Izdelava mobilnega robota z razvojnim sistemom roboPIC (10)

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

Delovanje senzorjev, takšnih in drugačnih, smo dodobra spoznali. Danes je prišel čas, da se naši roboti pričnejo sprehajati. Pogledali si bomo postopek izvedbe meritev ter vodenje enosmernih motorjev. Prvi koraki so počasni in okorni, tako je tudi pri robotih. Nato pa ... gre vsak dan hitreje.

Rešitve »domačih nalog«:

Za začetek si poglejmo rešitvi nalog iz 128. številke.

Napišite program, ki bo prižgal led diodo na priključku RB2, če bo vrednost rezultata razdalja, večja kot 0x02AB (število 0x02AB predstavlja šestnajstiški zapis decimalnega števila 683).

Naloga ne bi smela biti posebno težka. Najprej bomo izvedli meritev in rezultat shranili v spremenljivko razdalja. V naslednjem koraku preverimo, ali je rezultat večji od 0x02AB, ter prižgemo (ugasnemo) led diodo.

```
// DEKLARACIJE IN DEFINICIJE

#define BITNUM(adr,bit) ((unsigned)(&adr)*8+(bit))
#define                 frekvenca_kHz 1000

int frq=frekvenca_kHz;
int razdalja;
bit sen_en1 @ BITNUM(PORTD,6);
                        // priključku RD6 priredimo ime sen_en1
bit ledica @ BITNUM(PORTB,2);
                        // priključku RB2 priredimo ime ledica
                        // INICIALIZACIJA

void main (void)

{
TRISA = 0B11111111;     // PORTA = vhod (tudi RA0, RA1 in RA3)
TRISB = 0B11111011;     // RB2 = izhod
TRISC = 0B11111111;     // PORTC = vhod, ga ne uporabljamo
TRISD = 0B10111111;     // RD6 = izhod
TRISE = 0B00000111;     // PORTE = vhod, ga ne uporabljamo

// GLAVNI PROGRAM

for(;;)
{
sen_en1 = 1;            // omogočimo delovanje senzorja
wait_mili(3);           // počakamo na meritev
razdalja = get_ad(AN1,VCC);
                        // izberemo kanal št. 1 (AN1)
                        // in referenčno napetost 5V (VCC)
sen_en1 = 0;            // onemogočimo delovanje senzorja
if(razdalja>0x02AB)     // če je rezultat večji od 0x02AB
{
ledica=0;               // se dioda prižge
}
else
                        // v nasprotnem primeru
                        {
ledica=1;               // dioda ugasne
}
```

Napišite program, ki bo izpisal gornjih 8 bitov merilnega rezultata na priključne sponke vrat PORTB.

Malce težja naloga. Po izvršeni meritvi je potrebno merilni rezultat preoblikovati v obliko, primerno za izpis: rezultat najprej zaokrožimo na 8-bitno vrednost, nato sledi negiranje (vzrok so led diode, ki svetijo ob aktivni 0 na izhodih) in izpis.

```
// DEKLARACIJE IN DEFINICIJE

#define BITNUM(adr,bit) ((unsigned)(&adr)*8+(bit))
#define                 frekvenca_kHz 1000

int frq=frekvenca_kHz;
int razdalja;
bit sen_en1 @ BITNUM(PORTD,6);
                        // priključku RD6 priredimo ime sen_en1

// INICIALIZACIJA
void main (void)
{
TRISA = 0B11111111;     // PORTA = vhod (tudi RA0, RA1 in RA3)
TRISB = 0B00000000;     // POTRB = izhod
TRISC = 0B11111111;     // PORTC = vhod, ga ne uporabljamo
TRISD = 0B10111111;     // RD6 = izhod
TRISE = 0B00000111;     // PORTE = vhod, ga ne uporabljamo

// GLAVNI PROGRAM

for(;;)
{
sen_en1 = 1;            // omogočimo delovanje senzorja
wait_mili(3);           // počakamo na meritev
razdalja = get_ad(AN1,VCC);
                        // izberemo kanal št. 1 (AN1)
                        // in referenčno napetost 5V (VCC)
sen_en1 = 0;            // onemogočimo delovanje senzorja
razdalja = razdalja>>2; // rezultat pripravimo za izpis
                           razdalja = ~razdalja;
```

```
                    // negiramo; led diode gorijo ob aktivni 0
PORTB = razdalja;   // izpišemo vrednost na vrata PORTB
}
```

Branje senzorjev in signalizacija

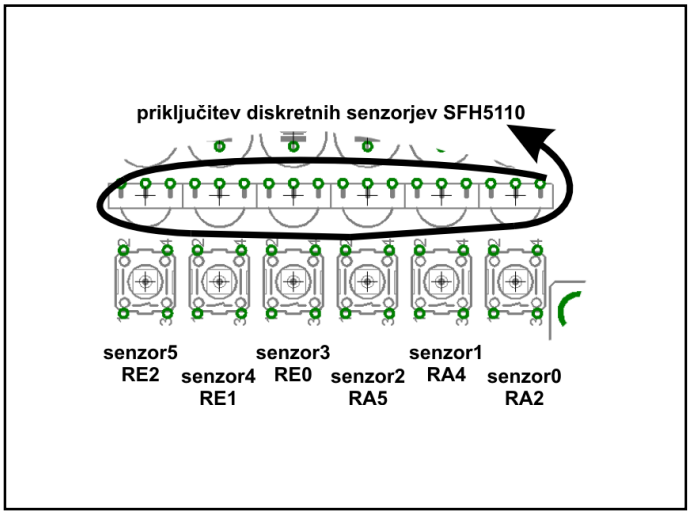
Sedaj, ko smo delovanje senzorjev v celoti spoznali, lahko pričnemo z njihovo uporabo. Senzorji so ključni elementi v procesu vodenja robota. Ti robotu podajajo potrebno informacijo o prostoru, ki se jo upošteva v programskem algoritmu, kjer se tudi odloča o načinu vodenja robota. Ogledali si bomo način preverjanja večjega števila diskretnih senzorjev in shranjevanje dobljenih informacij. Z vodenjem robota se bomo ukvarjali v naslednjem koraku.

V aplikaciji lahko uporabimo poljubno število senzorjev z diskretnim izhodom. Vendar je tu potrebno poudariti, da večje število senzorjev pomeni tudi veliko količino podatkov, katere je kasneje težje obdelati in smiselno uporabiti. Slednje lahko načrtovalcu predstavlja kar hud zalogaj. Drugo težavo predstavlja čas, ki je potreben za izvedbo vseh meritev. Čas vzorčenja ne sme biti predolg. Če med vzorčenjem robot opravi predolgo pot, potem se napake, ki se med vodenjem vedno pojavljajo, pričnejo potencirati. Takrat pravimo, da je postal robot nevodljiv. Za rešitev težave obstajata dve rešitvi: ali pospešiti hitrost merjenja ali pa upočasniti gibanje robota. Hitrost merjenja žal lahko pospešimo le na en način - zmanjšamo število uporabljenih senzorjev za toliko, da postane čas vzorčenja zadovoljivo kratek. Druga možnost je uporaba senzorjev z analognim izhodom ..., pustimo uporabo te možnosti za kdaj drugič.

Kot smo omenili, je senzor sestavljen iz para elementov: oddajne IR led diode in sprejemnika. Na tiskanem vezju je teh elementov kar nekaj. Pri uporabi moramo biti posebno pazljivi, da elementov med posameznimi pari med seboj ne pomešamo. Nič ne pomaga, če vklopimo IR led diodo na eni strani, odboj svetlobe pa opazujemo na drugem koncu robota. Ob neustrezni rabi, ali namestitvi elementov, bo branje senzorjev, žal, neuspešno.

Na razvojni plošči so priključki posameznih elementov med seboj kar lepo premešani, zato si moramo med pisanjem programa pomagati s tabelo 1.

Da ne bi prihajalo do presluhov med senzorji, bomo opravljali merjenja na senzorjih posamično, torej vsak senzor posebej. Če bi preverjali vse senzorje istočasno, bi lahko prihajalo do medsebojnih vplivov. Tako bi dobivali nekatere meritve napačne, kar bi slabo vplivalo kvaliteto vodenja robota.



Slika 1: Razporeditev priključkov posameznih senzorjev

Pojdimo h konkretnemu primeru. Ogledali si bomo program, ki bo izvedel merjenja s prvimi štirimi senzorji. Vsi senzorji bodo preverjali le eno razdaljo. Rezultate meritev bomo shranili v spremenljivko senzor_stat. Dobljeni rezultat bomo prikazali na indikacijskih led diodah na priključkih od RB0 do RB3 vrat PORTB. Če bo pripadajoči senzor zaznal bližino stene, bo ustrezna indikacijska led dioda zasvetila, v nasprotnem primeru pa bo le-ta ostala ugasnjena.

Koraki k rešitvi naloge:

- priključke mikrokontrolerja RD0, RD7, RD5 in RD4 definiramo kot izhode; nanje so priključene IR led diode,
- priključke RB0, RB1, RB2 in RB3 definiramo kot izhode; nanje so priključene indikacijske led diode,
- priključke RA2, RA4, RA5 in RE0 definiramo kot vhode; nanje so vezani sprejemniki,
- neuporabljene priključke nastavimo kot vhode,
- pred pričetkom izvajanja programa moramo zagotoviti, da so vse led diode, tako infrardeče kot indikacijske, ugasnjene,
- pričnemo s meritvijo na prvem senzorju:

a) vklopimo tok skozi IR led diodo na priključku RD0,
b) počakamo 1ms, da se stanje na senzorju umiri.
c) Po preteku mrtvega časa preverimo izhodno stanje senzorja in postavimo ustrezno logično stanje na ustreznem mestu v spremenljivki senzor_stat.

oznaka senzorja:	priključki oddajnih IR led diod:	sprejemni vhod:	opis:
senzor 0	RD0, RD1	RA2	meri eno ali dve razdalji
senzor 1	RD7, RD6	RA4	meri eno ali dve razdalji
senzor 2	RD5	RA5	meri eno razdaljo
senzor 3	RD4	RE0	meri eno razdaljo
senzor 4	RD3	RE1	meri eno razdaljo
senzor 5	RD2	RE2	meri eno razdaljo

Tabela 1: Razporeditev priključkov za priklop digitalnih senzorjev

d) Prekinemo tok skozi IR led diodo;

- meritve na ostalih senzorjih izpeljemo na podoben način,
- ob koncu prikažemo rezultat meritve, ki ga hrani spremenljivka senzor_stat, na led diodah na vratih PORTB,
- meritve večkrat ponovimo (uporabimo neskončno zanko for(;;)).

Ker so priključki, povezani z upravljanjem senzorjev, med seboj lepo pomešani, bomo posameznim priključkom priredili imena:

```
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);
```

Rešitev problema v programskem jeziku C:

```
char senzor_stat;

// INICIALIZACIJA
void main (void)          // IME GLAVNE FUNKCIJE

{
TRISA = 0B11111111;       // PORTA definiramo kot vhod, senzorji
ADCON1 = 0x07;            // PORTA definiramo kot digitalna vhodno-
                             izhodna vrata
TRISB = 0B11110000;       // RB0-RB3 so izhodi (led diode)
TRISC = 0B11111111;       // PORTC definiramo kot vhod, ga ne
                             uporabljamo
TRISD = 0B01001110;       // RB0, RD7, RD5, RD4 so izhodi (IR led
                             diode)
TRISE = 0B00000111;       // PORTE definiramo kot 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;                // poženemo tok skozi izbrano IR led
                             diodo
wait_mili(1);             // počakamo, da se stanje na senzorju
                             stabilizira
if (SEN_0==0)             // če senzor vidi steno (aktivna 0),
{                         // se bo na njegovem izhodu pojavila
                             logična 0
senzor_stat=(1<<0)|senzor_stat;
                          // če senzor vidi, se v pomožnem registru
                          // senzor_stat na ustreznem mestu postavi
                          // logična 1;
IRL_0 = 0;                // prekinemo tok skozi 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;
}

PORTB = ~senzor_stat;     // iz registra senzor_stat se prenesejo
```

vrednost *senzor_stat*	pomen	akcija	smer vrtenja motorja 2	vrednost		smer vrtenja motorja 1	vrednost	
				mot2a	mot2a		mot1b	mot1b
00000000	preveč oddaljen od stene	popravi desno	naprej	1	0	nazaj	0	1
00000001	na primerni razdalji	vozi naravnost	naprej	1	0	naprej	1	0
00000011	preblizu stene	popravi levo	nazaj	0	1	naprej	1	0
ostale kombinacije	nedoločen	ustavi ali kaj drugega	stop	0	0	stop	0	0

Tabela 2: Princip krmiljenja DC motorjev pri vodenju robota

```
                        // negirane vrednosti na vrata PORTB,
                           kjer led diode indicirajo stanje
                           senzorjev
                        // POZOR: led dioda sveti ob aktivni
logični 0
}
}
```

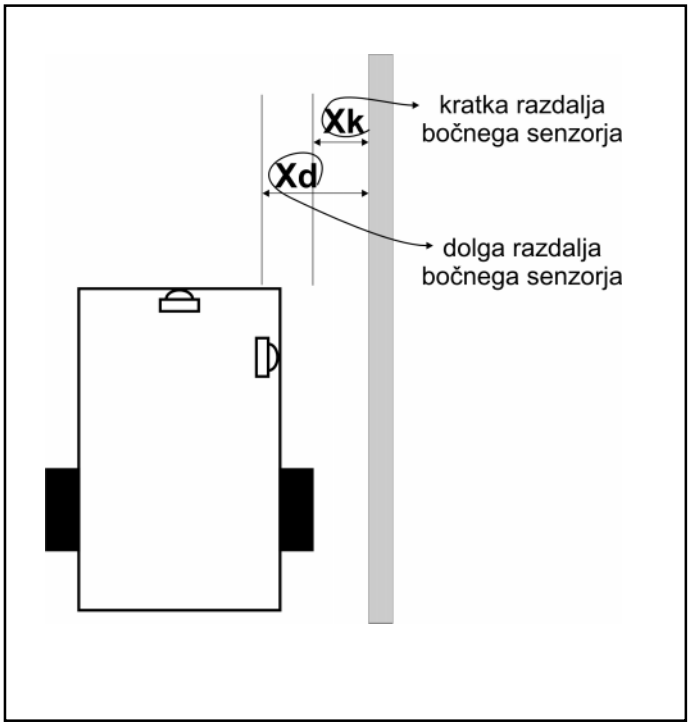
Samostojno rešite sledeči nalogi:

1. Napišite program, ki bo izvajal testiranje vseh šestih senzorjev.
2. 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.

Vodenje motorjev glede na stanje senzorjev

Meritve so opravljene, informacijo o prostoru imamo shranjeno. Ostal nam je še zadnji korak, to je premik robota v ustrezno smer. Robot se po prostoru premika s pomočjo dveh motorjev. Da se bo robot pravilno premikal, je potrebno smer vrtenja pogonskih motorjev pogosto menjati, po potrebi pa tudi spreminjati hitrost vrtenja.

Kot pripomoček pri programiranju si bomo izdelali tabelo, s katero bomo glede na izmerjene vrednosti razdalj določili smer pomikanja robota ter gibanje pogonskih koles. Poglejmo si primer vodenja robota, ki sledi ravni steni. Uporabljali bomo le en senzor, montiran na desni bočni strani, ki bo meril dve razdalji: dolgo in kratko. Rezultat shranimo v spremenljivko senzor_stat na mestih



Slika 2: Vodenje robota ob steni

bit0 in bit1. Bit0 predstavlja izmerjeno vrednost dolgega, bit1 pa vrednost kratkega stranskega senzorja. Za vsako vrednost spremenljivke senzor_stat v tabeli določimo njen pomen in akcijo, ki se bo izvedla. V naslednjem koraku določimo še smer vrtenja

pogonskih motorjev in pripadajoče izhodne vrednosti bitov mot2a, mot2b, mot1a in mot1b.

Program poskusimo še praktično izvesti:

- priključke mikrokontrolerja od RC0 do RC6 definiramo kot izhode; s tem omogočimo upravljanje motorjev. Vsi ostali priključki so vhodi,
- omogočimo delovanje motorjev s postavitvijo bitov RC1 in RC2 na logično vrednost 1,
- glede na vrednost spremenljivke senzor_stat poženemo motorje v ustrezno smer (uporabimo stavek switch-case).

Rešitev problema v programskem jeziku C:

```
bit mot2a @ BITNUM(PORTC,4),
   mot2b @ BITNUM(PORTC,5),
   mot1a @ BITNUM(PORTC,0),
   mot1b @ BITNUM(PORTC,3),
   mot1_en @ BITNUM(PORTC,2),
   mot2_en @ BITNUM(PORTC,1);

// INICIALIZACIJA
void main (void)          // IME GLAVNE FUNKCIJE

{
TRISA = 0B11111111;       // PORTA definiramo kot vhod, ga ne
                             uporabljamo
ADCON1 = 0x07;            // PORTA definiramo kot digitalna vhodno-
                             izhodna vrata
TRISB = 0B11111111;       // PORTB definiramo kot vhod, ga ne
                             uporabljamo
TRISC = 0B11000000;       // RC0 - RC6 so izhodi
TRISD = 0B11111111;       // PORTD definiramo kot vhod, ga ne
                             uporabljamo
TRISE = 0B00000111;       // PORTE definiramo kot vhod, ga ne
                             uporabljamo

mot1_en = 1;              // omogočeno delovanje gonilnika L293
mot2_en = 1;

//GLAVNI PROGRAM
switch (senzor_stat)
{
case 0B00000000: mot2a=1;
                          // motor X3 naprej
 mot2b=0;
 mot1a=0;                 // motor X4 nazaj
 mot1b=1;
 break;
case 0B00000001: mot2a=1;
                          // motor X3 naprej
 mot2b=0;
 mot1a=1;                 // motor X4 naprej
 mot1b=0;
 break;
case 0B00000011: mot2a=0;
                          // motor X3 nazaj
 mot2b=1;
 mot1a=1;                 // motor X4 naprej
 mot1b=0;
 break;
default:
 mot2a=0;                 // motor X3 stop
 mot2b=0;
 mot1a=0;                 // motor X4 stop
 mot1b=0;
 break;
}
stop:
goto stop;
}
```

Samostojno rešite sledečo nalogo:

1. Dopolnite program tako, da boste upoštevali vse možne kombinacije stanj treh senzorjev.

Napišite program za aplikacijo, ki za upravljanje robota uporablja servomotor.

Nismo še pri koncu

Robote smo torej spravili k življenju, tekmovanja Slovenske robotske lige so se že pričela. Kaj nam torej še ostane? Mogoče bi bilo dobro kaj povedati o strategiji vožnje. Kljub mnogim prizadevanjem se roboti vse pogosto zaletavajo v stene, vrtijo naokoli, obtičijo v katerem od vogalov ... O vzrokih teh nezaželenih pojavov in njihovih rešitvah bomo spregovorili prihodnjič. Pa srečno! ●

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,

[5] Infineon technologies, SFH5110, IR-Receiver for Remote Control Systems, http://www.ortodoxism.ro/datasheets/infineon/1-sfh5110.pdf

Programska oprema:

[1] Microchip, MPLAB IDE v6.61, http://ww1.microchip.com/downloads/en/DeviceDoc/mp661.zip;

[2] HI-TECH software, PICC lite COMPILER v8.05PL2, http://www.htsoft.com/products/PICClite.php;

[3] Microchip, PIC18F/PIC16F Quick Programmer, http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1824&appnote=en012031;

[4] Petr Kolomaznik, EHL elektronika, PIC downloader v1.08, http://www.ehl.cz;

[5] Herman Aartsen, TNO - The Netherlands, PIC Bootloader +, http://www.microchipc.com/