Izdelava mobilnega robota z razvojnim sistemom roboPIC (3)

Avtor: Silvan Bucik
E-pošta: silvan.bucik@tscng.net

Lep pozdrav vsem skupaj. Poletje je naokoli, spet je prišel čas za nove izzive. V poletni številki smo spoznali osnove programskega jezika C, pogledali smo si tudi preprost programček, ki prižge led diodo na izbranem priključku. Na to temo sem vam pustil tudi nekaj »domače naloge«, katere rešitve si bomo tudi danes pogledali.

Kaj nas v tokratnem prispevku čaka? Pogledali si bomo primer uporabe tipke ter prižigali led diode ob različnih pogojih. Za začetek pa bi predlagal, da si ogledamo rešitve zastavljenih nalog.

Rešitve »domačih nalog«

Naloga 1: Spremenite program tako, da se bo prižgala led dioda na priključku RB6.

```
// INICIALIZACIJA
void main (void)
{
TRISA = 0B11111111;       //PORTA = vhod
ADCON1 = 0x07;            //PORTA definiramo kot digitalna
                             vhodno-izhodna vrata
TRISB = 0B10111111;       //****     sedaj RB6 definiramo kot
                           izhod
TRISC = 0B11111111;       //PORTC = vhod, ga ne uporabljamo
TRISD = 0B11111111;       //PORTD = vhod, ga ne uporabljamo
TRISE = 0B00000111;       //PORTE = vhod, ga ne uporabljamo

// GLAVNI PROGRAM
RB6 = 0;                  //****dioda gori na priključku RB6
                           ob aktivni logični 0
stop:
goto stop;
}
```

Naloga 2: Spremenite program tako, da se bosta istočasno prižgali led diodi na priključkih RB3 in RB5.

```
// INICIALIZACIJA
void main (void)
{
TRISA = 0B11111111;       //PORTA = vhod
ADCON1 = 0x07;            //PORTA definiramo kot digitalna
                             vhodno-izhodna vrata
TRISB = 0B11010111;       // ****priključka RB3 in RB5
                             definiramo kot izhoda
TRISC = 0B11111111;       //PORTC = vhod, ga ne uporabljamo
TRISD = 0B11111111;       //PORTD = vhod, ga ne uporabljamo
TRISE = 0B00000111;       //PORTE = vhod, ga ne uporabljamo

// GLAVNI PROGRAM
RB3 = 0;                  // ****              diodi gorita
RB5 = 0;                  // ****       ob aktivni logični 0
stop:
goto stop;
}
```

Naloga 3: Prižgite vse led diode na vratih PORTB.

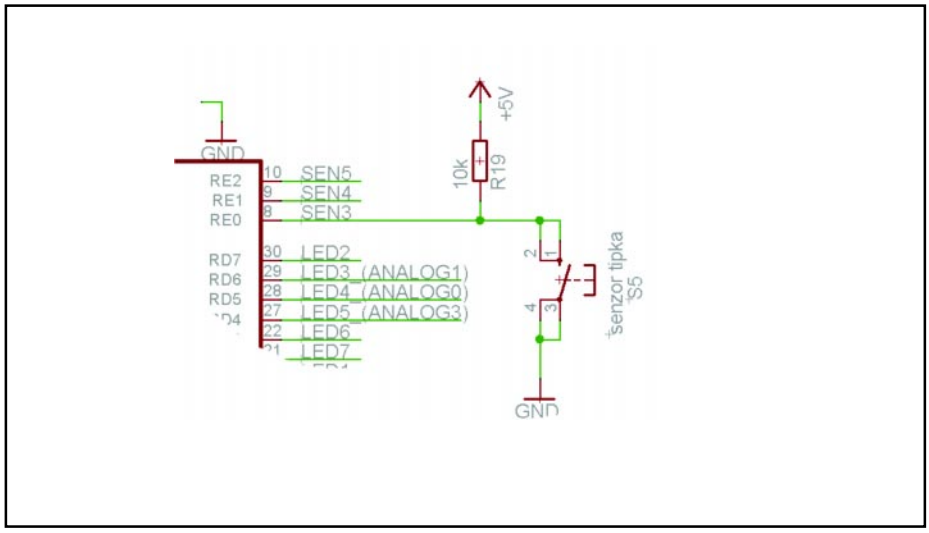
```
// INICIALIZACIJA
void main (void)
{
TRISA = 0B11111111;       //PORTA = vhod
ADCON1 = 0x07;            //PORTA definiramo kot digitalna
vhodno-izhodna vrata
TRISB = 0B00000000;       //**** vsi priključki vrat PORTB so
izhodi
TRISC = 0B11111111;       //PORTC = vhod, ga ne uporabljamo
TRISD = 0B11111111;       //PORTD = vhod, ga ne uporabljamo
TRISE = 0B00000111;       //PORTE = vhod, ga ne uporabljamo

// GLAVNI PROGRAM
PORTB = 0B00000000;       //****diode gorijo ob aktivni
                             logični 0
stop:
goto stop;
}
```

Prižig led diode ob pogoju pritisnjene tipke

Vsi priključki vrat PORTA in PORTE so namenjeni priključitvi zunanjih senzorjev (analognih in digitalnih). Za lažje preverjanje delovanja so na mestih, kjer so vezani digitalni senzorji, poleg senzorjev vzporedno vezane tudi tipke. Ob pritisku nanje simuliramo delovanje senzorja. Poglejmo si primer priključitev tipke na priključku RE0.

Na shemi smo zaradi lažjega razumevanja odstranili senzorski del. Ostalo je le tipkalo s pripadajočim pull up uporom z vrednostjo 10 kΩ. Tipka je vezana med priključkoma mase (^) in izhodno sponko SEN03 (vhod RE0). Med sponko SEN3 in napajalno linijo +5 V je dodan pull up upor, ki poskrbi za visok potencial, ko tipka ni pritisnjena. Torej, če ostane tipka nepritisnjena, bo vhodna sponka RE0 čutila preko pull up upora napetost +5 V, kar pomeni stanje logične 1. Ob pritisku na tipko pa nastane med priključkom RE0 in potencialom mase (^) kratek stik, kar povzroči na vhodu RE0 spremembo logičnega stanja iz logične 1 v logično 0.

Nekaterim se mogoče pojavlja vprašanje, čemu je potreben pull up upor, če pa bi vse skupaj delovalo tudi brez njega. Pull up upor zagotavlja visok potencial (+5 V), v trenutkih, ko tipka ni pritisnjena. V tem primeru bi ga res ne potrebovali. Kaj bi pa se



Slika 1: Princip vezave tipke na priključek mikrokontrolerja

zgodilo ob pritisku na tipko? Pri direktni vezavi na potencial +5 V bi ob pritisnjeni tipki skoznjo stekel kratkostični tok in uničil tipkalo. V vezavi s pull up uporom pa se to ne zgodi, saj le-ta preprečuje, da bi skozi tipkalo stekel velik tok.
Sedaj nas čaka naloga. Napisali bomo program, ki bo prižgal led diodo na izhodu RB3, a le v primeru pritiska tipke RE0. V nasprotnem primeru naj ostane dioda neprižgana. Vezavo led diod na vratih PORTB že poznamo, tako da je na tem mestu ne bomo ponovno opisovali.

Kako rešiti problem?

Vse zunanje priključke mikrokontrolerja definiramo kot vhode (tudi priključek RE0), razen priključka RB3, ki ga določimo kot izhod.
Če je tipka pritisnjena (na vhodu RE0 je logično stanje 0), prižgemo led diodo z vpisom logične 0 na pripadajoči izhod RB3.
Če tipka ni pritisnjena (na vhodu RE0 je logično stanje 1), led diodo ugasnemo z vpisom logične 1.
Vse skupaj moramo večkrat ponoviti, saj ni nujno, da bomo pritisnili na tipko takoj po zagonu programa, ampak bomo verjetno to storili nekaj trenutkov kasneje.

Rešitev problema v programskem jeziku C

```
// 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 = 0B11110111;       // RB3 določimo kot izhod, vsi ostali priključki so
                             vhodi
TRISC = 0B11111111;       // PORTC definiramo kot vhod, ga ne uporabljamo
TRISD = 0B11111111;       // PORTD definiramo kot vhod, ga ne uporabljamo
TRISE = 0B00000111;       // PORTE definiramo kot vhod, tudi RA0
RB3 = 1;                  // dioda na RB3 je v začetku ugasnjena

// GLAVNI PROGRAM

do
{
if (RE0 == 0)             // ob pritisku tipke je logično stanje enako 0
{
RB3 = 0;                  // če je tipka pritisnjena, led dioda sveti
}
else                      // ko tipka ni pritisnjena, je logično stanje enako 1
```

```
{
RB3 = 1;                    // če tipka ni pritisnjena, led
                               dioda ugasne
}
}
while (1>0);                // pogledamo še enkrat, če ni sedaj
                               kaj drugače
}
```

V programu ste opazili dve sintaksi programskega jezika C, ki ju do sedaj še nismo omenili. In sicer sta to stavek if-else ter zanka do-while. Predlagam, da se na tem mestu malo zaustavimo in ogledamo način njune uporabe.

Stavek if-else

Stavek if-else uporabljamo pri vejitvi toka programa. Uporabe stavka se poslužujemo v primerih, ko se nadaljevanje programa lahko izvaja na dva različna načina. Pot, po kateri se bo program nadaljeval, je odvisna od pogoja - vrednosti ene ali več spremenljivk.

```
Sintaksa:
if (pogoj)
{                           // če je pogoj izpolnjen se izvede
                               akcija _1
akcija_1;
                            }
else
{                           // v nasprotnem se izvede akcija _2
akcija_2;
                            }
```

V našem programu smo preverjali, ali je tipka pritisnjena. Če je bila tipka pritisnjena, se je izvedel del programa, ki je prižgal led diodo. V nasprotnem primeru pa se je pognal del programa, ki je led diodo ugasnil.

Programska zanka do-while

Zanka do-while označuje del programa, ki se izvaja, dokler je pogoj za izvajanje zanke izpolnjen.

```
Sintaksa:
do
   {
   akcija_1;                // vedno se izvede programski del
                               akcija_1
   }                        //
while (pogoj)               // če je pogoj na koncu zanke
                               izpolnjen
```

V našem primeru smo zanko do-while uporabili za izvajanje neskončne zanke. Ob zaključku zanke se preverja pogoj, ali je 1>0. To seveda vedno drži, torej je pogoj vedno izpolnjen, zato se zanka neštetokrat izvede. To je seveda tudi smiselno, saj je preverjanje stanja tipke potrebno pogosto preverjati. Čas, kdaj bo uporabnik pritisnil tipko, ni določen, zato mora program »bedeti nad dogajanjem«, da bo lahko v primernem trenutku prižgal led diodo.

Programski skok goto

V rešitvah nalog iz prejšnje številke ste zagotovo opazili tudi stavek goto. Stavek goto povzroči takojšnji skok na označeno mesto v programu. Programerji, ki so vajeni programiranja v zbirnem jeziku, znajo povedati, da se brez tega ukaza v zbirniku ne da kaj prida narediti. Običajni zbirniški jeziki poznajo le logične odločitve (v jeziku C ustrezajo stavku if-else) in programske skoke, ki jih v C-ju poznamo kot skoke goto. Uporabe programskih zank kot so while, do-while, in for (ostale zanke bomo spoznali kasneje) ti jeziki ne podpirajo. Čeprav se programski skok goto v jeziku C manj uporablja, pa nam vseeno pogostokrat pride prav.

```
Sintaksa:
goto oznaka;                // program se nadaljuje na oznaki
                               oznaka
.                           // ta
.                           // del
.                           // se
.                           // ne
.                           // izvaja
.
oznaka:                     // oznaka ima podpičjenadaljeva
                            nje;
                            // tu se nadaljuje izvajanje
                            programa
```

Primer zaustavitve toka programa

V naših programčkih smo stavek goto uporabili za zaustavitev toka izvajanja programa, kar je izgledalo takole:

```
stop:                       // program skače na samega sebe
goto stop;                  // tako zaustavimo izvajanje
                               programa
```

Sedaj pa »domača naloga«

Samostojno rešite sledeči nalogi:

1. Spremenite program tako, da bo led dioda na priključku RB3 svetila, ko bo tipka RA0 nepritisnjena. Ob pritisku nanjo pa naj led dioda ugasne.
2. Spremenite program tako, da bo ob pritisku tipke RA4 gorela led dioda na izhodu RB5. Ob spustu tipke pa naj zasveti led dioda na izhodu RB3.

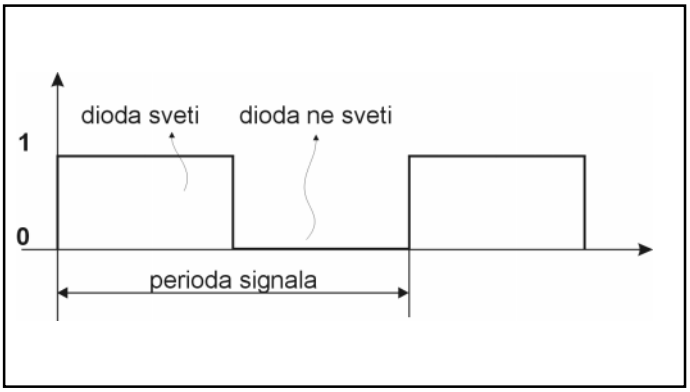
Utripanje diode

Sedaj pa napišimo program, ki bo povzročil utripanje led diode na priključku RB0. Frekvenca utripanja naj bo približno 5 Hz.

Utripanje predstavlja zaporedje prižiganja in ugašanja led diode. Dioda najprej zasveti in nekaj časa gori, nato ugasne in v tudi tem stanju ostane nekaj časa. Opisano zaporedje se ves čas ponavlja. Sedaj pa nas zanima, kolikšen je potrebni čas zakasnitve. Ta je odvisen od izbire periode oziroma frekvence utripanja. Dioda sveti natančno polovico periode, enako dolgo ostane tudi ugasnjena. Čas zakasnitve izračunamo po slednji enačbi:

$$čas\_zakasnitve = \frac{perioda}{2} = \frac{1}{2 \cdot frekvenca}$$ Enačba 1)

V našem primeru, pri frekvenci utripanja 5 Hz, dobimo čas zakasnitve enak 100 ms.



Slika 2: Princip delovanja utripajoče led diode

Sedaj pa pojdimo k reševanju zastavljene naloge

- Vse zunanje priključke mikrokontrolerja definiramo kot vhode, razen priključka RB0, ki je izbran kot izhod.
- Prižgemo led diodo na priključku RB0.
- Počakamo 100 ms, kar predstavlja polovico periode (uporabimo knjižnično funkcijo wait_mili (zakasnitev_v_ms)).
- Ugasnemo led diodo.
- Še enkrat počakamo 100 ms - polovico periode.
- Zadnje štiri akcije nato ponavljamo, led dioda prične utripati.

Rešitev problema v programskem jeziku C

```
// INICIALIZACIJA
int frq=frekvenca_kHz;
void main (void)          // IME GLAVNE FUNKCIJE
{
TRISA = 0B11111111;       // PORTA definiramo kot vhod
ADCON1 = 0x07;            // PORTA definiramo kot digitalna
                             vhodno-izhodna vrata
TRISB = 0B11111110;       // RB0 določimo kot izhod, vsi
                             ostali priključki so vhodi
TRISC = 0B11111111;       // PORTC definiramo kot vhod
TRISD = 0B11111111;       // PORTD definiramo kot vhod
TRISE = 0B00000111        // PORTE definiramo kot vhod

// GLAVNI PROGRAM
while (1>0)
     {
     RB0 = 0;             //na izhodu RB0 se pojavi napetost
                            0V (logična 0) - dioda

// se prižge
     wait_mili(100);      // počakamo pol periode
     RB0 = 1;             //na izhodu RB0 se pojavi napetost
                            +5V (logična 1) - dioda ugasne
     wait_mili(100);      // še enkrat počakamo pol periode
     }
                          // vse skupaj ponavljamo (while
                             zanka), da prične led dioda
                             utripati
}
```

V programu opazimo dve novosti: zanko while in knjižnično funkcijo wait_mili. Poglejmo si podrobnosti njune uporabe.

Programska zanka while

Del programa, ki ga zajema programska zanka while, se izvaja tako dolgo, dokler je izpolnjen pogoj za izvajanje zanke. Preverjanje se izvede v začetku zanke, nato se v primeru izpolnitve pogoja del programa akcija_1 izvede. Bistvena razlika med zankama while in do-while je v načinu preverjanja pogojev izvajanja. Pri zanki while se preverjanje izvede v začetku - pred izvedbo zanke. Pri zanki do-while pa se preverjanje izvede ob zaključku izvajanja zanke. Iz tega je razvidno, da se zanka do-while zagotovo izvede vsaj enkrat, ne glede na to, ali je pogoj izpolnjen.

```
Sintaksa:
while (pogoj)             // če je pogoj izpolnjen,
     {
     akcija_1;            // se izvede programski del
akcija_1
     }                    // v nasprotnem sledi izstop iz
                             zanke
```

Zanko while smo tudi v sedanjem primeru uporabili za izvajanje neskončne zanke (preverja se pogoj, ali je 1>0). Smiselnost uporabe neskončne zanke je v zahtevi, da mora dioda ves čas utripati. Utripanje je posledica stalnega prižiganja in ugašanja led diode, zato mora program ves čas teči in skrbeti za pravočasno prižiganje in ugašanje led diode.

Knjižnična datoteka delay.c

Knjižnična datoteka delay.c, je datoteka »domače izdelave«, ki nam omogoča enostavnejše programiranje. V datoteki se nahajata dve funkciji, s pomočjo katerih v programu enostavno izvedemo potrebne zakasnitve. To sta funkciji:

- wait_mili(argument)

in

- wait_sec(argument).

S funkcijo wait_mili realiziramo zakasnitve do 255 ms, za daljše zakasnitve uporabimo funkcijo wait_sec. Vrednost argumenta določa čas zakasnitve, in sicer čas v milisekundah za prvo funkcijo oziroma čas v sekundah v drugem primeru. Vrednost argumenta pri obeh funkcijah lahko spreminjamo v območju od 0 do 255, kar omogoča realizacijo zakasnitev od 0 do 255 ms (funkcija wait_mili()) oziroma od 0 do 255 s z uporabo funkcije wait_sec(().

Uporaba

- wait_mili(čas_milisekund);
- wait_sec(čas_sekund);

Samostojno rešite še sledeči nalogi

1. Spremenite program tako, da bo led dioda utripala hitreje (na primer 10 Hz), ali počasneje (na primer 2Hz).
2. Napišite program, da bodo utripale vse led diode na vratih PORTB.

Nagradni nalogi

Podjetje ELBACOMP d.o.o. (http://www.elbacomp.si), podjetje za prodajo in distribucijo elektronskih komponent, bo podelilo vsakomur, ki bo pravilno rešil zastavljeni nalogi, lepo praktično nagrado. Nagrada je mikrokontroler PIC16F877A. Bralce naj spomnimo, da je omenjeni mikrokontroler tudi sestavni del razvojnega sistema roboPIC.

1. Napišite program, ki bo omogočil prižig led diod na priključkih RB2 in RB7 le ob sočasnem pritisku tipk RA2 in RE3. V kateremkoli ostalem primeru diodi ne smeta svetiti.
2. Napišite program, ki bo povzročil utripanje led diode na priključku RB1 le ob pritisku tipke RA5.

Do naslednjega snidenja

Prihodnjič bi se malo posvetili podrobnostim programskega jezika C. Zopet malo teorije ... Pa še to. Vsa dokumentacija in programska oprema v zvezi z razvojnim sistemom roboPIC je na voljo na spletnih straneh revije Svet elektronike. Pa lep pozdrav! ●

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,

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/