Univerza v Ljubljani
Fakulteta za elektrotehniko

DIGITALNA URA

Seminar
Dokumentacija

Avtor: Jernej Vrščaj
Vpisna št.: 64070459
Ljubljana, september 2012

1. KAZALO

2. KLJUČNE BESEDE **/ KEY WORDS**

- čas **/ time**
- digitalna ura **/ digital clock**
- budilka **/ alarm clock**
- **microchip**
- **pic16F870**
- kvarčni kristal **/ quartz crystal**
- piskač **/ buzzer**
- led dioda **/ led diod**
- 7-segmentni led prikazovalnik **/ 7-segment led display**
- multipleksiranje **/ multiplexing**
- regulacija osvetlitve prikazovalnika **/ light intensity display regulation**
- led senzor **/ led sensor**
- tipke **/ push-buttons, keys**
- vezje ob izpadu omrežja **/ back-up circuit**

3. UVOD

Čas je naš spremljevalec že od začetka obstoja. Vedno bolj smo nekako obremenjeni z njim, saj razvoj človeka hitro napreduje, potrebe po hitrostnih standardih se večajo, hitrejši procesorji, transport... Razumljivo, kajti ne živimo večno, no saj zaenkrat še ne. Nekaj tako primarnega kot je čas, sem iz filozofskega vidika skušal zajeti v tej nalogi. Gledano iz praktičnega, pa ker sem potreboval uro v svoji sobi. Digitalna ura, kot samostojni izdelek v tej projektni nalogi, je kompaktna, atraktivna, funkcionalna, precejšni del pa sem posvetil njeni natančnosti, kar je tudi potrebna lastnost vsake ure. Ker je to prvi resnejši projekt, sem moral dobro pretuhtati katere komponente pridejo v upoštev, kateri mikrokrmilnik ter ostala periferija, izbrati primerno ohišje, saj je potrebno izdelek realizirati tako, da bi nekako lahko sodil na prodajalne police. Na svetovnem trgu je ta produkt zelo populariziran, saj se ga da dobiti v vseh oblikah, velikostih, z manj ali več funkcionalnostmi, za zelo ugodno ceno. Moja ura se najbrž cenovno ne bo približala slednjim, bom pa na koncu podal ekonomski vidik, se pravi, finančni ter tudi časovni pregled, ki ga je ta projekt zahteval.

4. SPECIFIKACIJE NAPRAVE

- Prikaz trenutnega časa na štirih 7-segmentnih led prikazovalnikih.
- Nastavljiv čas preko dveh tipk, ure (H) **,** minute (M).
- »Reset« ure preko tipke (R), ura začne teči od 0:00
- Opcija nastavitve alarma, preko tipke (A).
- Led indikator označuje, da je alarm vklopljen.
- Utripanje dvopičja (:) vsako sekundo med urami in minutami nakazuje sekunde.
- Samodejno spreminjanje osvetlitve prikazovalnika glede na dnevno svetlobo.
- Primarno napajanje je omrežna napetost, adaptirana na 9V DC.
- Sekundarno napajanje (ob izpadu omrežja) preko 9V baterije.

5. ČASOVNI IN FINANČNI PLAN

5.1 ČASOVNI PLAN

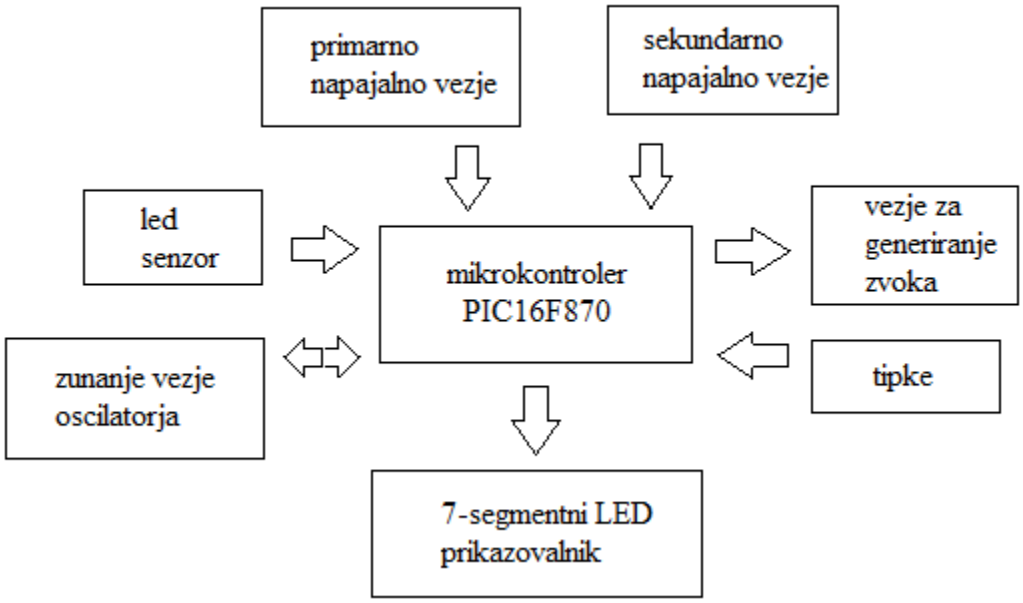
Glede na to, da je to moj prvi resnejši projekt z mikrokrmilniki, si bom vzel toliko časa, kot bo potrebno. Seveda je realnost druga zgodba, kjer se lovi zastavljene roke. Delček časa mi bo vzela sama ideja in načrtovanje izdelka. Ocenjujem na dan ali dva. Kar se tiče »softwerskega« dela, se ne bom omejeval, saj se katera ideja »rodi« tudi čez noč. Računam nekje 2 do 4 tedne. Komponente se kupuje sproti, po potrebi. Čas za izdelavo tiskanine in ohišja (tega bom dal delati), se lahko zakasni tja do 1 meseca.
Predpostavim torej časovni okvir 2 mesecev.

5.2 FINANČNI PLAN

komponente (~ 20 €)
+ ohišje (~ 30 €)
~ 50 €

6. ZASNOVA NAPRAVE

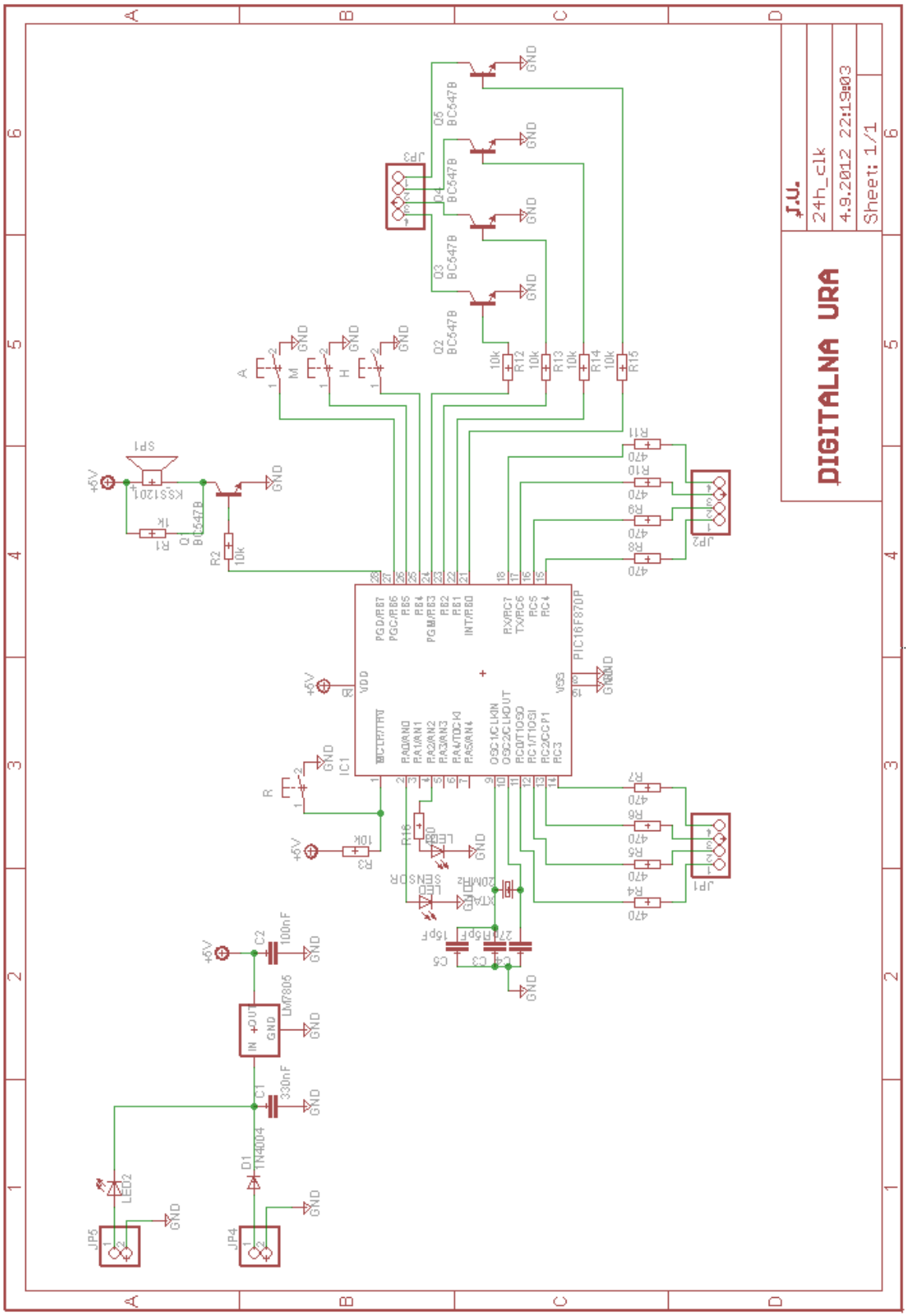
6.1 BLOČNA SHEMA

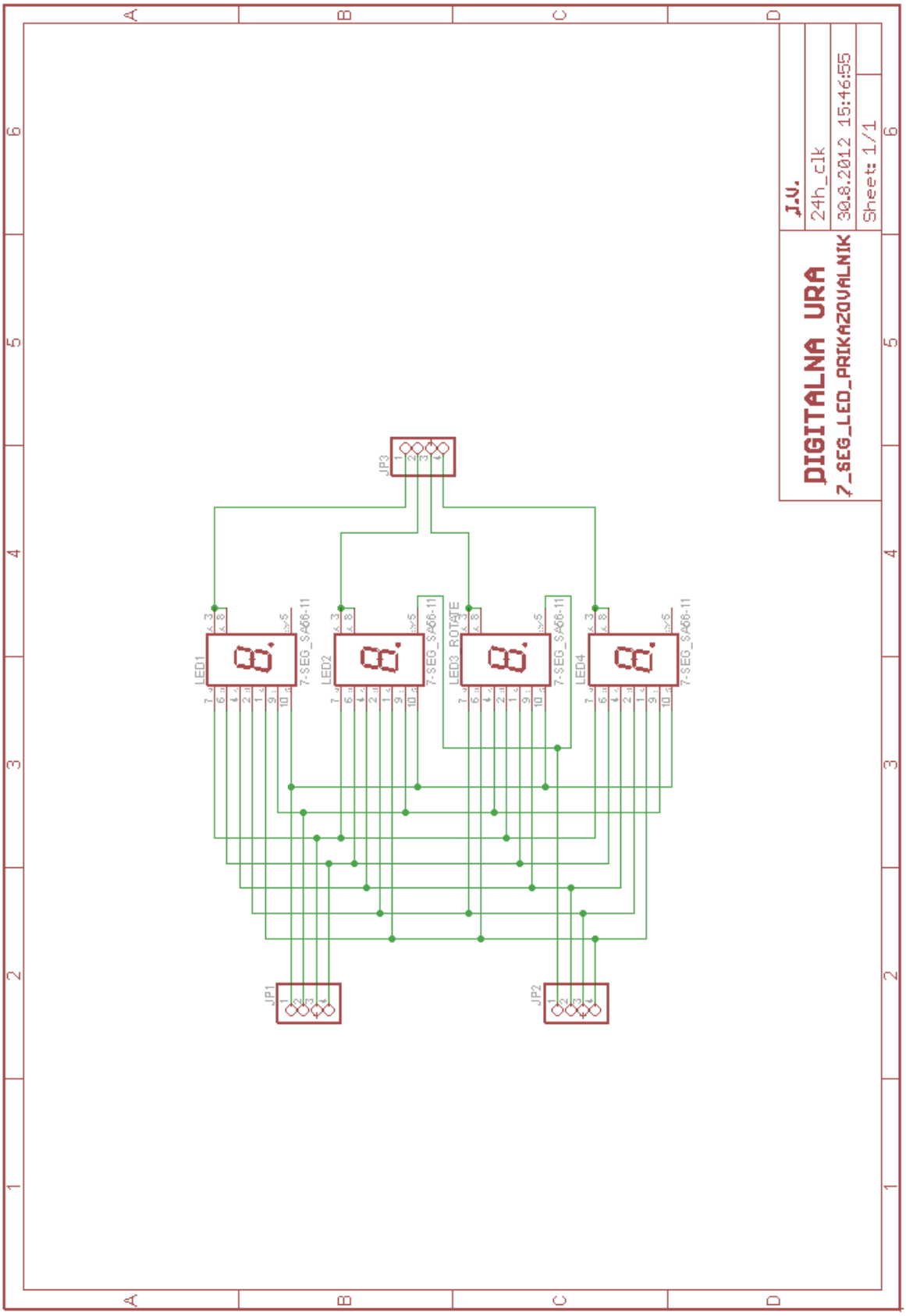


Slika 1: Blokovna shema naprave

Jedro naprave je, kot je iz sheme razvidno, microchipov PIC16F870. Ta mikrokrmilnik je bil izbran, glede na zahteve projekta. Štirje 7-segmentni led prikazovalniki služijo za prikaz časa. S pomočjo tipk (H) in (M) nastavljamo željeni čas. Vsak pritisk poveča trenutno uro ali minuto, glede na to, katera tipka je pritisnjena. S tipko (A) lahko nastavimo alarm ali ga izklopimo. Ura ima tudi tipko »reset«, s katero poženemo program od začetka, izpis na prikazovalniku pa je 0:00
Ob ujemanju trenutnega časa z nastavljenim časom alarma, se sproži piskač, ki odda zvok približno na vsako sekundo. Za napajanje mikrokrmilnika poskrbita dve vezij. Primarno napajanje je konstantno, iz omrežja prilagojeno na 9V enosmerne napetosti. Ob izgubi omrežja, se zgodi preklop na 9V baterijo. Eno ali drugo aktivno vezje, gre še skozi vsesplošni 5V napetostni regulator LM7805, ki zregulira končno napetost na 5V, ki jo mikrokrmilnik potrebuje.
Led diodo, lahko uporabimo kot senzor svetlobe, ob pravilni vezavi na mikrokrmilnik, ki preko adc-ja pridobi bitno vrednost svetlobe v prostoru.
Načeloma vezje oscilatorja spada pod sklop mikrokontrolerja, vendar sem ga v blokovni shemi izpostavil, saj je eden od glavnih faktorjev, ki vpliva na točnost ure.
Bolj podroben opis podsklopov, sledi v nadaljevanju.

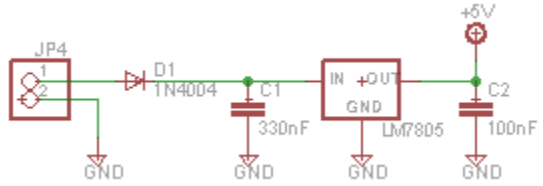
6.2 ELEKTRIČNA SHEMA




DIGITALNA URA
7_SEG_LED_PRIKAZOVALNIK
J.V.
24h_clk
30.8.2012 15:46:55
Sheet: 1/1
JP3
LED1
LED2
LED3
ROTATE
LED4
7-SEG_SA66-11
JP1
JP2

6.2.1 OPIS DELOVANJA POSAMEZNIH PODSKLOPOV

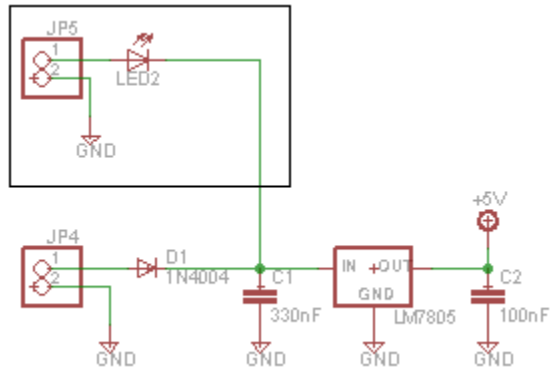
6.2.1.1 PRIMARNO NAPAJANJE



Slika 2: Primarno napajalno vezje

Na vhodu dobimo, preko adapterja, 9V enosmerne napetosti iz omrežja. D1 je varovalno dioda, v primeru zamenjave vhodnih priključkov (+ in -). Napetostni padec čez diodo je, ob pravilni priključitvi, ~0.7V. Ta napetost gre skozi napetostni regulator LM7805, ki da na izhodu končno napetost 5V. C1 in C2 sta gladilna kondenzatorja.

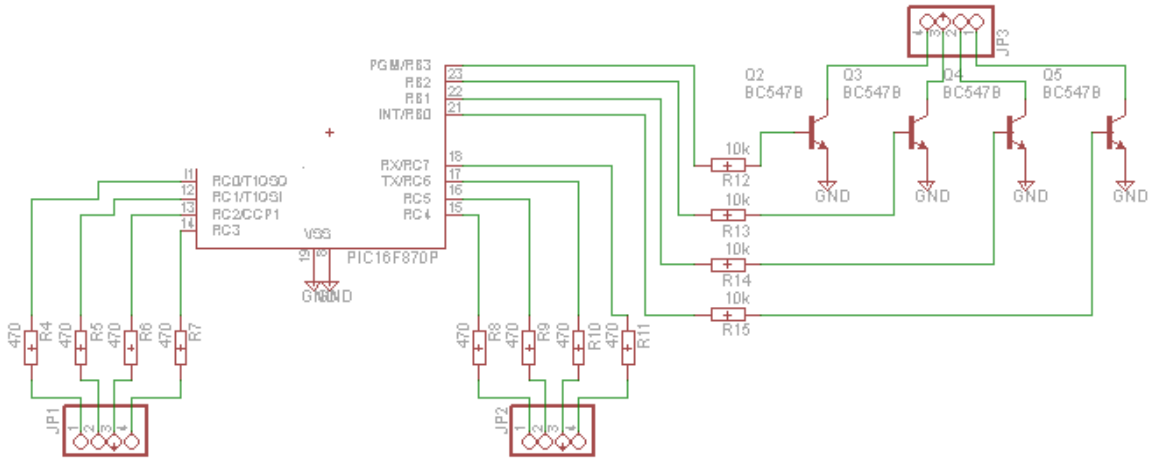
6.2.1.2 SEKUNDARNO NAPAJANJE



Slika 3: Sekundarno napajalno vezje

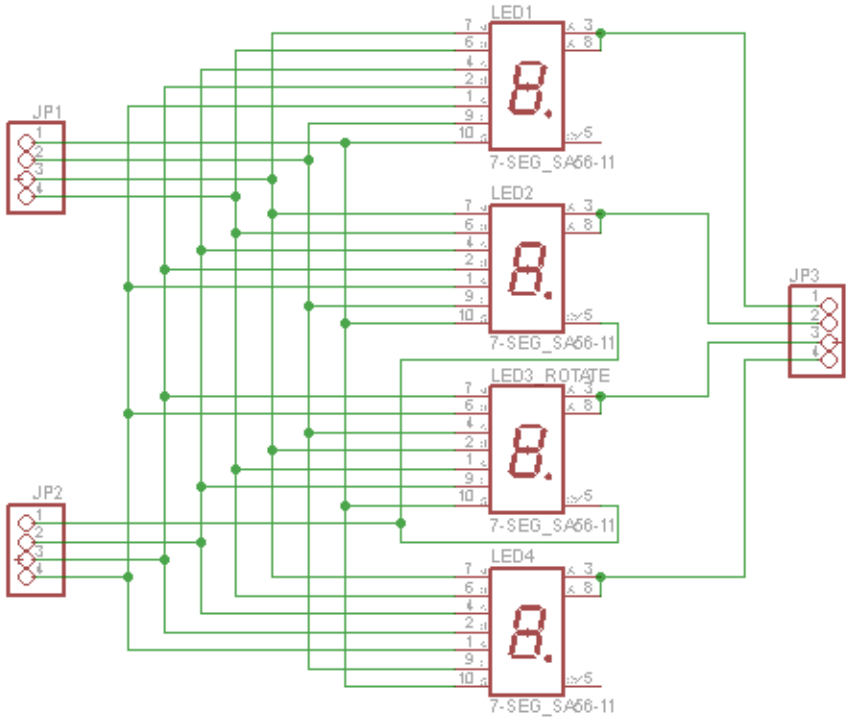
V primeru izgube omrežja, se dioda D1 zapre, LED2 pa postane prevodna. LED2 je tudi indikator napajanja preko 9V baterije, za kar pa žrtvujemo padec 2V, ko LED2 prevaja. Na vhod regulatorja tako dobimo napetost ~7V.

6.2.1.3 7-SEGMENTNI LED PRIKAZOVALNIK



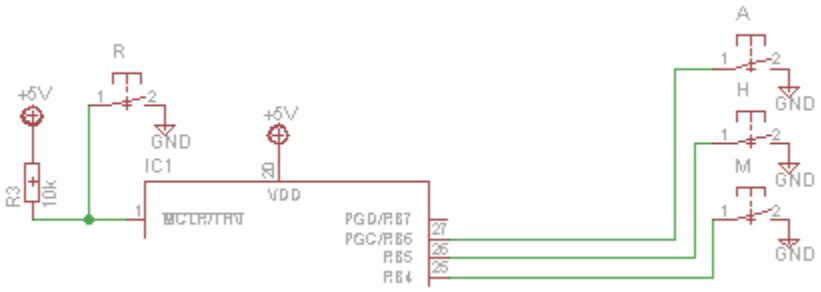
Slika 4: Multipleksiranje 7-segmentnega led prikazovalnika

Da bi prihranili večje število pinov mikrokrmilnika, lahko štiri 7-segmentne led prikazovalnike multipleksiramo. Namesto, da so ves čas, vsi štirje aktivni, lahko preko štirih tranzistorjev vklapljamo enega za drugim v tako hitrem časovnem intervalu, da z našimi očmi ne opazimo razlike, se pravi, se osvežujejo z dovolj visoko frekvenco.
Pin 5(dp) je aktiven samo pri 2 in 3 prikazovalniku, pri čemer je 3 zasukan za 180°, zaradi dvopičja.



Slika 5: Štirje 7-segmentni led prikazovalniki s skupno katodo

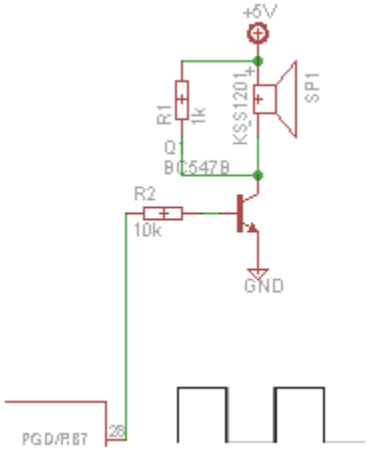
6.2.1.4 TIPKE



Slika 6: Tipke (A), (H), (M), (R)

Vse štiri tipke so momentalne (ON)/OFF. Tipki (H), (M) sta namenjeni nastavitvi ur in minut. S tipko (A) vklopimo alarm. Tipka (R) pa služi za »reset« naprave. Vhodni pini mikrokrmilnika RB4, RB5, RB6 so preko »softwerskih pull-up« uporov na visokem logičnem nivoju. MCLR pin je negiran, zato mora biti na visokem nivoju. Tu lahko spet uporabimo »pull-up« upor. S pritiskom katerekoli tipke, za kratek čas, spremenimo stanje iz visokega v nizki logični nivo (logična '1' v logično '0').

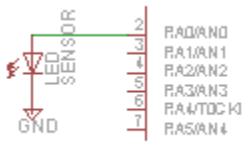
6.2.1.5 VEZJE ZA GENERIRANJE ZVOKA



Slika 7: Vezje za generiranje zvoka

Piskač začne oddajati zvok, ko se na njemu pojavi razlika napetosti v enakomernih časovnih intervalih. Piskanje ob alarmu, proizvedemo z enakomernimi pravokotnimi pulzi, ki odpirajo in zapirajo tranzistor. Ta, ko je odprt, povzroči padec napetosti 5V skozi piskač. Zaprt tranzistor pa onemogoči ta padec 5V in vzporedna vezava upora k piskaču poskrbi, da je padec napetosti čim manjši preko piskača, praktično 0V. Periodo pulzov lahko poljubno spreminjamo, ter tako posledično izbiramo željene frekvence piskanja.

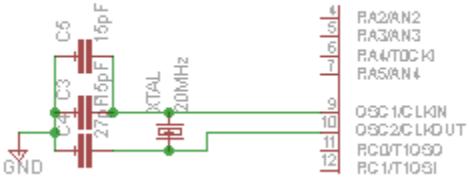
6.2.1.6 LED SENZOR



Slika 8: Led senzor

Vezava te svetlobne diode, omogoča prilagajanje osvetlitve 7-segmentnega prikazovalnika, glede na svetlobo okolice. Ponoči je ta intenzivnost osvetlitve manjša, podnevi večja. Pin RA0 na mikrokrmilniku definiramo kot vhodni pin za analogno-digitalno pretvorbo. Na diodi dobimo manjšo napetost, ki se spreminja glede na osvetljenost okolice. To napetost poberemo na vhodnem pinu in jo preko ADC-ja pretvorimo v nekaj bitno vrednost. Nato preko PWM-ja (pulse-width modulation) določimo osvetljenost prikazovalnika.

6.2.1.7 ZUNANJE VEZJE OSCILATORJA



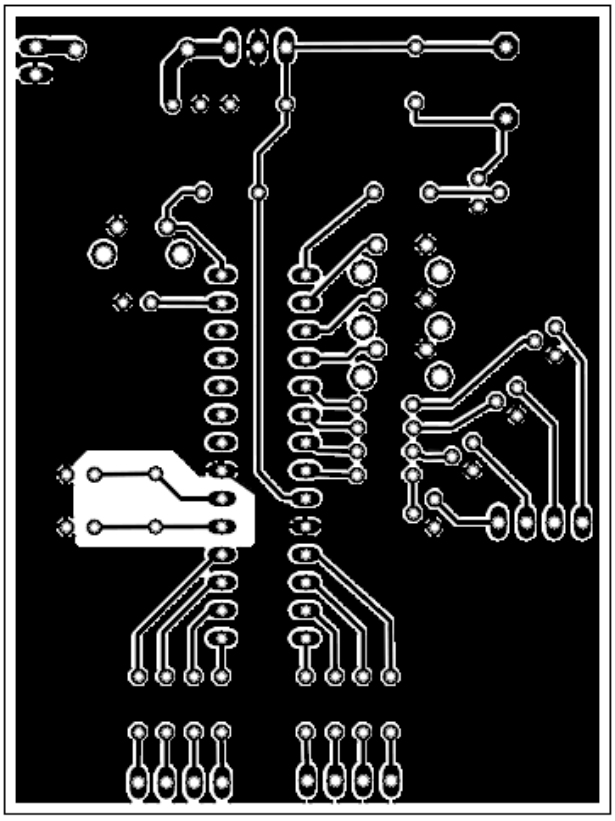
Slika 9: Zunanje vezje oscilatorja

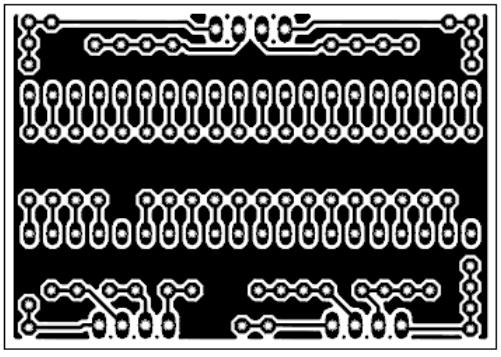
Da bi dosegli čim boljšo točnost osciliranja, je eden izmed pogojev, da se približamo kapacitivnosti bremena kristala (CL), ki jo ta zahteva. Ponavadi jo poda proizvajalec kristala. V mojem primeru, ta znaša 18pF. Velja naslednja formula:

$$C_L = \frac{C1 * C2}{C1 + C2} + C_S$$

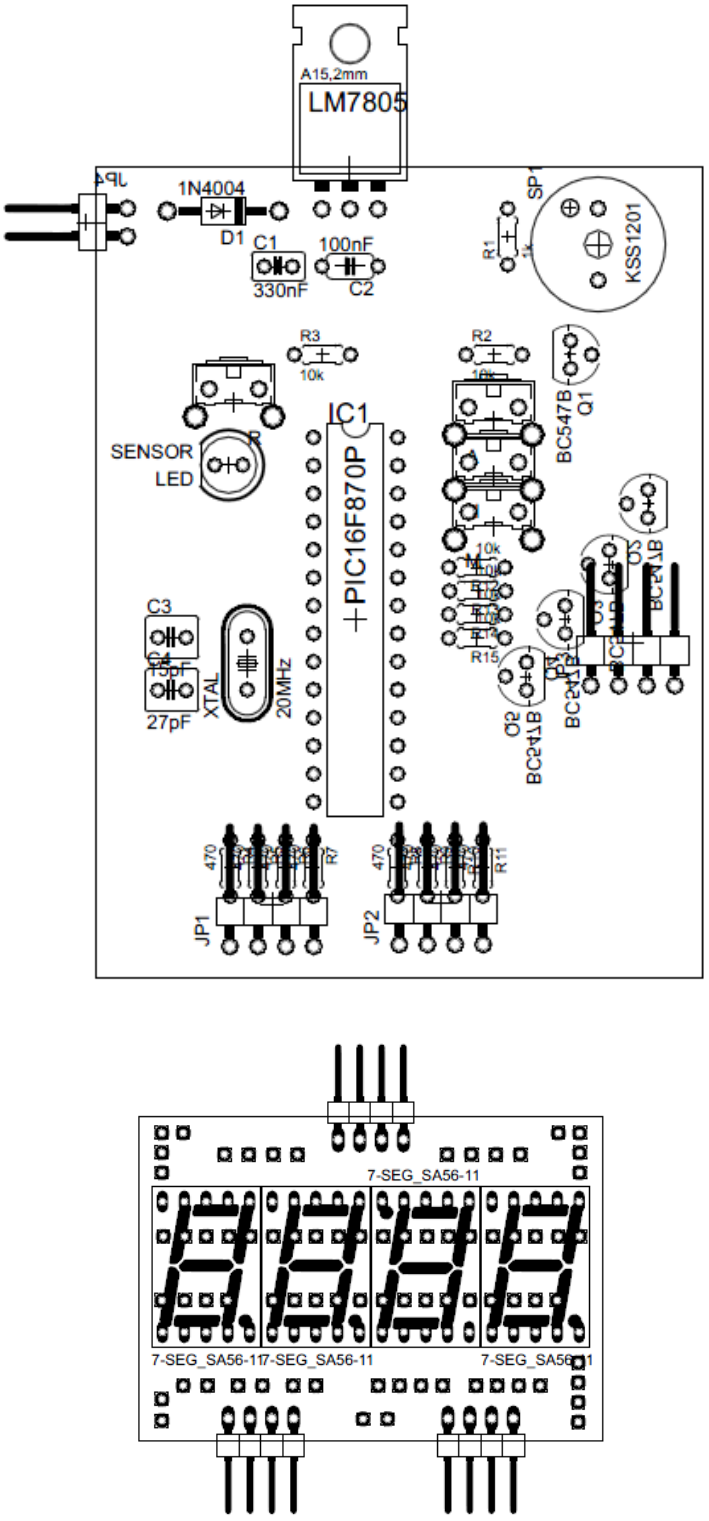
Za dosego te vrednosti moramo poznati vrednost C1 in C2 ter Cs (ostale kapacitivnosti, linij,.. ~5pF). Potrebujemo torej le še C1 in C2, katera sta ponavadi enaka. Vendar Cs ni vedno 5pF in je potrebno s »trimanjem«, to je, spreminjanje vrednosti vzporedno vezanega kondenzatorja k drugemu, zagotoviti dano CL. Šele v tem primeru bo kristal nihal s predpisano frekvenco.

6.3 TISKANO VEZJE IN POSTAVITEV ELEMENTOV





Slika 10: Tiskanina glavnega vezja in 7-seg-led prikazovalnika



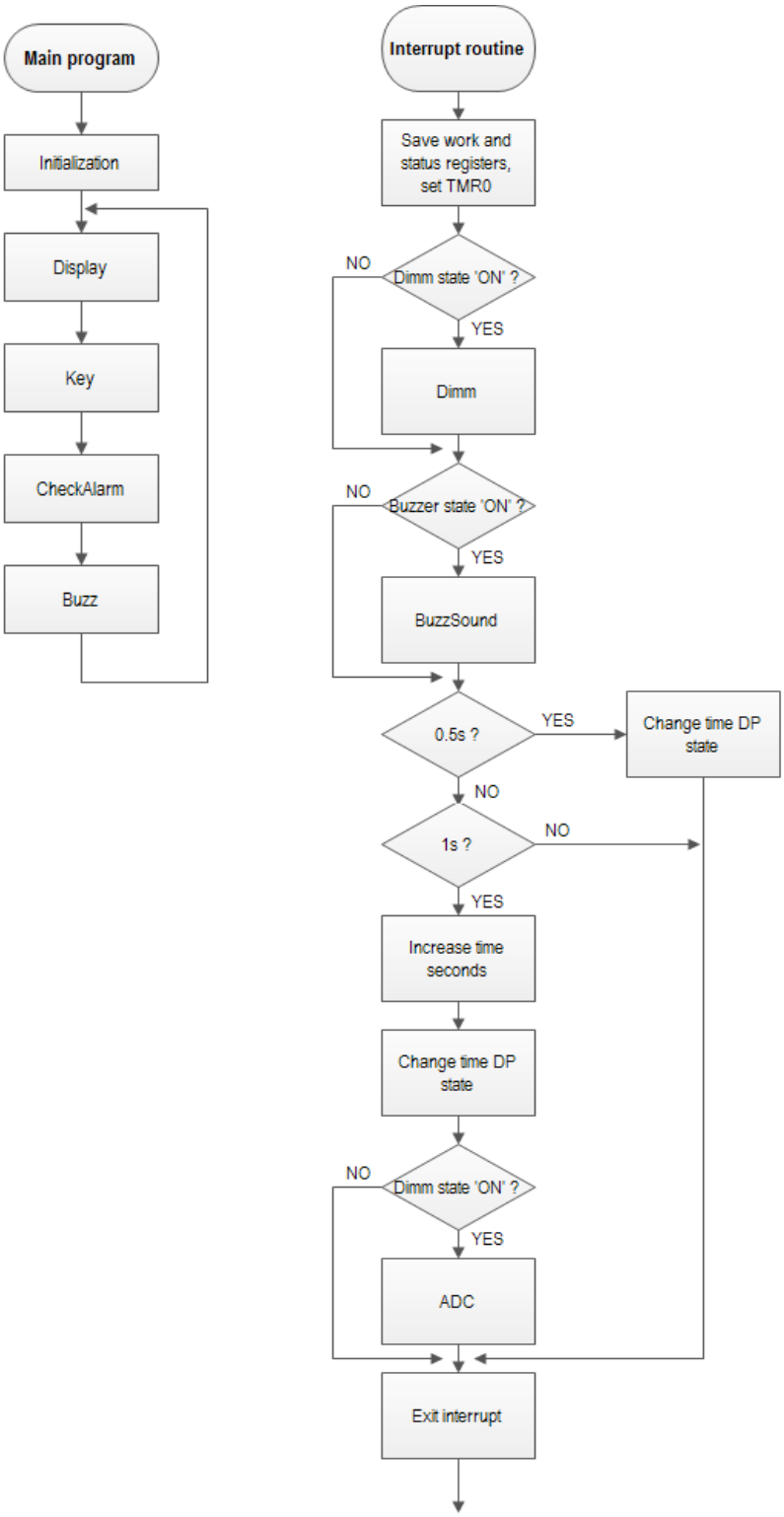
Slika 11: Postavitev elementov glavnega vezja in 7-seg-led prikazovalnika

6.4 OHIŠJE

Slika 12: Izdelek v pleksi ohišju

6.5 PROGRAM

Podan je bločni diagram poteka programa, ki je zaradi boljše preglednosti nad kodo, v angleščini.
Program sestoji iz dveh delov, glavnega programa in prekinitve. Vsakega sestavljajo še posamezne rutine.

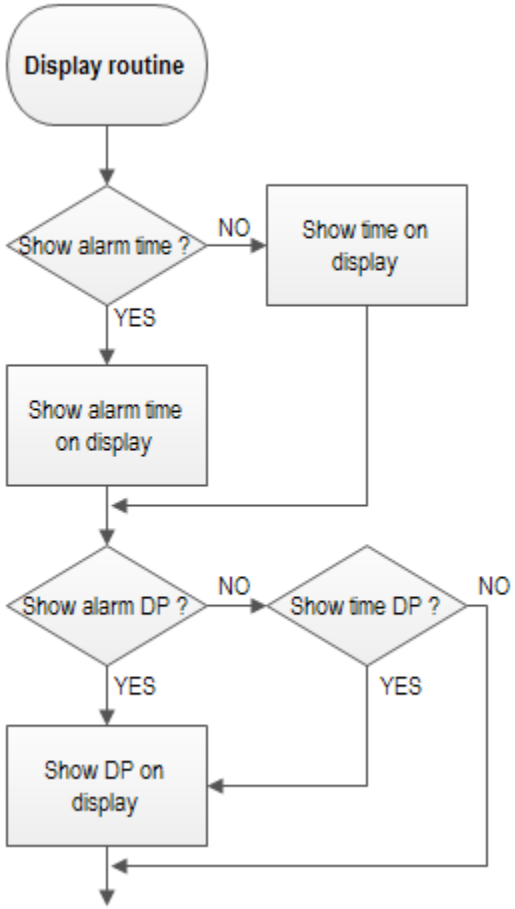


6.5.1 GLAVNI PROGRAM

INICIALIZACIJA

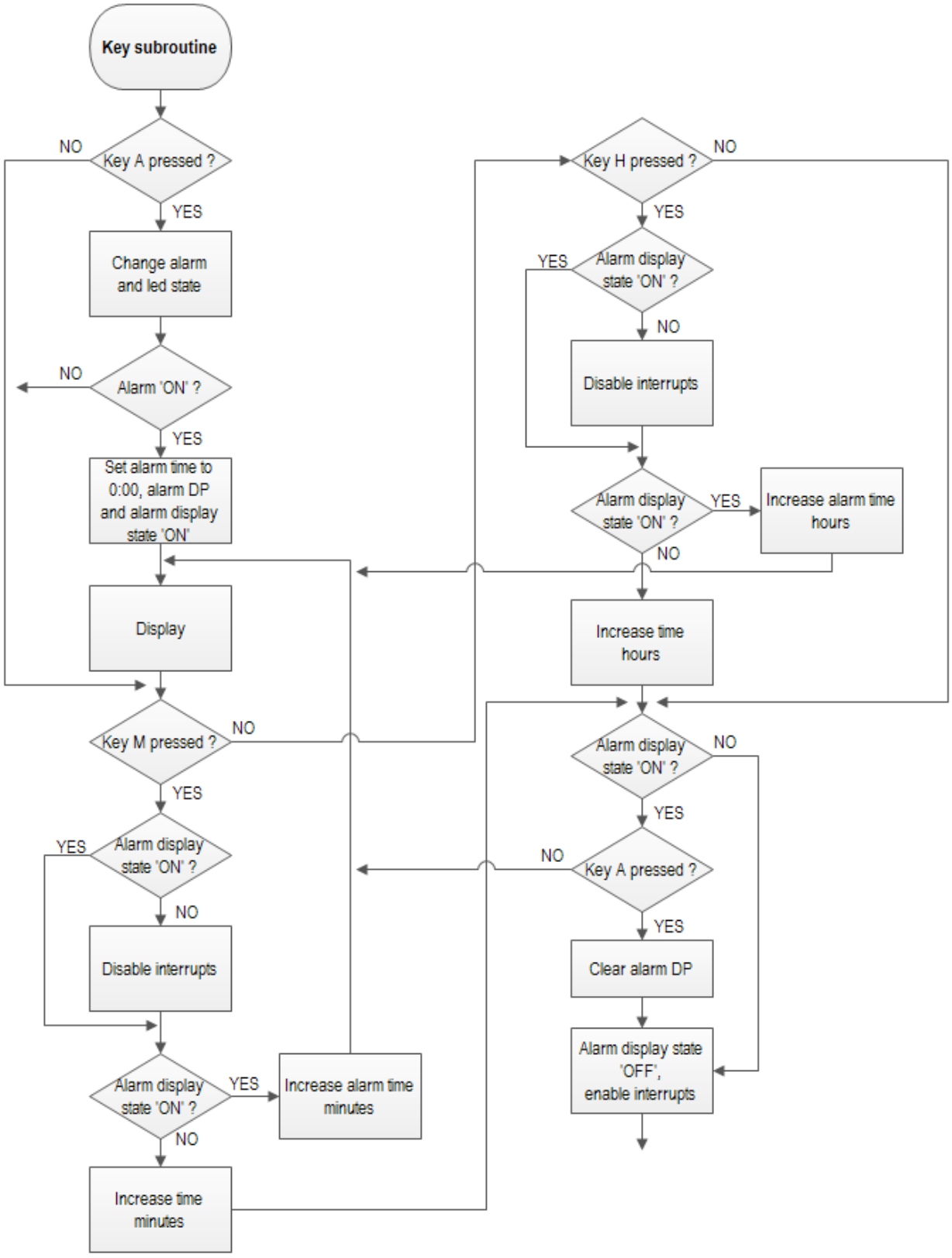
Ob vklopu ali »resetu« naprave, prvo kar se zgodi, je inicializacija programa. Tu se opravijo vse potrebne nastavitve mikrokontrolerja in pripravijo podatki za nadaljne delo.

DISPLAY



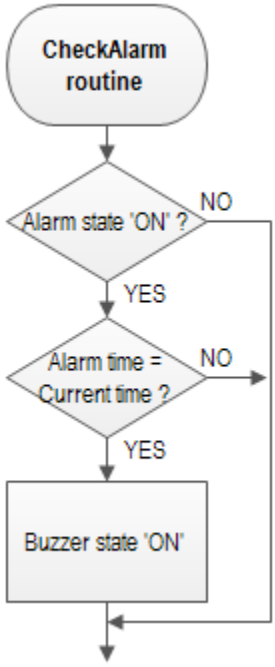
Ker multipleksiramo zaslon, prvo izberemo kateri od štirih prikazovalnikov bo aktiven, se odločimo ali prikazujemo trenutni čas ali čas alarma, ter poberemo številko (0 – 9, ali prazno), preko »data tabel« in jo prikažemo na izbranem prikazovalniku. Prikažemo še DP(dvopičje med urami in minutami) trenutnega ali alarmnega čas, če je tako zahtevano. DP trenutnega časa se pojavi vsako sekundo, alarmni pa je konstantno prisoten, ob nastavljanju alarma.

KEY


Key subroutine
Key A pressed ?
NO
YES
Change alarm and led state
Alarm 'ON' ?
NO
YES
Set alarm time to 0:00, alarm DP and alarm display state 'ON'
Display
Key M pressed ?
NO
YES
Alarm display state 'ON' ?
YES
NO
Disable interrupts
Alarm display state 'ON' ?
YES
Increase alarm time minutes
NO
Increase time minutes
Key H pressed ?
NO
YES
Alarm display state 'ON' ?
YES
NO
Disable interrupts
Alarm display state 'ON' ?
YES
Increase alarm time hours
NO
Increase time hours
Alarm display state 'ON' ?
NO
YES
Key A pressed ?
NO
YES
Clear alarm DP
Alarm display state 'OFF', enable interrupts

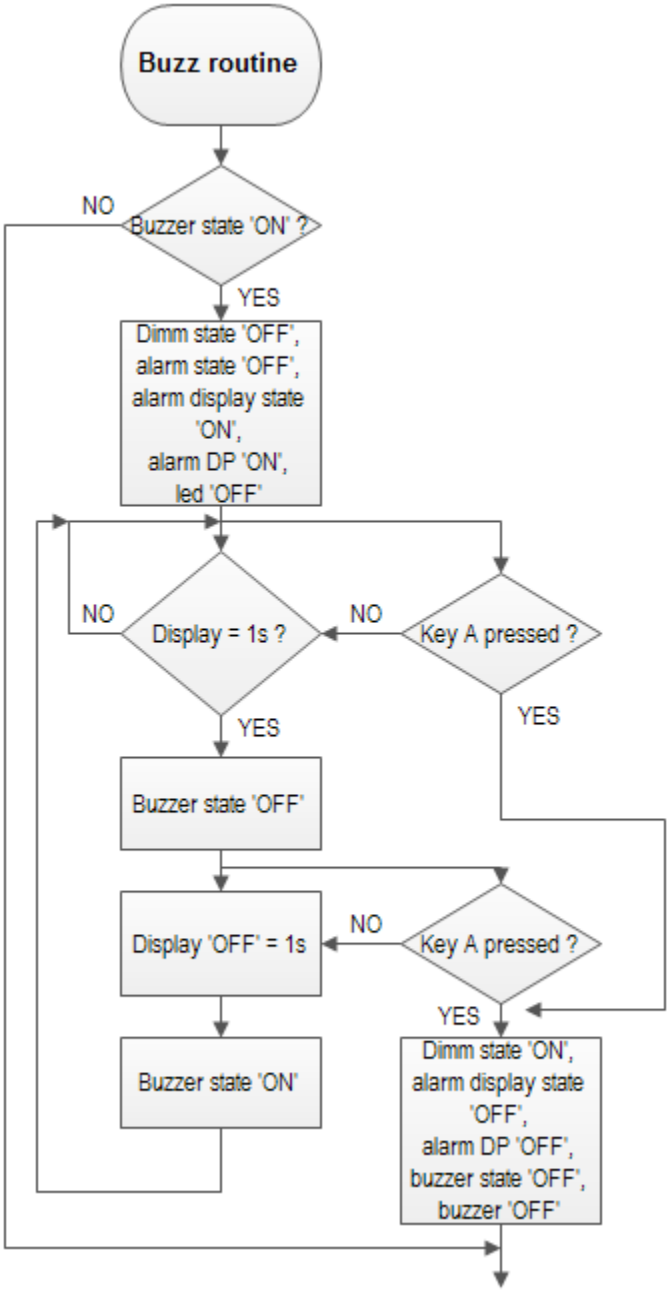
V tej rutini odčitavamo, ali je bila katera tipka pritisnjena. Ob pritisku tipke se sproži »softwerski debouncing«, s katerim se izognemo nevšečnostim ob pritisku tipke. Na primer, da bi ob enem pritisku tipke za povečanje minut, minute povečali za dve, namesto ene.
Začnimo s tipko (A), to je tipka, s katero nastavimo, vklopimo ali izklopimo alarm. Ob prvem pritisku se postavi alarm na aktivno stanje, kar indikira tudi ledica. Če ob nastavljenem alarmu pritisnemo to tipko še enkrat, deaktiviramo alarm. Ob prvem pritisku se na zaslonu prikaže prednastavljen čas (0:00). Če tipka (A) ni pritisnjena, gremo na naslednjo.
Tipka (M) je naslednja tipka, katero stanje odčitavamo. Če je pritisnjena, prvo poizvemo, katere minute naj povečamo (minute trenutnega časa ali minute alarma). Prekinitve se izključijo, če povečujemo minute trenutnega časa, saj nočemo, da bi ob nastavitvi željene minute, dobili drugo vrednost. Ker pa tudi nočemo, da bi po nastavitvi minute ,se ta takoj po sprostitvi tipke prelila v naslednjo, sekunde postavimo na 0. Ob nastavitvi minut alarma se ta korak obide. Te se takoj osvežijo na zaslonu. Če tipka (M) ni pritisnjena, gremo na naslednjo tipko (H), s katero nastavljamo ure. Postopek je enak zgornjemu.
Proti koncu rutine, čakamo na pritisk tipke (A), za potrditev alarma in izhod. Če ni nobena od naštetih tipk pritisnjena ali pa je bila narejena sprememba ure ali minute trenutnega časa, potem samo še vklopimo nazaj prekinitve in gremo ven iz ruitne.

CHECK ALARM



V tej rutini samo preverjamo ali se trenutni čas ujema z nastavljenim časom alarma in ga sprožimo.

BUZZ

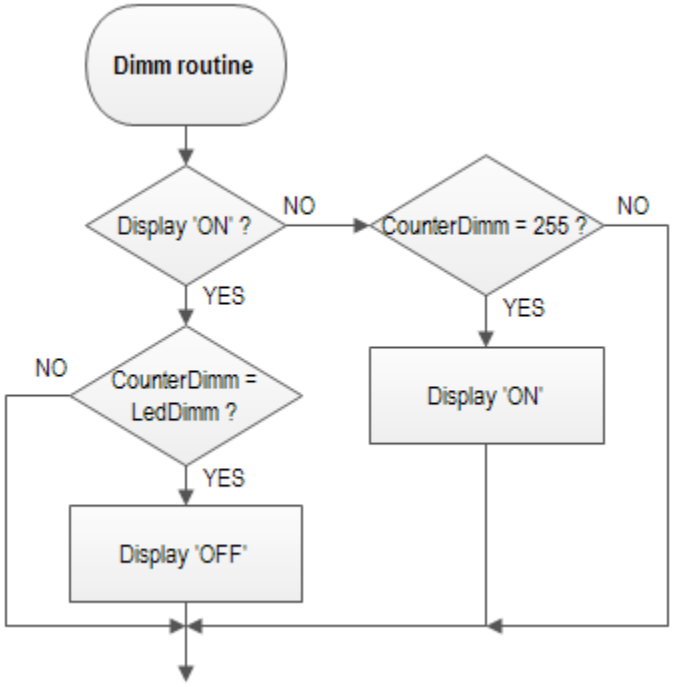


Ob ujemanju trenutnega časa z časom alarma, prenesemo ta čas na zaslon, onemogočimo prilagajanje osvetlitve zaslona, izklopimo led indikator alarma in aktiviramo piskač. Pustimo piskanje ter prikaz časa na zaslonu za eno sekundo, sproti odčitavamo ali je bila pritisnjena tipka (A), s katero bi izklopili alarm. Po preteku ene sekunde izklopimo zaslon in piskanje ter spet odčitavamo spremembo na tipki (A), ponovno za eno sekundo, nato se cikel ponovi. Po pritisku tipke sledi izhod iz rutine, pri čemer izklopimo piskač ter prevzamemo nastavitve prisotne pri prikazu trenutnega časa.

6.5.2 PREKINITEV

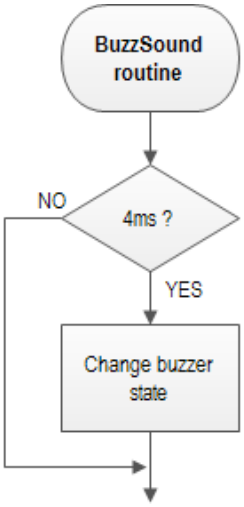
Prekinitvena rutina se sproži vsakih 50us. Na začetku shranimo vse določene registre in osvežimo vrednost TMR0.

DIMM



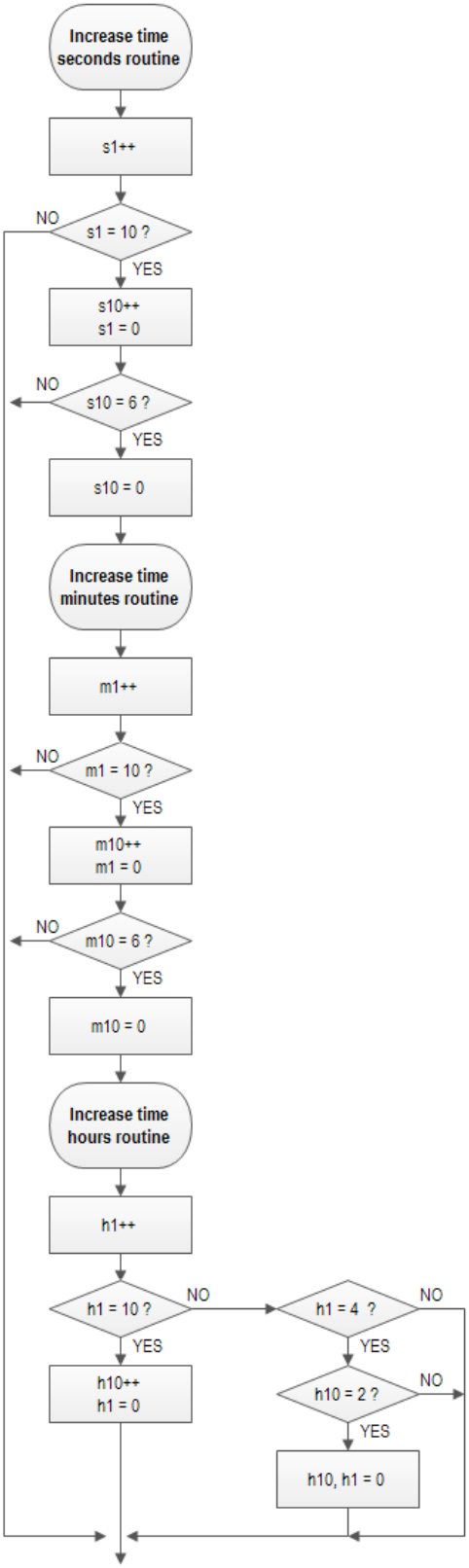
Prilagajanje osvetlitve zaslona omogočeno (izklopljeno samo, ko se sproži alarm). Gre za PWM (Pulse-Width-Modulation), s katerim vklapljamo ali izklapljamo zaslon. To razmerje je odvisno od vrednosti, ki jo poberemo skozi ADC.

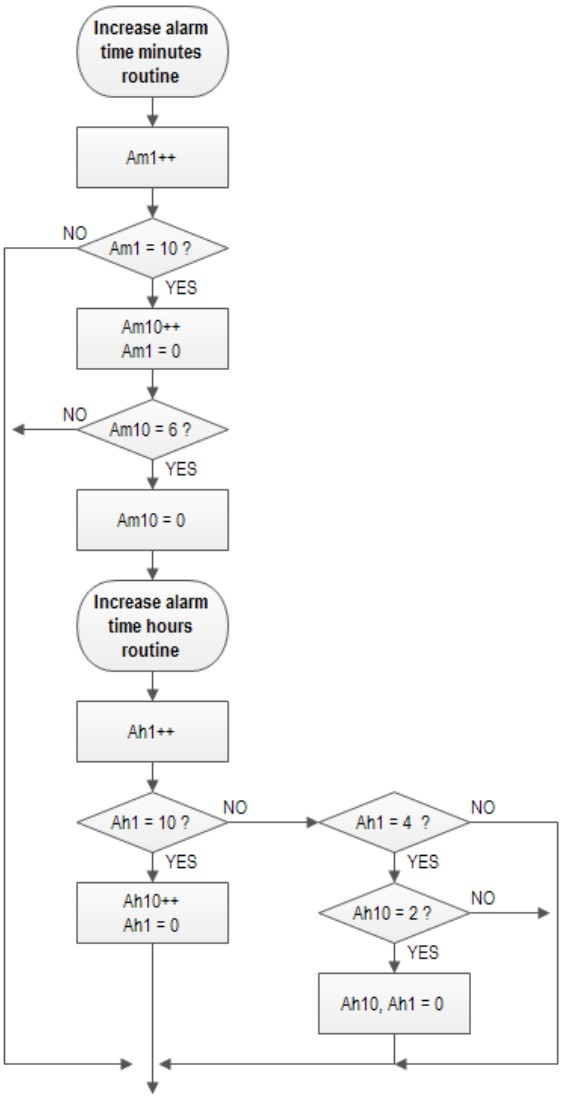
BUZZ SOUND



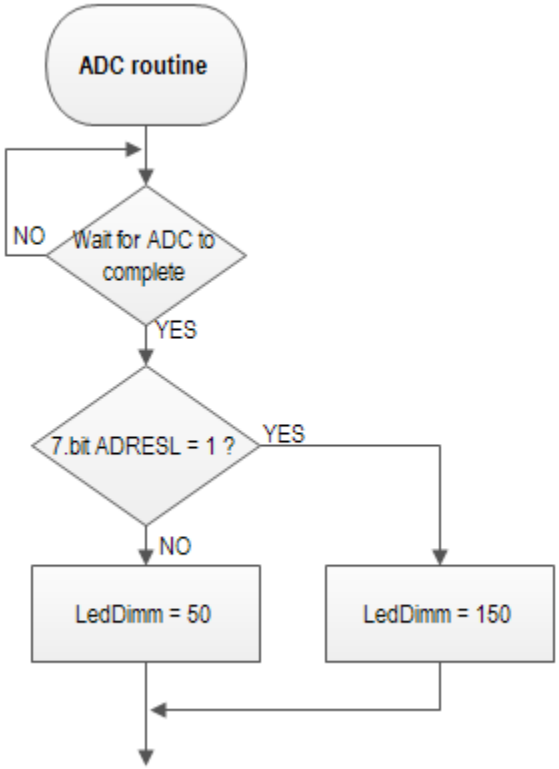
Ob sprožitvi alarma, spreminjanje stanja na piskaču vsake 4ms, »rodi« piskanje.

INCREASE TIME SECONDS (MINUTES, HOURS) AND ALARM MINUTES, HOURS





ADC

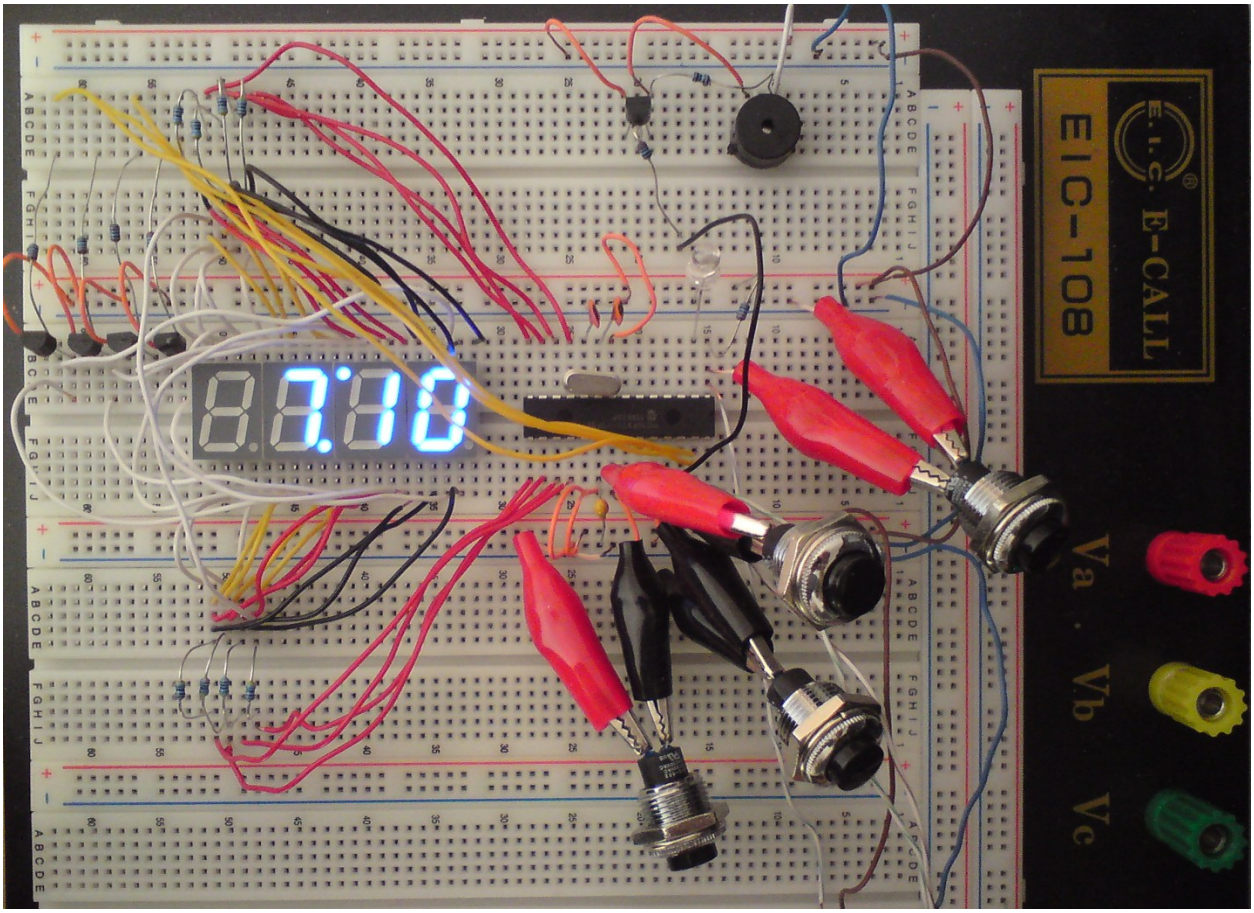


ADC pretvorba nastopi vsako sekundo. Nekaj časa vzame za pridobivanje bitne vrednosti, iz vhodnega pina, na katerega je vezana dioda. Pridobljeno vrednost zapiše v register ADRESL. Rezultat sem razdelil glede na 7.bit v ADRESL registru, ki pove ali je prostor zatemnjen (0) ali svetel (1). Temu prirejeni vrednosti uporabimo za PWM.

Podrobna razlaga ukazov, je komentirana v sami kodi.

7. TESTIRANJE NAPRAVE

Vezje, ki sem si ga zamislil za digitalno uro sem prvo preizkusil na testni plošči, »protoboardu«, ki je zaradi vsestranskosti primeren za nadaljne potrebne modifikacije.



Slika 13: Preizkusno vezje na protoboardu

Samo vezje niti ni tako kompleksno, se bom pa v nadaljevanju posvetil načrtovanju točnosti ure, tako »softwersko« kot »hardwersko«, kar je večji izziv. Ponavadi se pri digitalni uri uporabi kak RTC čip, vendar tudi brez njega se da doseči zadovoljivo točnost, do +/- 1 minute na leto. Par besed bo namenjenih tudi o prilagajanju osvetljenosti zaslona glede na zunanjo svetlobo.

7.1 TOČNOST URE

PROGRAMSKI DEL

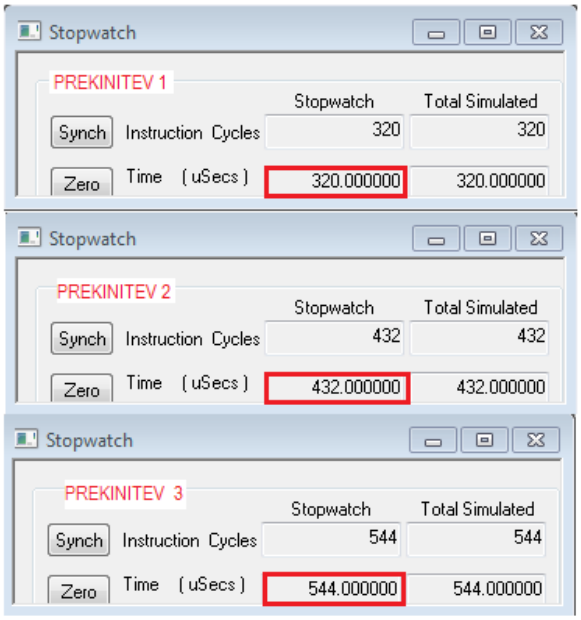
Programska koda je bila spisana v microchipovem okolju MPLAB, v jeziku assembler.
Čas se povečuje v prekinitveni rutini, zato se mora ta vedno zgoditi v enakih časovnih intervalih.

Ponavadi je koda generiranja prekinitve z TMR0, napisana takole:

Primer 1

Kristal: 4MHz, instrukcijski cikel: 1us, preddelilnik: 1:1

```
;******************** INTERRUPT ***************************;

Int                             ; 100us
   bcf    INTCON,GIE            ; disable interrupts
   movwf  Save_w                ; save work
   swapf  STATUS,w
   movwf  Save_s                ; save status
   bcf    INTCON,T0IF           ; clear TMR0 interrupt flag
   movlw  .155
   movwf  TMR0                  ; set TMR0 = 155
B      -
       -
       -

ExitInt                         ; exit interrupt
   swapf  Save_s,w
   movwf  STATUS                ; restore status register
   swapf  Save_w,f
   swapf  Save_w,w              ; restore work register
   bsf    INTCON,GIE            ; enable interrupts
   retfie

;**********************************************************;
```



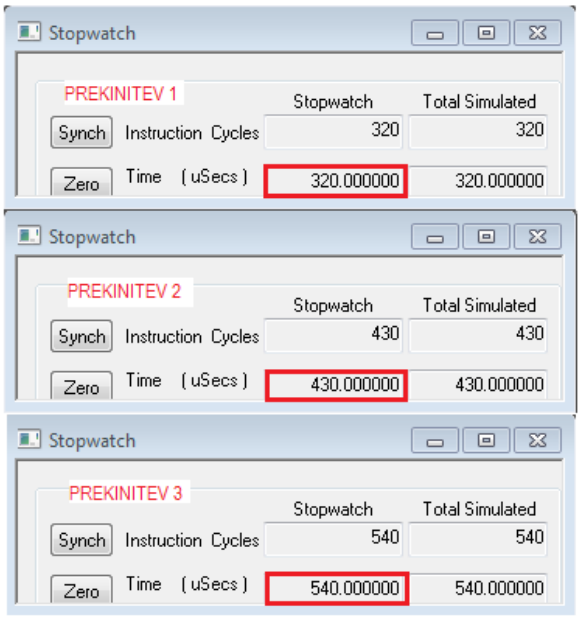
Slika 14: Prvi primer netočne prekinitve

Velja, da na začetku prekinitve določene regitre shranimo, pobrišemo zastavico, ki označuje, da je prišlo do preliva TMR0 registra, itd. Nato zapišemo vrednost v TMR0, ki pove čez koliko časa se naj spet sproži prekinitev. Naša prekinitev naj bi bila 100us, zato zapišemo v TMR0 vrednost 155, saj od 155 do 255 je točno 100us, vendar rezultati, ki so bili narejeni z MPLAB simulatorjem, na desni strani pričajo drugače. Prva prekinitev je logično več, saj smo ob inicializaciji pobrisali vrednost TMR0, druga prekinitev se zgodi čez 112us, tretja prav tako in tako naprej. Se pravi odstopanje od 100us, se s prekinitvami sešteva.

Primer 2

Kristal: 4MHz, instrukcijski cikel: 1us, preddelilnik: 1:1

```
;******************** INTERRUPT ***************************;

Int                             ; 100us
   bcf    INTCON,GIE            ; disable interrupts
   movwf  Save_w                ; save work
   swapf  STATUS,w
   movwf  Save_s                ; save status
   bcf    INTCON,T0IF           ; clear TMR0 interrupt flag
   movlw  .157
   movwf  TMR0                  ; set TMR0 = 157
B      -
       -
       -

ExitInt                         ; exit interrupt
   swapf  Save_s,w
   movwf  STATUS                ; restore status register
   swapf  Save_w,f
   swapf  Save_w,w              ; restore work register
   bsf    INTCON,GIE            ; enable interrupts
   retfie

;**********************************************************;
```



Slika 15: Drugi primer netočne prekinitve

Ob vpisu vrednosti v TMR0, se ta register začne povečevati šele čez čas 2 strojnih ciklov, kar pomeni, da moramo vrednost števca povečati za 2. Prekinitev se tako pojavi na 110us, kar pomeni, da smo malo izboljšali rezultat, vedno še vedno odstopa od idealnih 100us.

```
;******************** INTERRUPT ***************************;

Int                             ; 100us
   bcf    INTCON,GIE            ; disable interrupts
   movwf  Save_w          ?     ; save work
   swapf  STATUS,w
   movwf  Save_s                ; save status
   bcf    INTCON,T0IF           ; clear TMR0 interrupt flag
   movlw  .157
   movwf  TMR0                  ; set TMR0 = 157
B      -
       -
       -

ExitInt                         ; exit interrupt
   swapf  Save_s,w
   movwf  STATUS                ; restore status register
   swapf  Save_w,f
   swapf  Save_w,w              ; restore work register
   bsf    INTCON,GIE            ; enable interrupts
   retfie

;**********************************************************;
```

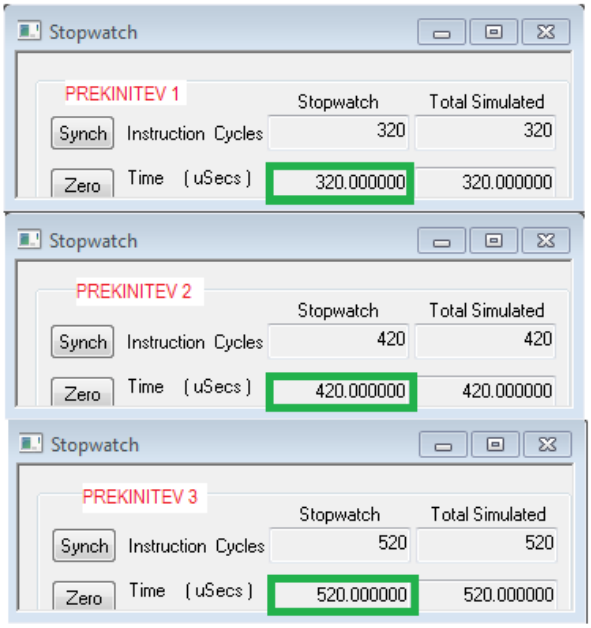
Slika 16: Drugi primer netočne prekinitve - rešitev ?

Kar smo pozabili, so ukazi pred vpisom nove vrednosti v TMR0. Njihovo trajanje nismo upoštevali v določanju vrednosti TMR0.

Primer 3

Kristal: 4MHz, instrukcijski cikel: 1us, preddelilnik: 1:1

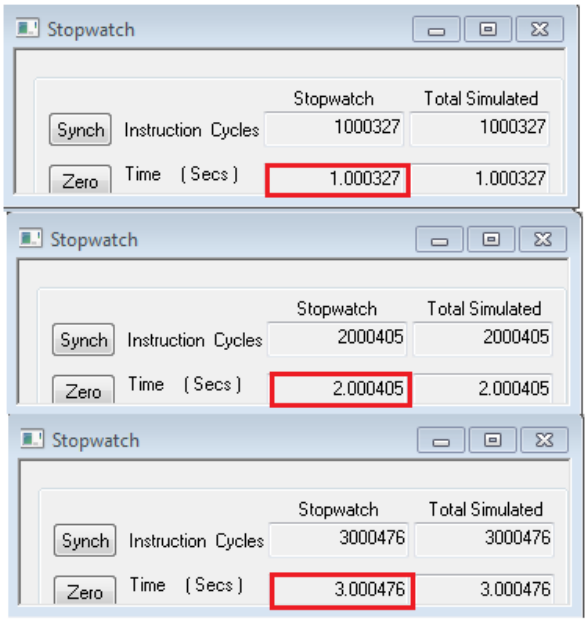
```
;******************** INTERRUPT ************************;

Int                             ; 100us
   bcf    INTCON,GIE            ; disable interrupts
   movwf  Save_w                ; save work
   swapf  STATUS,w
   movwf  Save_s                ; save status
   bcf    INTCON,T0IF           ; clear TMR0 interrupt flag
   movlw  .167                  ;
   movwf  TMR0                  ; set TMR0 = 167
          -
          -
          -

ExitInt                         ; exit interrupt
   swapf  Save_s,w
   movwf  STATUS                ; restore status register
   swapf  Save_w,f
   swapf  Save_w,w              ; restore work register
   bsf    INTCON,GIE            ; enable interrupts
   retfie

;*********************************************************;
```



Slika 17: Tretji primer netočne prekinitve - rešitev

Tukaj odigra svojo vlogo simulator, s katerim nastavimo pravilno vrednost TMR0. Tu upoštevamo, da skok v prekinitev vzame 2 cikla in pa ukaze v vprašaju. Prekinitev, kot vidimo, je točno 100us.

Če bi nadaljevali simulacijo, bi na vsake toliko časa, prekinitev trajala 1us več. Kar se zgodi je, da nekateri ukazi (goto, call, return,..), trajajo 2 cikla in če se bi se prekinitev morala sprožiti po prvem ciklu tega ukaza, se ta izvrši z zaostankom 1 strojnega cikla, saj se ukaz izvrši v celoti.



Slika 18: Tretji primer, sekunde

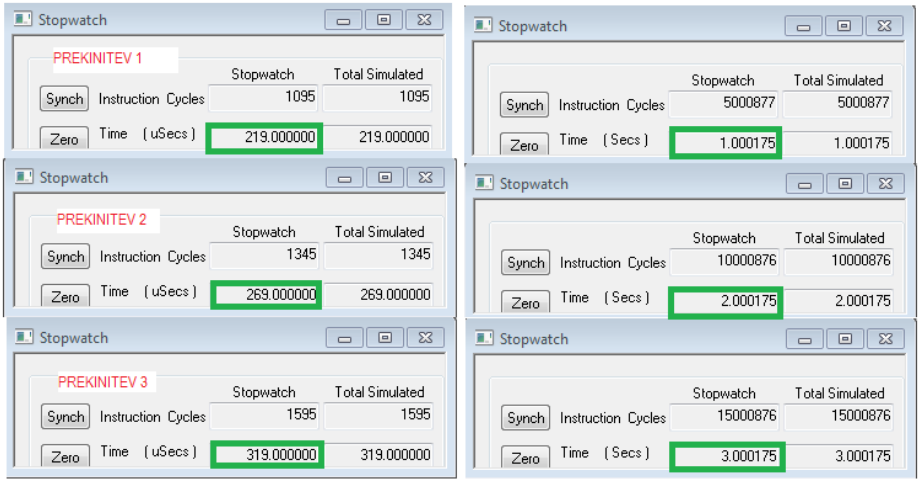
Prekinitev v projektu

V mojem primeru, se prekinitvena rutina sproži vsakih 50us. Uporabljam 20MHz kristal, se pravi en strojni cikel (instrukcija) traja 0.2us, nekatere instrukcije porabijo tudi dva cikla. Za proženje prekinitve uporabljam časovnikTimer0, oziroma register TMR0, ki sproži prekinitev, ko se njegova vrednost prelije iz 255 v 0. Prescaler je nastavljen na 1:4, se pravi, da se vrednost TMR0 poveča na 4 strojne cikle.

```
;********************************* INTERRUPT ***********************************;

Int                             ; 50us
   bcf    INTCON,GIE            ; disable interrupts
   btfsc  TMR0,0
   goto   $+.3
   nop                          ; TMR0 = 0 (2x 0,2us (int), 1x 0,2us)
   nop
   nop                          ; TMR0 = 1 (1x 0,2us (1cy), 2x 0,2us (int), 1x 0,2us)
   nop
   nop
   movwf  Save_w                ; save work
   swapf  STATUS,w
   movwf  Save_s                ; save status
   bcf    INTCON,T0IF           ; clear TMR0 interrupt flag
   movlw  .198                  ;
   movwf  TMR0                  ; set TMR0 = 198
B      -
       -
       -


ExitInt                         ; exit interrupt
   swapf  Save_s,w
   movwf  STATUS                ; restore status register
   swapf  Save_w,f
   swapf  Save_w,w              ; restore work register
   bsf    INTCON,GIE            ; enable interrupts
   retfie

;*******************************************************************************;
```

Slika 19: Način izpopolnjevanja točnosti časa

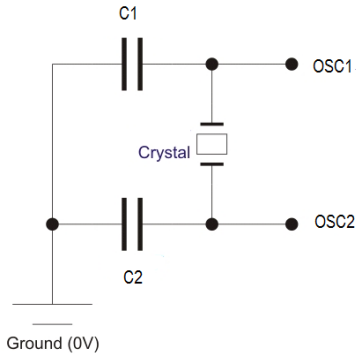
V primeru nastopa prekinitve brez zamude imamo v prvi vrstici kode, 2 strojna cikla (vhod v prekinitev), v drugi 3, kar pomeni, da je TMR0 še vedno 0, saj potrebuje 4 cikle, da se poveča. Ukaz *btfsc* ob preskoku porabi 2 cikla in preostane nam še 5 ukazov brez operacije (*nop*), skupno porabimo 10 ciklov do devete vrstice.
V primeru zamude prekinitve za 1 strojni cikel (ukaz z dvema cikloma se je moral zaključiti), imamo 3 cikle v prvi vrstici, v drugi 4, TMR0 se postavi na 1. Ukaz *btfsc* ne preskoči, porabi 1 cikel, *goto* ukaz traja 2 cikla. Ostane nam še 3 ukazi *nop*, skupno torej spet 10 ciklov do devete vrstice. Na tak način sem uskladil čas do zapisa nove vrednosti v TMR0, ne glede na to, kateri od zgornjih dveh primerov se pojavi.



Slika 20: Prekinitve in sekunde

ZUNANJE VEZJE OSCILATORJA

Zdaj, ko smo programsko dosegli točnost, nam preostane še načrtovanje vezje, ki bo prinašalo čim manj napake k času.



Slika 21: Vezje kristala oscilatorja

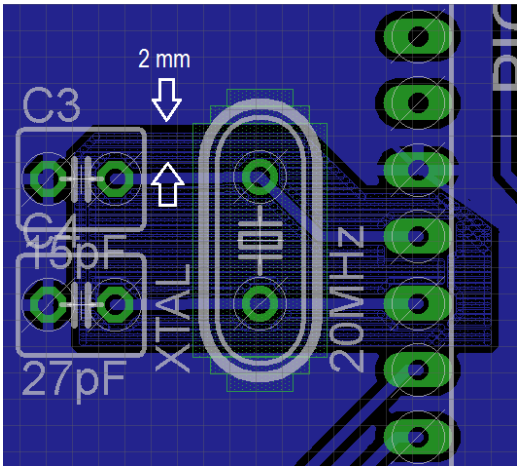
Kristal ima podano bremensko kapacitivnost ali »load capacitance« CL, pri kateri bo nihal točno s predpisano frekvenco.

$$C_L = \frac{C1 * C2}{C1 + C2} + C_S$$

C1 in C2 kondenzatorja sta ponavadi iste vrednosti (15-33pF).
Cs (stray capacitance,) je »nezaželena« kapacitivnost pinov, linij,..(3-15pF).

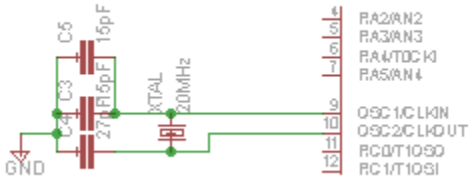
CL pri mojem 20MHz kristalu znaša 18pF.
Na protoboardu je kristal nihal točno pri C1=C2=22pF, kar pomeni, da je bila kapacitanca linij 7pF. Na protoboardu se pričakuje veliko parazitno kapacitivnost.

Med načrtovanjem tiskanega vezja sem se hotel znebiti nezaželenih kapacitivnosti, zato sem izoblikoval »otoček«, ki »ščiti« kristal pred nezaželenimi efekti drugih signalov.



Slika 22: »Varovalni otoček«

Ko sem kristal testiral z 22pF kondenzatorjema, je čas močno prehiteval, kar pomeni, da se je nezaželena kapacitivnost zmanjšala. Z naslednjo standardno vrednostjo 27pF sem se že približal točnemu času, saj je prehiteval le 1,5s/dan. 33pF pa je že močno upočasnilo nihanje.
Nezadovoljen z točnostjo, sem se poslužil druge metode, s katero sem reguliral vrednost vzporedno vezanega kondenzatorja C3, k C1. Vzporedno vezanima kondenzatorjem se sešteva kapacitivnost in tako lahko najdemo prave vrednosti(C1=15pF, C2=27pF in C3=15pF), ki zagotavljajo dober približek CL in s tem točnost ure (1s/teden).
S pomočjo zgornje enačbe lahko določimo še Cs, ki znaša 3,8pF.



Slika 23: Zunanje vezje oscilatorja

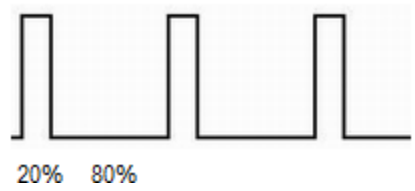
7.2 REGULIRANJE OSVETLJENOSTI ZASLONA

Podatek o svetlobi v prostoru dobimo preko led diode, ki je vezana na naslednji način.

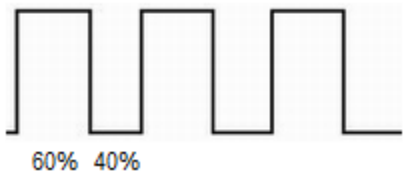
Slika 24: Led senzor

RA0 je definiran kot vhodni analogni pin.
Na diodi dobimo manjšo napetost, ki se spreminja glede na osvetljenost prostora. Preko ADC-ja analogno vrednost napetosti spremenimo v binarno število ter jo shranimo v ADRESL register. Ob slabši osvetljenosti se to število giblje od (0000 0000–**0**111 1111 = 0-127), ob boljši pa od (**1**000 0000–1100 1000 = 128-200). Vsakemu območju priredimo vrednost, preko 7.bita, ki ločuje dan od noči in jo uporabimo za PWM, s katerim prižigamo in ugašamo zaslon.

PWM



Slika 25: Osvetlitev zaslona ob slabši svetlobi



Slika 26: Osvetlitev zaslona ob boljši svetlobi

8. KOSOVNICA

referenca	vrednost	ime	ohišje	število	proizvajalec	cena(€)/kos
mikrokrmilnik	/	PIC16F870	DIP	1	Microchip	3.13
R4–R11, R16	470Ω	upor	Axial	9	Multicomp	0.035/10
R2,R3,R12-R15	10kΩ	upor	Axial	6	Multicomp	0.035/10
R1	1kΩ	upor	Axial	1	Multicomp	0.035/10
C4	27pF, 100V	kondenzator	Radial	1	Vishay	0.155/10
C3,C5	15pF, 50V	kondenzator	Radial	2	Multicomp	0.2/25
C2	0.1uF, 35V	kondenzator	Radial	1	Multicomp	0.25/5
C1	0.33uF, 35V	kondenzator	Radial	1	Multicomp	0.25/5
LED1, LED SENSOR	Super-Blue	LED	Radial	2	Multicomp	0.23/5
LED2	Red	LED	Radial	1	SPC Technology	0.084/5
Q1-Q5	BC547B	tranzistor	TO-92	5	Multicomp	0.147/5
LM7805	5V	nap. regulator	TO-220	1	Fairchild Semiconductor	0.91
R,A,M,H	/	tipke	/	4	Multicomp	1.84
7 SEG LED1,2,3,4	Blue	7-SEG-LED	/	4	Forge Europa	5.07
JP1-JP4	6-way	pin-header	/	4	Harwin	0.31/10
XTAL	20MHz	kristal	/		TXC	0.54
Batery	9V	/	/	1	XCell	10.8
Ohišje	/	/	/	1	Acrytech	40
pcb	/	/	/	2	FE	/

9. TEHNIČNE SPECIFIKACIJE

- Napajanje, omrežje 230V/50Hz AC, adapter 9V DC ali baterija 9V
- 5V napetostni regulator
- Štirje 7-segmenti LED prikazovalniki
- Mikrokrmilnik PIC16F870 (8-bitni, FLASH)
- 4 tipke
- 2 barvni led diodi
- 1 led dioda / senzor
- Odstopanje časa: < +/- 1min/leto
- Pleksi ohišje
- Mere ohišja: 100x70x50 mm
- Teža: 860 gramov

10. NAVODILA ZA UPORABO

Ob vklopu ali resetu ure, čas začne teči od 0:00. Ura ima prednostno napajanje iz omrežja. Če ta ni prisoten, se ura napaja preko 9V baterija, kar označuje rdeča led dioda. Za nastavljanje ure uporabimo tipki (H) za ure ali (M), za minute. Vsak pritisk poveča le to. Za nastavitev alarma uporabimo tipko (A). Prednastavljen čas alarma je 0:00. Po nastavitvi željenega časa potrdimo nastavitev alarma spet s pritiskom tipke (A). Da je alarm vklopljen, priča modra led dioda. Ob prehodu na letni/zimski čas mora uporabnik sam posodobiti čas. Ura ima tudi možnost ponovnega vklopa preko tipke (R).

11. ČASOVNI IN FINANČNI PREGLED

Časovni pregled

Kot je bilo načrtovano, mi je projekt vzel v grobem 3 mesece.
Slab mesec programiranja (8ur/dan).
2 tedna dopusta.
En mesec in pol testiranj in čakanja na ohišje (3 tedni).

Finančni pregled

komponente 55,4 €
+ ohišje 40 €
~ 95 €

Račun je skoraj še enkrat višji, predvsem zaradi dragih 7-segmentnih led prikazovalnikov. Ta izdelek za svetovni trg res ni konkurenčen ostalim uram (predvsem tudi zaradi neobičajnega (in dragega) ohišja).

11. REFERENCE

- Knjiga Programirajmo PIC mikrokontrolerje
- https://sites.google.com/site/programiranjepic/home
- http://www.electronicecircuits.com/electronic-circuits/pic-16f84-12-24-hour-clock-circuit-and-programming
- http://www.kemisa.es/english/digital_clock_pic16f84_circuit.php
- http://www.mstracey.btinternet.co.uk/pictutorial/picmain.htm
- http://www.micro-examples.com/public/microex-navig/doc/096-led-light-sensor
- Microchip forum
- Microchip datasheets
- http://ww1.microchip.com/downloads/en/devicedoc/30569b.pdf
- Elektronik.si forum

12. PRILOŽENE DATOTEKE

Prilagam:

- Dokumentacijo (poročilo, predstavitev (Powerpoint))
- Kodo (.asm)
- Gerber mapa
- Eagle mapa (shema načrta, tiskanina)