prične izpeljavo zavoja pri srednji oddaljenosti od stene (Xs)

ter pri enaki oddaljenosti zavoj tudi zaključi. Pri tem je potrebno določiti ustrezni radij krivulje (r2) ter zavoj tudi ustrezno časovno uskladiti. Ključno vlogo pri tem odigra razdalja med pozicijo namestitve bočnega senzorja in osjo zadnjih koles (Yb). Če je ta razdalja, glede na nastavitve bočnega senzorja, predolga, obstaja nevarnost, da bo robota med izpeljavo ovinka zavrtelo za kot, večji kot 180° (zavrtel bi se naokrog), preden bi spet uspel zaznati stransko steno (Slika 2 prvi primer). Ta scenarij nikakor ne pride v poštev.

V nasprotnem primeru, ko je razdalja med bočnim senzorjem in osjo zadnjih koles (Yb) prekratka, bo robot med vožnjo skozi ovinek takoj naletel na stransko steno (Slika 2 drugi primer) in če ne bo imel posebne sreče, bo tam tudi obtičal. To težavo je moč rešiti s tem, da se radij krivulje (r2) ustrezno podaljša. Vendar pa se tu skrivata dve težavi. Prva nevarnost je nalet na stransko steno zaradi preozkega hodnika oziroma predolgega radija zavoja (r2) (Slika 2 tretji primer).

Pri reševanju nastale težave si pomagamo z zgornjo sliko. Iz gornje slike razberemo, da velja:



Slika 5: Vožnja skozi ovinek ob upoštevanju vpadnega kota po izpeljanem ovinku

$$2r2 = Xk + d + H - A$$

Potemtakem znaša radij loka (r2) največ

$$r2 = \frac{Xk + d + H - A}{2}$$

Naslednjo težavo predstavlja vpadni kot, pod katerim se robot približa stranski steni po izdelanem zavoju. Nevarno je, če se med vožnjo robot preveč zasuče in prileti na stransko steno pod kotom, večjim kot 90° (Slika 2 četrti primer). Posledica tega je obrat robota in nadaljevanje vožnje v smeri proti startni točki. Tu je optimalne nastavitve nekoliko težje določiti.

Pomagamo si z gornjo sliko. Iz nje razberemo naslednji relaciji:

$$tg\beta = \frac{r2 + \frac{\Delta}{\cos\beta}}{Yb + \frac{Xd}{tg\beta}}$$

$$r2 = Xk + d + \Delta$$

Iz prve enačbe izrazimo D ter ga vstavimo v drugo enačbo. Po nekaj matematičnih akrobacijah dobimo končni rezultat:

$$r2 = \frac{Yb \cdot \sin\beta + Xd \cdot \cos\beta + Xk + d}{1 + \cos\beta}$$

Tukaj ni kaj dodati. Kot končna enačba pove, da je ovinek »na slepo« zahtevna prvina, ki se jo mora robot »naučiti«. Veliko eksperimentiranja bo potrebno, preden bo robot tekoče izdeloval tovrstne zavoje.

Konec teorije? Zaenkrat ...

Dobljeni rezultati nam bodo v pomoč pri izbiri pogonov, oblike konstrukcije, namestitvi senzorjev in njihovem umerjanju. Vsekakor pa so izračunane vrednosti le okvirnega značaja. Na koncu je vse skupaj vendarle potrebno preizkusiti tudi v praksi in po potrebi katero od nastavitev tudi spremeniti. Za konec naj omenimo, da je vsa opisana teorija zbrana v obliki Excelove datoteke, ki jo najdete na spletnih straneh revije Svet elektronike. Vse informacije boste našli na straneh foruma. ●

Literatura:

[1] Microchip, PIC16F87x data sheet, Microchip Technology Incorporated, 2001, http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf;
[2] B. Peršič, Gradnja mikroprocesorskih sistemov, Ljubljana: Fakulteta za elektrotehniko, 1998,
[3] HI-TECH, PICC Lite C Manual, HI-TECH Software, 2002, http://www.htsoft.com/downloads/manuals.php;
[4] B.W. Kernighan, D.M. Ritchie, Programski jezik »C«, Ljubljana: Fakulteta za elektrotehniko, 1991.