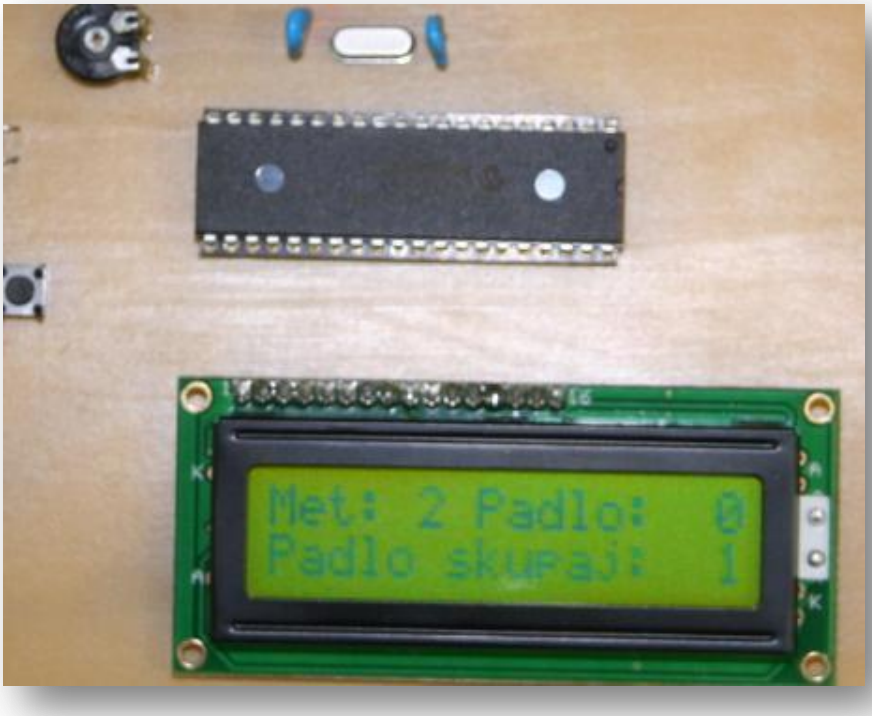
»Mladi za napredek Maribora 2013«

30. srečanje

Krmiljenje bowling centra

Raziskovalno področje: Elektrotehnika, elektronika

Raziskovalna naloga



Avtor: TADEJ GOLOB

Mentor: MILAN IVIĻ

Ġola: SREDNJA ELEKTRO-RAĻ UNALNIĠKA ĠOLA MARIBOR

Maribor, januar 2013

1. Kazalo vsebine

Kazalo slik

2. POVZETEK

Raziskovalna naloga predstavlja delo na področju krmiljenja bowling steze.

Rečemo lahko, da se je bowling rodil takrat, ko je človek prvič zakotalil okrogel kamen, da bi podrl postavljene palice. Po nekaterih podatkih so pri izkopavanjih v Egiptu našli zametke bowlinga, ki izvirajo okoli leta 5200 pred našim štetjem. Po pravilih, takšnih ki veljajo še danes, pa se igra bowling od leta 1895, ko so v New Yorku ustanovili ameriško bowling zvezo ABC.

Na začetku, so igralci bowlinga število podrtih palic in kasneje kegljev preštevali. Z razvojem elektronike in avtomatizacije, pa se število podrtih kegljev in število metov izračunava in prikazuje avtomatsko. Toda kako je izdelana avtomatika, ki poskrbi za avtomatski prikaz potrebnih podatkov? Z željo po raziskovanju, sem se odločil izdelati model bowling steze, na katerem bo omogočen avtomatski prikaz števila metov, število podrtih kegljev posameznega meta in skupno število podrtih kegljev po več metih.

Za nalogo je uporabljen mikrokontroler, ki s pomočjo ustreznih senzorjev pridobiva potrebne podatke, ki so potrebni za prikaz želenih podatkov.

3. ZAHVALA

Za nasvet, pomoč in potrpljenje se zahvaljujem mojemu mentorju. Nekatera dela so mi olajšali tudi sošolci, ki so me spodbujali in se jim posebej zahvaljujem. Zahvaljujem pa se tudi staršem, ki so mi pomagali pri izdelavi makete.

4. VSEBINSKI DEL

4.1 Uvod

Namen raziskovalne naloge je izdelati model bowling steze s prikazom števila metov, števila podrtih kegljev posameznega meta in skupno število podrtih kegljev. Cilj je izdelati krmilje z mikrokontrolerjem, uporabiti ustrezne senzorje za zaznavo števila metov in števila podrtih kegljev in na LCD[1] prikazovalniku prikazati podatke.

Pri izbiri mikrokontrolerja je vplivalo število priključkov, saj jih za svoje delovanje kar nekaj potrebuje LCD prikazovalnik, po tudi število kegljev ni malo. Ko je bil izbran mikrokontroler, je sledila izbira senzorjev. Ker je nad kegljami montirana razsvetljava, je bila logična uporaba fotouporov. Pod vsakim kegljem je nameščen fotoupor, ki se osvetli, če kegelj podremo. Na podlagi števila osvetljenih fotouporov, bo možno ugotoviti njihovo število. Potrebno je bilo določiti čas, po katerem se še ugotavlja število podrtih kegljev.

Ko je bil problem štetja podrtih kegljev rešen, je nastalo vprašanje, kako seštevati število podrtih kegljev po več metih. Po proučevanju programiranja mikrokontrolerja, je nastala rešitev v izdelavi takega programa, ki bo sproti sešteval podrte keglje po metih in to število prikazoval na LCD prikazovalniku, dokler s tipko ne ponastavimo začetnega prikaza.

Zaradi enostavnosti in cene, je tudi za število metov uporabljen fotoupor, ki ga preko proge osvetljuje LED dioda. Pri vsakem metu krogla prekine vir svetlobe do fotoupora. Število teh prekinitev je hkrati tudi število metov.

4.2 Postopek razvijanja naloge

Ko sem prišel do idejne rešitve, sem se moral odločiti, kako stvar realizirati in izdelati model, na katerem bo vidno praktično delovanje naloge.

Najprej sem se začel ukvarjati s fotoupori LDR, kako jih uporabiti v nalogi? Mikrokontroler mora dobiti na vhodnih priključkih spremembo logičnega stanja, če se fotoupori osvetlijo.

[1] LCD (angl.: Liquid Crystal Display), prikazovalnik na tekoče kristale.

Zato je bilo potrebno spremembo napetosti na fotouporu preoblikovati v spremembo logičnega nivoja.

Nadalje je bilo potrebno osvojiti osnovno znanje o mikrokontrolerjih PIC. To znanje sem pridobival na krožku v šoli in ga pri uporabi v moji nalogi še dopolnjeval. Prav tako je bilo potrebno znanje iz področja programiranja mikrokontrolerja. Program je moral biti izdelan tako, da bo zadovoljeval vse moje zahteve, ki sem si jih zadal pri nalogi. Posebno poglavje mi je predstavljalo prikazovanje želenih podatkov na LCD prikazovalniku, saj na začetku sploh nisem imel predstave, kako bodo ti podatki prikazani.

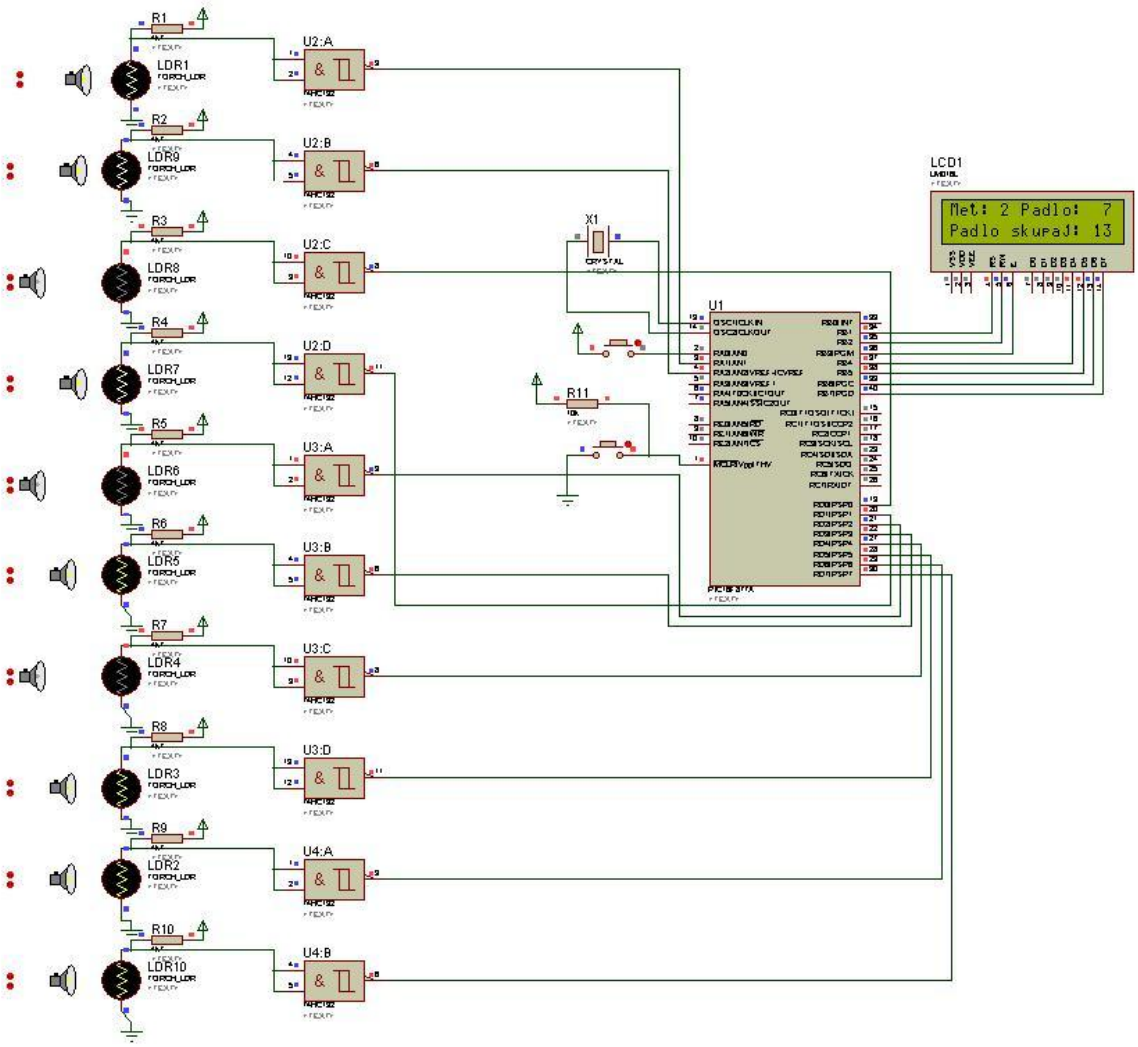
Ko sem prišel tako daleč, da sem delovanje programa lahko simuliral v programskem okolju, je sledila izdelava elektronskega vezja in tiskanine, ki je bila potrebna za prikaz delovanja na izdelanem modelu.

4.3 Opis izdelanega modela

Model je sestavljen iz bowling steze s pripadajočim številom kegljev. Pod vsakim kegljem je nameščen fotoupor, prostor s kegljami pa je osvetljen.

Za preizkušanje delovanja naloge sem najprej sestavil elektronsko vezje na eksperimentalno ploščico. Zaradi lažjega preizkušanja delovanja sem namesto fotouporov, ki so nameščeni pod keglji, uporabil tipke. Pritisk na določeno tipko je predstavljal podrti kegelj oziroma osvetlitev fotoupora pod njim. Tudi število metov sem simuliral s pritiski na tipko. Po vsaki spremembi programa za mikrokontroler, ki sem ga s pomočjo programatorja PICkit 2 vpisoval v mikrokontroler, sem lahko takoj videl odziv na delovanju vezja. Takšen način dela mi je zelo koristil pri razvoju programa ter pri iskanju posameznih rešitev za nastale probleme, ki jih ni bilo malo.

Za nastavitev napetosti na fotouporih v odvisnosti od svetlobe sem nameraval uporabiti trimer potenciometre, zaporedno vezane s fotoupori. Zamisel sem opustil zaradi nameravane montaže dodatne razsvetljave nad keglji. Delovanje modela z bowling stezo je namreč močno odvisno od zunanje razsvetljave, kjer se model nahaja.



Slika 1: Prikaz delovanja naloge (vir: avtor naloge)

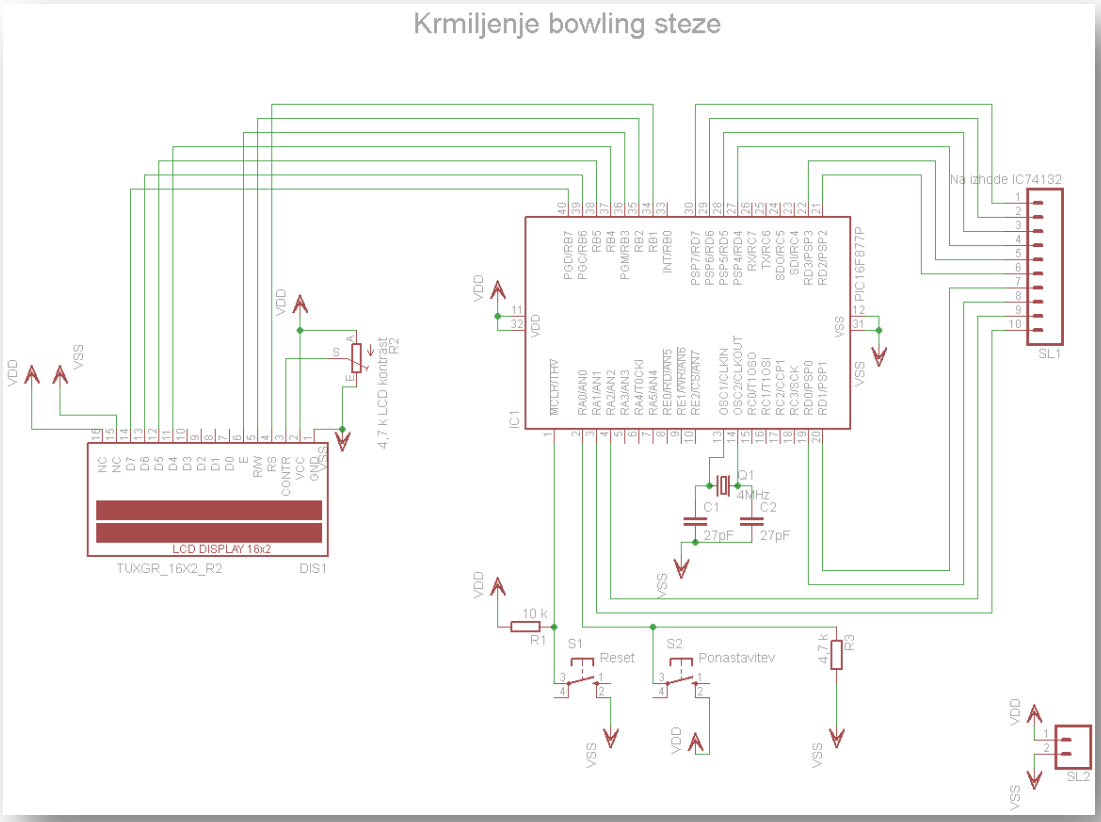
Ploščica s fotoupori in integriranimi vezji IC74132 je nameščena pod stezo, ploščica z mikrokontrolerjem in LCD prikazovalnikom pa ob stezo. Za ugotavljanje števila metov je ob stezi nameščena še visoko svetleča LED[2] dioda, preko nje na drugi strani steze pa fotoupor LDR[3]. Krogla pri svojem potovanju do kegljev prekine svetlobne žarke do fotoupora. Za trenutek se napetost na fotouporu poveča na vrednost, ki spremeni logično stanje na priključku mikrokontrolerja. Na podlagi tega dobimo podatek o številu meta.

[2] LED (angl.: Light Emitting Diode), svetleča dioda.
[3] LDR (Light Dependent Resistor), svetlobno odvisen upor.

5. Tehnična in tehnološka dokumentacija

4.4.1 Elektronski načrt

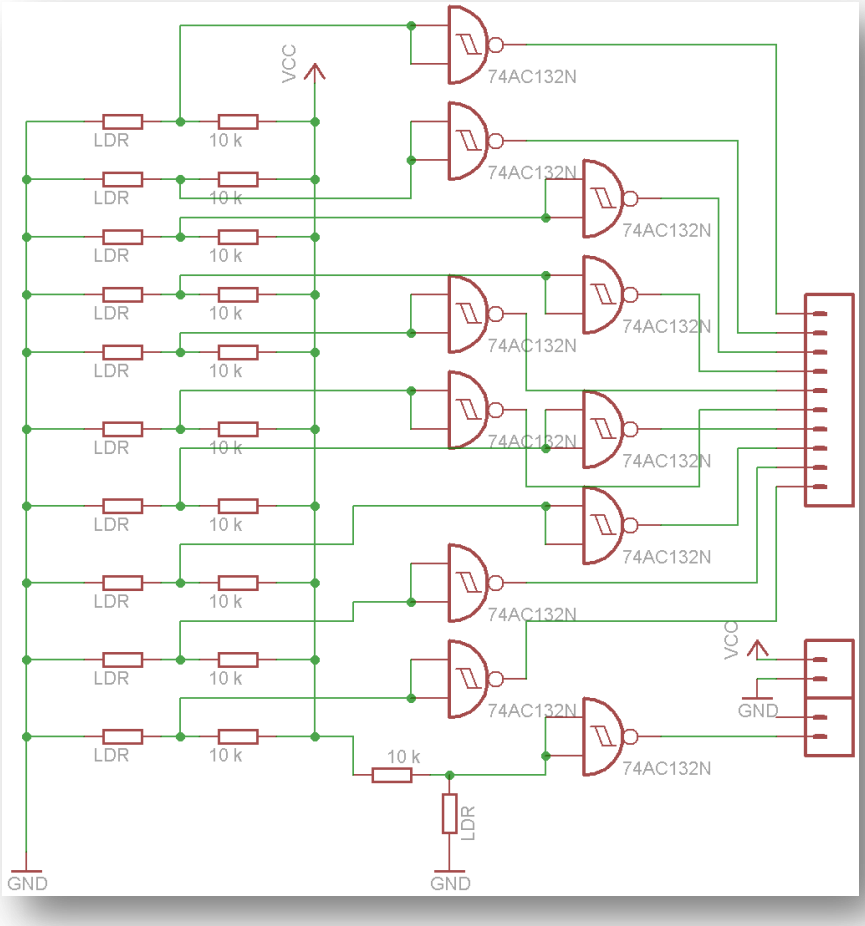
Elektronski načrt vezja, izdelan v programskem okolju Eagle prikazuje Slika 2. Fotoupori so na vhode mikrokontrolerja priključeni preko NIN logičnih vrat s Schmittovim prožilnikom. Na izhodu teh logičnih vrat dobimo vedno napetostni nivo logične 0 ali logične 1, odvisno od velikosti vhodnega signala.



Slika 2: Elektronski načrt vezja z mikrokontrolerjem (vir: avtor naloge)

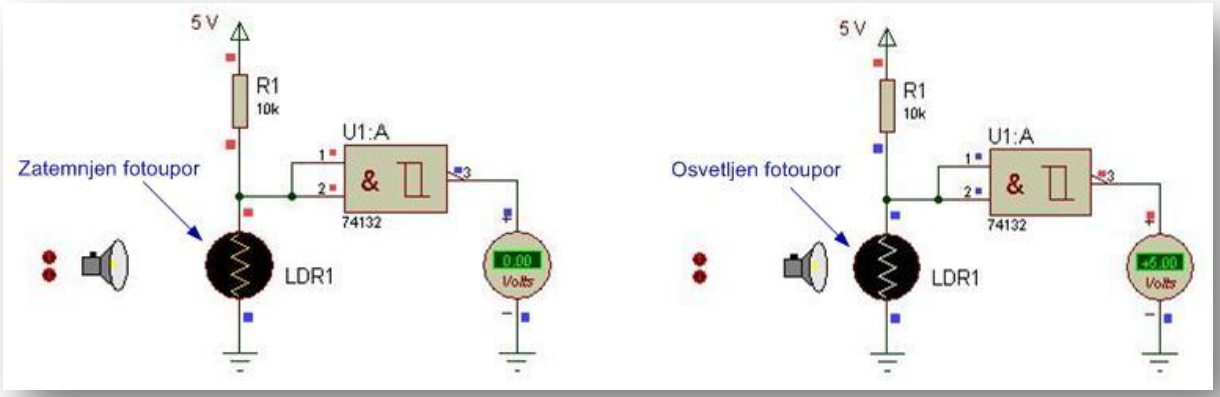
Na PORTB mikrokontrolerja je priključen dvovrstični LCD prikazovalnik z lastno osvetlitvijo. Kontrast napisa nastavljamo s trimer potenciometrom 4,7 kΩ, katerega drsnik je povezan na tretji priključek LCD prikazovalnika. Na priključka za zunanji oscilator je mikrokontrolerju priključen kristalni oscilator 4 MHz, za odpravljanje motenj pa služita dva kondenzatorja vrednosti 27 pF. Mikrokontrolerju sta priključeni še dve tipki, ena za

resetiranje, druga pa za ponastavitev napisa na LCD prikazovalniku s pripadajočima pull-up oziroma pull-down uporoma. Vezje je priključeno na napajalno napetost Vdd = 5 V.



Slika 3: Elektronsko vezje s fotoupori in integriranimi vezji IC74132 (vir: avtor naloge)

Elektronsko vezje s fotoupori in integriranimi vezji IC74132 je prikazano na sliki 3. Na podlagi praktičnega preizkušanja so upori, ki so fotouporom zaporedno vezani, vrednosti 10 kΩ. Za doseganje potrebnih logičnih nivojev, ki jih potrebujejo vhodni priključki mikrokontrolerja so fotoupori povezani na vhode NAND logičnih vrat. Ko je fotoupor osvetljen, je na izhodu logičnih vrat napetost 5 V (logična 1), če pa fotoupor zatemnimo, je na izhodu logičnih vrat napetost 0 V (logična 0). Izhodi NAND logičnih vrat so povezani na vhodne priključke mikrokontrolerja.

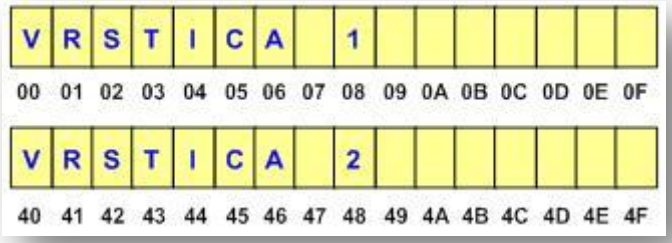


Slika 4: Fotoupor in NAND logična vrata (vir: avtor naloge)

4.4.2 LCD prikazovalnik

Mikrokontroler skrbi za prikaz potrebnih podatkov na LCD prikazovalniku. LCD prikazovalniki omogočajo izpisovanje črk, številk in ostalih znakov. Uporabljamo jih kot vmesnike med uporabnikom in elektronsko napravo. V nalogi je uporabljen dvovrstični LCD prikazovalnik z lastno osvetlitvijo, ki ima v vsaki vrstici po 16 znakov (DEM 16216 SYH-PY). Vsebuje lasten kontroler (KS0070B-00), ki skrbi za vklapljanje ustreznih pik na zaslonu. Vsak znak je sestavljen iz 5 x 8 pik. Vsak znak na LCD prikazovalniku ima svoj naslov v DDRAM[4] pomnilniku, kot prikazuje slika 5. Začetni naslov skrajnega levega znaka v prvi vrstici je 0x00, skrajnega levega znaka v drugi vrstici pa 0x40. Vsaka vrstica je navidezno dolga po 40 znakov, kar znaša skupaj 80 znakov. Od vseh 40 znakov v vsaki vrstici jih trenutno vidimo na LCD prikazovalniku le 16.

[4] DDRAM (angl.: Display Data Random Access Memory), prikaz podatkov bralno-pisalnega pomnilnika.



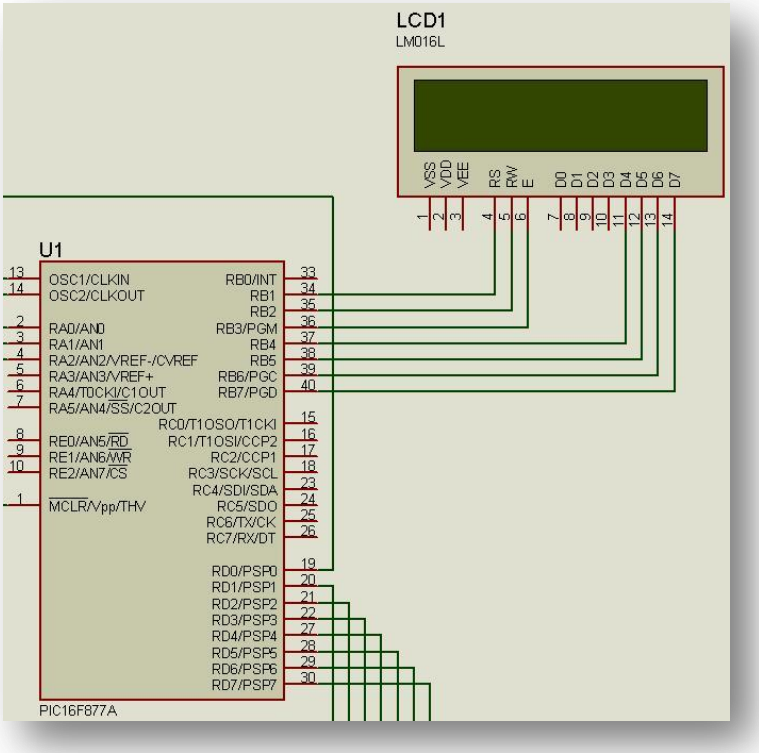
Slika 5: Dvovrstični LCD prikazovalnik z naslovi (vir: avtor naloge)

LCD prikazovalniki brez možnosti osvetlitve zaslona imajo 14 priključkov, taki z možnostjo osvetlitve zaslona pa imajo 2 priključka več, torej 16 priključkov. Funkcija posameznega priključka je standardna, vedeti moramo le, kje se nahaja prvi priključek. Če priključki za določen tip LCD-ja niso označeni, poiščemo podatke na spletu. Priključki od 7 do 14 LCD prikazovalnika so namenjeni podatkovnim linijam (od D0 do D7). Preko njih se prenašajo podatki od mikrokontrolerja do LCD-ja, če pišemo v LCD, in od LCD-ja do mikrokontrolerja, če beremo iz LCD-ja. LCD zasloni lahko delujejo v 8 ali v 4-bitnem načinu. V nalogi je izbran 4-bitni način delovanja, saj s tem privarčujemo pri priključkih mikrokontrolerja. Podatki se pri 4-bitnem načinu delovanja prenašajo po 4 bite hkrati, zato moramo poslati obe polovici bajta posebej, pošiljamo pa jih po podatkovnih linijah od D4 do D7. Tretji priključek LCD-ja služi nastavitvi kontrasta. Kontrast reguliramo z napetostjo na tem priključku. V ta namen uporabimo potenciometer kot delilnik napetosti.

LCD prikazovalnik krmilimo tako, da mu pošiljamo ukaze ali podatke. Če postavimo priključek 4 (RS) v stanje logične 0, bo LCD sprejel podatek kot ukaz, če pa ga postavimo na nivo logične 1, bo LCD sprejel navaden podatek. Priključek 6 (E) je namenjen vklopu prikazovalnikove logike. Ko pošiljamo neki podatek na LCD, ga moramo nekako obvestiti, da je na podatkovnih linijah nov podatek. To storimo tako, da ta priključek vklopimo (postavimo na 1) in izklopimo (postavimo na 0). Ob tem prehodu bo LCD sprejel nov podatek. Pri 4-bitnem načinu delovanja pošljemo najprej zgornje 4 bite podatka na priključke 11–14 (D4–D7), vmes vklopimo in izklopimo priključek 6 (E) in nato na iste priključke pošljemo še spodnje 4 bite podatka. Zatem zopet vklopimo in izklopimo priključek 6 (E). Vsak vpis ukaza ali podatka vzame LCD-ju nekaj časa, da ga obdela. V tem času, podajajo ga proizvajalci

LCD-jev, mu ne smemo pošiljati naslednjega ukaza ali podatka, zato uporabimo metodo zakasnitve, ki bo v našem primeru ca. 5 ms. Časa zakasnitve ni treba točno nastaviti, le manjši od predpisanega ne sme biti.

LCD prikazovalnik moramo pred uporabo inicializirati. To pomeni, da moramo izvesti določeno zaporedje operacij, da ga postavimo v želeni način delovanja.



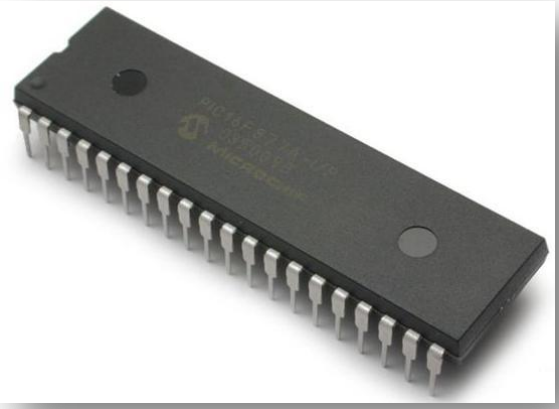
Slika 6: Priključitev LCD prikazovalnika na mikrokontroler (vir: avtor naloge)

4.4.3 Mikrokontroler PIC16F877a

Osrednji del vezja je mikrokontroler PIC16F877a [1]. Njegove glavne značilnosti so:

- Je Microchipov mikrokontroler v PDIP ohišju z 40 priključki.
- Priključki so razporejeni na PORTA, PORTB, PORTC, PORTD in PORTE in jih lahko programsko krmilimo.
- Napetost napajanja Vdd je 5 V.

- Vsebuje 8 k flash programskega pomnilnika, 368 bajtov RAM pomnilnika in 256 bajtov podatkovnega EEPROM pomnilnika.
- Centralno procesna enota je 8 bitna.
- Vsebuje tri timerje in pozna 15 vrst prekinitev.
- Pozna 35 instrukcij za programiranje v zbirnem jeziku.
- CPU izvaja instrukcije, vgrajen ima 8-bitni delovni register. Vanj se shranjujejo podatki po izvedbi posameznih instrukcij.
- Programski pomnilnik je sestavljen iz 8192 lokacij. V vsako lokacijo lahko vpišemo 14-bitni podatek. Instrukcija zavzame eno lokacijo. Posebno vlogo imata prva in peta lokacija. V prvi se nahaja ukaz, ki se bo izvedel takoj po vklopu mikrokontrolerja. V peti se začne del programa, ki se izvede ob prekinitvi.
- Maksimalna frekvenca zunanjega oscilatorja je 20 MHz.



Slika 7: Mikrokontroler PIC16F628a [2]

Podatkovni RAM[5] pomnilnik je razdeljen na štiri banke: banko 0, banko 1, banko2 in banko3. Vsebuje SFR[6] in GPR[7] registre. SFR registri zavzemajo prvih 32 lokacij v vsaki banki. Preko njih komuniciramo z ostalimi enotami mikrokontrolerja. GPR registri zavzemajo 368 bajtov RAM pomnilnika. Če želimo brati ali pisati v register podatkovnega pomnilnika RAM, se

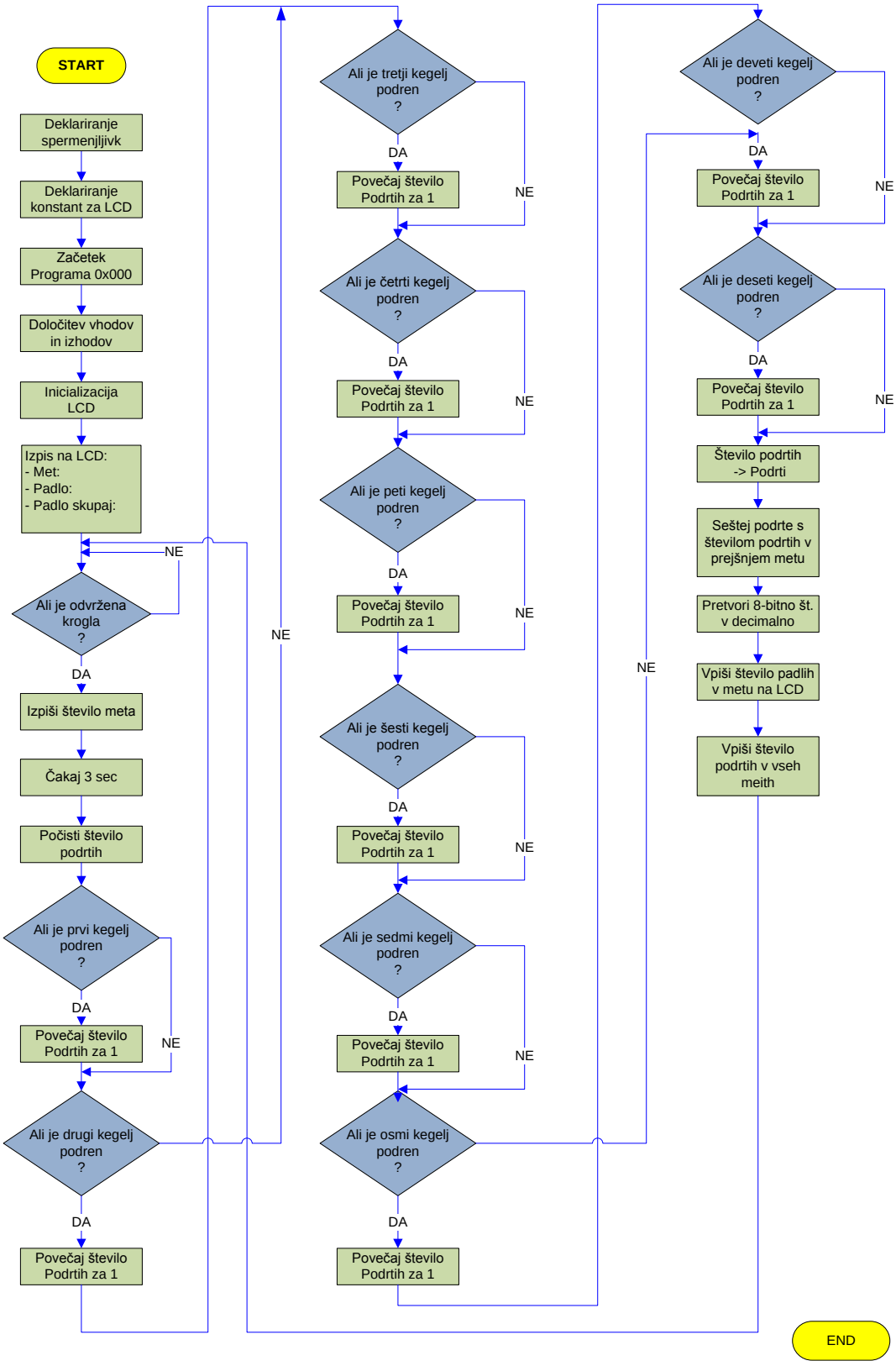
[5] RAM (angl.: Random Access Memory), bralno-pisalni pomnilnik.
[6] SFR (angl.: Specisl Function Registers), posebni funkcijski registri.
[7] GPR (angl.: General Purpose Registers), registri namenjeni splošni uporabi.

moramo nahajati v banki, v kateri je želeni register. Vsebine GPR registrov, ki so določeni s strani proizvajalca, se izgubijo, ko mikrokontrolerju izklopimo napajanje.

Za izvajanje vseh instrukcij v mikrokontrolerju PIC skrbi 8-bitna aritmetična logična enota ALU[8]. Lahko sešteva in odšteva 8-bitna števila ter z njimi izvaja nekatere logične operacije (AND, OR, XOR, NOT …). Po izvršitvi vsake instrukcije se v registru STATUS postavijo trije biti na ustrezne vrednosti. Iz njihovih vrednosti lahko ugotovimo, kakšen je dobljeni rezultat. Ta je lahko pozitiven, negativen, enak 0 ali napačen. Te tri bite imenujemo zastavice.

[8] ALU (angl.: Arithmetic Logic Unit), aritmetična logična enota.

4.4.4 Diagram poteka



4.4.5 Program za krmiljenje naprave

Program je napisan v zbirnem jeziku v okolju MPLAB IDE v8.80. V RAM podatkovnem pomnilniku so od naslova 0x20 naprej, kjer je prostor za GPR registre, definirane konstante in spremenljivke, potrebne za posamezna zakasnitve, za LCD zaslon, za napise, ki so zapisani v tabelah in za prikaz številk.

S konfiguracijskimi biti je izbran zunanji oscilator, stabilizacija, izključitev časovnika stražnega mehanizma in izključitev zaščite pred kopiranjem. Konfiguracijske bite določimo na začetku programa, takoj za direktivo #include, ta vključi v program datoteko p16f877a.inc, ki vsebuje imena registrov v mikrokontrolerju:

```
        list        p=16f877a           ; Tip mikrokontrolerja
        #include    <p16f877a.inc>      ; Vključi v program datoteko p16f877a.inc,
                                        ; ki vsebuje imena registrov v mikrokontrolerju.

        __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _LVP_OFF & _XT_OSC
```

Po vklopu mikrokontrolerja in po ponastavitvi program vedno začne na naslovu 0x000. Na naslovu 0x004, to je peta lokacija programskega pomnilnika, se začne prekinitvena rutina ali prekinitveni program. Da se po vklopu ali ponastavitvi mikrokontrolerja ne začne izvajati prekinitveni program, smo pred direktivo **org** 0x004 z instrukcijo **goto** povzročili skok v programu na oznako oziroma labelo, ki ji sledi v parametru. Instrukcija **goto** s parametrom Glavni_zacetek torej pošlje izvajanje programa mimo prekinitvene rutine na naslov Glavni_zacetek, kjer se nadaljuje izvajanje programa:

```
        org         0x000
        goto        Glavni_zacetek
        org         0x004

Glavni_zacetek
```

Mikrokontrolerju moramo določiti vhodne in izhodne pine zaradi vezja, s katerim deluje. Za krmiljenje vezja po elektronskem načrtu so pini RA0 - RA3 in RD0 - RD7 določeni kot vhodi, pini RB0 - RB7 pa kot izhodi. Na vhodne pine so priključeni fotoupori in tipka za ponastavitev prikaza na LCD zaslonu, na izhodne pa LCD zaslon. Vhodne pine določimo s postavitvijo ustreznih bitov registra TRISA oziroma TRISD na 1, izhodne pa s postavitvijo ustreznih bitov registra TRISB na 0:

```
        bsf     STATUS,RP0          ;Banka 1
        movlw   0x06
        movwf   ADCON1              ;Pini na PORTA so digitalni
        movlw   b'00001111'
        movwf   TRISA               ;Pini RA0, RA1, RA2 in RA4 so vhodni, drugi so izhodni
        movlw   b'11111111'
        movwf   TRISD               ;Vsi pini na PORTD so vhodni
        clrf    TRISB               ;Vsi pini na PORTB so izhod
```

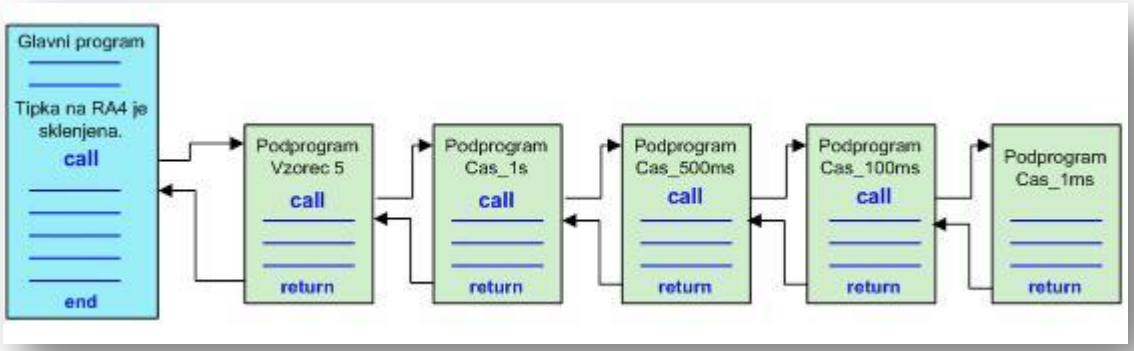
Kot smo že omenili, moramo LCD prikazovalnik pred uporabo inicializirati. To pomeni, da moramo izvesti določeno zaporedje operacij, da ga postavimo v želeni način delovanja:

```
        ;****************** Inicializacija LCD ************************

LCD_Init
        bcf     RW
        bcf     RS              ;RW in RS liniji na 0
        movlw   .15
        call    CakajMs         ;cca 15ms pavze za LCD
        movlw   0x30
        movwf   PORTB           ;Na D7 - D4 pošljemo 0x30
        bsf     E
        bcf     E               ;Opozorimo LCD na nov podatek
        movlw   .5
        call    CakajMs         ;cca 5ms pavze
        movlw   0x30
        movwf   PORTB           ;Na D7 - D4 pošljemo 0x30
        bsf     E
        bcf     E               ;Opozorimo Lcd na nov podatek
        call    Cakaj119u       ;119 mili sekund pavze
        movlw   0x30
        movwf   PORTB           ;Na D7 - D4 pošljemo 0x30
        bsf     E
        bcf     E               ;Opozorimo Lcd na nov podatek
        movlw   .5
        call    CakajMs
        movlw   0x20
        movwf   PORTB           ;Na D7 - D4 pošljemo 0x20
        bsf     E
        bcf     E
        call    Cakaj119u
                                ;Sedaj smo v 4-bitnem načinu
        movlw   0x28            ;2x16 znakov, 5x8 znaki, 4-bitno
        call    LCD_pisi        ;Pošljemo 0x28 na Lcd
        movlw   0x08            ;Izklop zaslona
        call    LCD_pisi        ;Pošljemo 0x08 na Lcd
        movlw   0x01            ;Brisanje DDRAM-a
```

```
        call    LCD_pisi        ;Pošljemo 0x01, izbrišemo zaslon
        movlw   .2
        call    CakajMs         ;Zahtevana zakasnitev, večja od 2ms
        movlw   0x06            ;Pisanje v desno, brez pomikanja zaslona
        call    LCD_pisi        ;Pošljemo 0x06 na Lcd
        movlw   0x0C            ;Vklop zaslona, velik kurzor
        call    LCD_pisi        ;Pošljemo 0x0F na Lcd
        return                  ;Konec podprograma
```

Kot vidimo že iz podprograma LCD_init, lahko iz podprograma kličemo drugi podprogram, ampak največ v globino osem.



Slika 8: Primer klicanja podprogramov v globino 5 (vir: avtor naloge)

Po inicializaciji LCD-ja pokličemo podprogram LDC_Niz1 za izpis napisa "Met:", ki se izpiše na začetku prve vrstice:

```
;****** Podprogram za izpis niza iz tabele1
LCD_Niz1
        clrf    Naslov          ;Naslov = 0
        bsf     RS              ;Podatkovni način
        movlw   .4
        movwf   Stevec2         ;Niz ima 4 znake (Met:)
Delaj1
        movf    Naslov,w
        movwf   Naslov
        call    Tabela1
        call    LCD_pisi        ;Podatek -> LCD
        incf    Naslov,f        ;Naslednji podatek
        decfsz  Stevec2,f       ;Ali smo končali?
        goto    Delaj1          ;Ne še
```

```
        bcf     RS
        return                  ;Da, izhod iz podprograma
```

Iz tega podprograma kličemo podprogram Tabela1, v kateri je napisan niz, ki se izpiše na LCD prikazovalnik:

```
;--------- Naši nizi, napisani so v tabelah ----------

Tabela1 addwf   PCL,f
        DT              "Met:"          ;Tabela1 z nizom
```

V spremenljivko Stevec2 vpišemo vrednost 4, ker ima niz "Met:" 4 znake. Programski pomnilnik mikrokontrolerja pogosto uporabljamo za shranjevanje podatkov, napisanih v obliki tabele. Ta ima obliko podprograma. Preden tabelo pokličemo z instrukcijo call, moramo v delovni register vpisati zaporedno številko (odmik) podatka, ki ga želimo dobiti iz tabele. Prvi element tabele ima odmik 0. Ko se tabela izvrši, dobimo v delovnem registru vrednost izbranega podatka v tabeli. Tak način pridobivanja podatkov iz tabele nam omogoči direktiva DT (**d**efine **t**able – definiraj tabelo).

Podobno kot napis "Met:" izpišemo še napisa "Padlo:" in "Padlo skupaj:", le da smo za drugi napis nastavili naslov DDRAM-a na 0x07, za tretji napis pa na 0x40. Drugi napis se začne na osmem znaku prve vrstice prikazovalnika (prvi znak ima naslov 0x00), tretji napis pa na prvem znaku druge vrstice prikazovalnika.

Slika 9: Napisi na LCD prikazovalniku (vir: avtor naloge)

Ko so vsi trije napisi izpisani, program začne preverjati, ali je bila krogla odvržena:

```
;***** Preberemo stanje tipke: RA0 registrira št. metov krogle ***********
;***** Zapišemo število meta ***********
```

```
Beri
        btfss   PORTA,0         ;Ali je bila krogla odvržena? (Jo je senzor zaznal?)
        goto    Beri            ;Ne

        movlw   0x04            ;Da, piši število metov v prvo vrstico od petega znaka naprej
        call    DDRAM_n         ;na petem znaku je desetica, na šestem enica (meti od 1 do 99)
        movf    Stevec3,w       ;Stevec3 premaknemo v Podatek1
        movwf   Podatek1        ;V register Podatek1 vpiši število meta
        call    Pisi_st1        ;Izpišemo podatek (št. meta) na LCD
        incf    Stevec3,f       ;Stevec3 povečamo za 1
```

Število metov izpisujemo v prvo vrstico od petega znaka naprej, na petem znaku so desetice, na šestem pa enice. Največje število metov je lahko 99. Ko senzor zazna met krogle, počakamo tri sekunde, da padejo podrti keglji. Pod vsakim podrtim kegljem se je osvetlil fotoupor. Število osvetljenih fotouporov je hkrati število podrtih kegljev. Pred vsakim metom inicializiramo ustrezno spremenljivko, da začnemo šteti podrte keglje od začetka:

```
;********** Preberemo stanje podrtih kegljev **********
Preverjanje_podrtih
        clrf    Podrti          ;Inicializacija (po vsakem metu začne šteti podrte keglje od začetka, od 0)
Preveri_prvi_kegelj
        btfss   PORTD,0
        goto    Preveri_drugi_kegelj
        incf    Podrti,f        ;Register Podrti povečuje za 1 za vsak podrti kegelj
;********** Preberemo stanje podrtih kegljev **********        goto    Preveri_tretji_kegelj
        incf    Podrti,f        ;Register Podrti povečuje za 1 za vsak podrti kegelj
Preveri_tretji_kegelj
```

Na enak način preverimo stanje vseh desetih kegljev. Število podrtih kegljev shranjujemo v spremenljivki Podrti. Po končanem preverjanju podrtih kegljev v posameznem metu, seštejemo podrte keglje po metih in podatek shranimo v spremenljivko Vsota_podrtih:

```
        movf    Podrti,w
        addwf   Vsota_podrtih   ;Seštevaj podrte keglje po metih, podatek shrani v Vsota_podrtih
```

Število podrtih kegljev posameznega meta izpišemo na naslova DDRAM-a 0x0D (od 14-tega znaka naprej v prvi vrstici prikazovalnika), skupno število podrtih pa na naslov 0x4D (od 14-ti znaka naprej v drugi vrstici prikazovalnika).

Slika 10: Izpis števila metov, podrtih v metu in vseh podrtih kegljev (vir: avtor naloge)

Za izpis števila na LCD prikazovalnik ga moramo najprej pretvoriti v desetiško število. Postopek pretvorbe opravimo z deljenjem s pomočjo odštevanja. Najprej določimo stotice tako, da številu odštevamo 10010 toliko časa, dokler ne dobimo negativnega rezultata in štejemo potrebne korake. Ti koraki so enaki stoticam. Dobljeni negativni vrednosti pri zadnjem odštevanju prištejemo nazaj 100. Ostanek nato na enak način delimo z 10, da dobimo desetice. Ostanek po izračun desetic so kar enice. V programu opravlja pretvorbo podprogram Pisi_st. Pred klicem tega podprograma moramo v spremenljivko Podatek1 vpisati 8-biten podatek, ki ga želimo izpisati na LCD prikazovalnik, to pa so prvič podrti keglji po posameznem metu in drugič skupno število podrtih kegljev. Ker imajo dobljene številke vrednosti od 0 do 9, jih moramo še pretvoriti v ASCII[9] kodo ustreznih znakov. V standardni kodni tabeli ASCII imajo številke od 0 do 9 zaporedno naraščajoče kode, zato opravimo pretvorbo enostavno tako, da vrednosti številke prištejemo ASCII kodo znaka 0. Podprograma za izpis števila na LCD prikazovalnik in za 8-bitno pretvorbo:

```
;************* Podprogram za izpis števila na LCD **************

Pisi_st
        bsf     Nicla,0
        movlw   .100
        call    Pretvori8               ;Pretvori stotice
        movlw   .10
        call    Pretvori8               ;Pretvori desetice
        movf    Podatek1,w              ;Enice v delovni register
        addlw   '0'                     ;Pretvori v ascii kodo
        call    LCD_pisi
        return
```

[9] ASCII (angl.: American Standard Code for Information Interchanga), ameriški standardni nabor za izmenjavo informacij.

```
;************** Podprogram za pretvorbo 8-bitnega števila v desetiško *********

Pretvori8                                   ;Pretvori 8-bitno število v desetiško
        clrf    Cifra                       ;Inicializacija
Nadaljuj
        subwf   Podatek1,f      ;Vrednost registra W odšteje od vrednosti Podatek1, shrani nazaj v W
        btfsc   STATUS,C                    ;Ali je rezultat pozitiven?
        goto    Ena_vec                     ;Da
        addwf   Podatek1,f                  ;Ne, naredi rezultat pozitiven
        movf    Cifra,w
        addlw   '0'                         ;Izračunaj ASCII kodo
        btfsc   Nicla,0                     ;Je to vodilna ničla?
        movlw   ' '                         ;Da, izpiši presledek LCD
        call    LCD_pisi                    ;Ne, izpiši cifro na LCD
        return
Ena_vec
        bcf     Nicla,0
        incf    Cifra,f
        goto    Nadaljuj
```

Spremenljivka Nicla je uporabljena zato, da za vrednost številke na primer 1 ne dobimo prikaza 001 pri stoticah oziroma 01 pri deseticah. Namesto ničel pred 1 se izpiše presledek.

6. DRUŽBENA ODGOVORNOST

Pri nalogi krmiljenja bowling steze sem se zavedal pomena avtomatizacije, ki naj bi jo elektrotehniki in računalnikarji razvijali za namen čim bolj prijaznega okolja za uporabnike, tako rekreativce kot profesionalce. Ti lahko pri svojem udejstvovanju, rekreaciji in tekmovanju, zaupajo v tehnične službe oziroma naravoslovno znanost, ki skrbi za tehnološki napredek na vseh področjih človekovega udejstvovanja.

7. SKLEP

Z uspešnim zaključkom moje raziskovalne naloge sem zelo zadovoljen in ponosen. Veliko sem se naučil tako na področju elektrotehnike, kot na področju programiranja ter razumevanja mikrokontrolerjev.

Pri izdelavi naloge se mi je zdel največji izziv programiranje, ko pa sem izdelal maketo, kamor sem moral povezati in priključiti vse elemente in sklope naloge, so se problemi kar vrstili. Če odmislim težave pri izdelavi ogrodja makete, kjer sem moral uporabiti ročne

spretnosti, sem se največ naučil pri reševanju problemov, kot so bili razni zunanji vplivi na delovanje sistema. Enkrat mi naloga ni delovala pravilno zaradi premočne zunanje svetlobe, spet drugič zaradi slabe zunanje svetlobe. Tudi pokritost fotouporov s keglji je zelo pomembna za pravilno delovanje.

Izkušnje ki sem jih pridobil pri tej raziskovalni nalogi bom uporabil pri nadaljnjem raziskovanju, saj že imam v mislih več nalog, ki jih želim izdelati.

8. VIRI

1. Programiranje PIC [spletni vir]. Dostopno na URL: https://sites.google.com/site/programiranjepic/home (16. 10. 2012)
2. [1] Dostopno na URL: http://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf (16. 12. 2012)
3. [2] Dostopno na URL: http://airqualitymonitoring.blogspot.com/2012/10/pic16f877a-microcontroller.html (26. 01. 2013)