UNIVERZITET U NIŠU
ELEKTRONSKI FAKULTET
Aleksandra Medvedeva 14, Niš

OSNOVNE AKADEMSKE STUDIJE
Modul: ELEKTRONIKA

PROJEKTNI RAD

SISTEM ZA UPRAVLJANJE OSVETLJENJEM UNUTAR JEDNOG OBJEKTA

MASTER AUTOMAT
Projektovanje elektronskih sistema

M E N T O R: Dr Borisav Jovanović

S T U D E N T I: Aleksa Ranđelović, 17378
Nikoleta Živić, 17109

Niš, januar 2021.

SADRŽAJ

1. PROJEKTNI ZADADATAK

Projektovati sistem za upravljanje osvetljenjem koji omogućava kontrolu nivoa osvetljenosti unutar jednog objekta. U objektu postoji 16 prostorija i u svakoj od njih nalazi se po jedan mikrokontroler koji upravlja osvetljenjem. Pored toga u sistemu postoji i centralni PC računar. Mikrokontroleri u prostorijama imaju sledeće funkcije:

- Mere osvetljenost unutar prostorije i napolju.
- Automatski podešavaju ugao pod kojim stoje trake brisoleja, kao i intezitet svetla koji daju sijalice u prostoriji, kako bi se održao definisani nivo osvetljenosti.
- Omogućava uštedu energije, tako što se sijalice pale tek onda kada osvetljenost ne može dalje da se pojačava daljim otvaranjem brisoleja.
- Poseduje senzore prisustva u svakoj prostoriji, tako da se električno osvetljenje uključuje samo ako je u prostoriji neko prisutan.
- Imaju 4 predefinisane postavke za osvetljenost: sunčan dan (maksimalno otvoreni brisoleji, svetlo ugašeno), privatnost (zatvoreni brisoleji, svetlo srednjeg intezitita), intima (zatvoreni brisoleji, prigušeno svetlo), noć (zatvoreni brisoleji, ugašeno svetlo).
- Poseduje tastaturu kojima se može izabrati jedan od predefinisanih postavki osvetljenosti ili režim automatske kontrole, manuelno pojačanje ili smanjenje osvetljenosti.
- Detektuju otkaz sijalica i tu informaciju šalju PC računaru.
- Komuniciraju sa centralnim računarom.

PC računar treba da omogući:

- Prikupljanje informacija sa mikrokontrolera u prostorijama objekta.
- Definisanje nivoa osvetljenosti koji treba da se postigne za svaku prostoriju posebno.
- Promenu predefinisanih postavki za osvetljenost, kao i dodavanje novih.

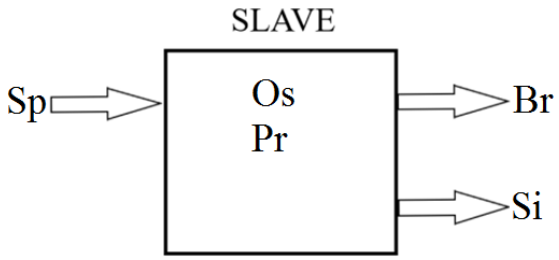
2. FUNKCIONALNA I NEFUNKCIONALNA SPECIFIKACIJA

2.1. FUNKCIONALNA SPECIFIKACIJA

U cilju ispunjenja funkcionalnosti sistema koja je definisana zadatkom, formirana je arhitektura u kojoj postoji centralni elektronski uređaj (u daljem tekstu Master automat) i šesnaest elektronskih uređaja (Slave automat) koji se instaliraju u prostorijama. Master automat povezan je na centralni računar i koristi se za kontrolu i prikupljanje informacija sa Slave automata. Sve prikupljene informacije prikazuju se na monitoru računara i dostupne su osobi koja vodi računa o stanju u prostorijama. Slave automati primaju informacije sa senzora koji su instalirani u nekoj određenoj prostoriji. Slave automati prosleđuju prikupljene informacije sa senzora dalje Master automatu.

Kako je tema ovog poglavlja realizacija Master automata, pažnja biće posvećena njegovoj specifikaciji.

Master automat sadrži izlaz (LED diodu), koja svetli na svake dve sekunde i tako signalizira postojanje komunikacije između Slave i Master automata. On takođe poseduje i LCD displej, na kome se prikazuju vrednosti svih ulaznih i svih izlaznih promenljivih za 16 Slave automata , kao i da li postoji komunikacija između Master automata i određenog Slave automata. Na ulazu Slave automata nalazi se potenciometar, on simulira senzor, koji definiše *spoljno osvetljenje (Sp)*, koje može imati vrednost od 0 (krajnja leva pozicija potenciometra) do 99 (krajnja desna pozicija potenciometra). Master automat Slave automatu šalje *zadato osvetljenje (Os)*. Kao i broj programa *(Pr),* tj. *režim rada*. Na izlazu Slave automata nalaze se: *stanje o brisolejima (Br),* čiji je opseg od 0 (potpuno podignuti brisoleji) do 99 (potpuno spušteni brisoleji) i *jačina sijalice (Si),* čija se vrednost takođe kreće u opsegu od 0 do 99.



Slika 1: Ulazi/izlazi Slave automata

Ovaj sistem posduje pet različitih režima rada:

1. Program0 – *Sunčan dan* - (maksimalno otvoreni brisoleji, svetlo ugašeno ⇔ Br=99, Si=0),
2. Program1 – *Privatnost* – (zatvoreni brisoleji, svetlo srednjeg intenziteta ⇔ Br=0, Si=50),
3. Program2 – *Intima* -(zatvoreni brisoleji, prigušeno svetlo ⇔ Br=0, Si=20),
4. Program3 – *Noć*-(zatvoreni brisoleji, ugašeno svetlo ⇔ Br=0, Si=0) i
5. Program4 – *Programabilni režim* – Na osnovu zadatog osvetljenja (Os) i spoljnog osvetljenja (Sp), a pomoću formule:

$$\boldsymbol{Os = \frac{Sp*Br}{100} + Si,}$$

Izračunati stanje brisoleja (Br), kao i jačinu sijalice (Si). (Napomena: Potrebno je voditi računa da pre uključivanja sijalice, Br treba iznositi 99).

Master automat je programiran tako da se preko njega šalju predefinisane vrednosti Slave automatu. Glavni oblik komunikacije, koji je iskorišćen pri projektovanju sistema je tipa Master-Slave. Master automat periodično komunicira sa Slave automatima koji se nalaze u sobama, pri čemu Master automat zadaje komandu pojedinačnom Slave automatu, a prozvani Slave automat odgovara na pristiglu komandu. Redni broj Slave automata (prostorije) biramo pomoću dva tastera RB0 i RB1, pritiskom na njih menjamo stanje brojača (uvećavamo ili smanjujemo), čime određujemo sa kog Slave automata će se prikazati informacije na LCD displeju Master automata.

2.2. NEFUNKCIONALNA SPECIFIKACIJA

Sistem ne treba da bude posebno izolovan u pogledu spoljašnjih uslova kao što su vlaga i rad na visokim ili izuzetno niskim temperaturama budući da sistem radi u zatvorenom prostoru. Master i Slave automati napajaju se preko elektroenergetske mreže AC napona napajanja 220V i to su uređaji koji imaju malu potrošnju.

3. ARHITEKTURA SISTEMA

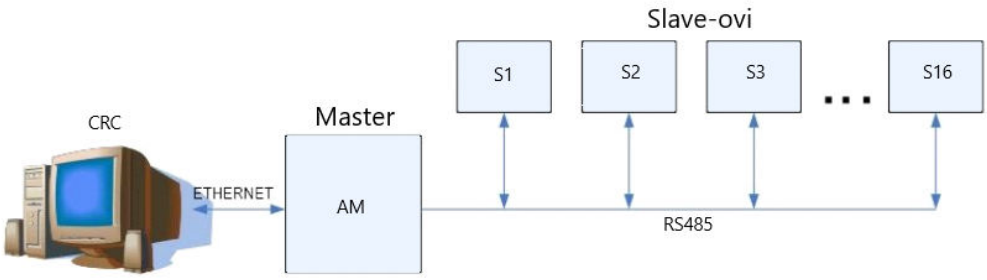
Arhitektura celokupnog sistema sastoji se iz:

1. CRS (Centralni Računarski Sistem)
2. Master automat (AM)
3. Slave automati (S_1-S_{16})

CRS , centralni računarski sistem, služi za očitavanje i prikazivanje korisnih podataka sa Master automata, kao i komunikaciju sa mikrokontrolerima, prikupljanje informacija u prostorijama objekta, kao i promenu predefinisanih postavki za osvetljenost i dodavanje novih. Povezivanje sa Master automatom se vrši preko Ethernet veze.

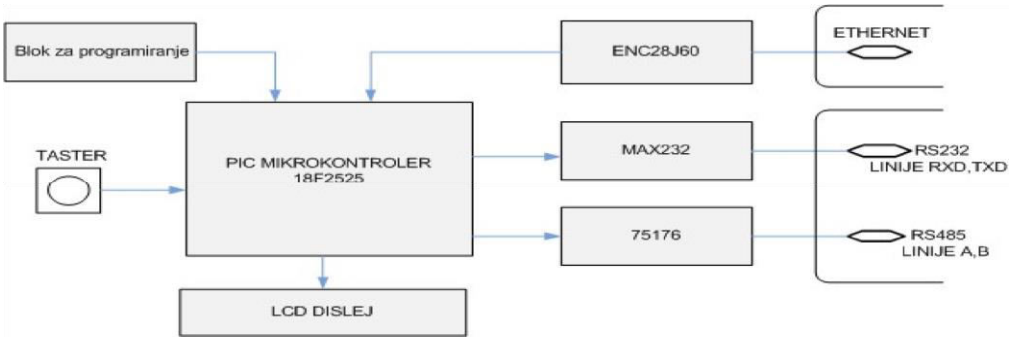
Master automat je povezan sa više Slave automata preko RS-485 veze ili ako je u pitanju samo jedan Slave automat preko RS-232 veze.

Slave automati se nalaze u svakoj prostoriji (16 prostorija). Da li je povezan preko RS-485 ili RS-232 veze sa Maester automatom bira se podešavanjem kratkospojnika na Slave pločici.



Slika 2: Prikaz arhitekture sistema

Osnovna blok šema Master automata prikazana je na slici 3. Rad automata zasnovan je na mikrokontroleru. Ugrađen je Microchip P18F2525. Pored mikrokontrolera Master automat obuhvata blokove za RS232/485 komunikaciju (kola drajvera MAX232 i 75176), blok za Ethernet komunikaciju (kolo drajvera ENC28J60), blok koji služi za programiranje mikrokontrolera i LCD displej, za prikazivanje podataka. Takođe Master automat poseduje i dva tastera, koji služe za menjanje stanja na LCD displeju, ali mi koristimo samo jedan.



Slika 3: Prikaz blok šeme Master automata

Na Master i Slave pločama nalaze se LCD displeji, koji mogu prikazati po šesnaest bitova u dva reda.

Na LCD displeju Master automata prikazuju se vrednosti svih ulaznih i svih izlaznih promenljivih za 16 Slave automata, kao i da li postoji komunikacija između Master automata i određenog Slave automata. Pritiskom na taster menja se stanje LCD displeja.

Na displeju se može prikazati 32 karaktera, koji su raspoređeni u dva reda. U drugom redu nalaze se vrednosti promenljivih koje su navedene u prvom redu. Promenljive na displeju redom:

1. Sl – broj Slave automata, može imati vrednost od 0 do 15
2. Pr – režim rada (broj programa), može imati vrednost od 0 do 4
3. Os – zadato osvetljenje, može imati vrednost od 0 do 99
4. Sp – spoljno osvetljenje, može imati vrednost od 0 do 99
5. Si – jačina sijalice može, imati vrednost od 0 do 99
6. Br – stanje brisolja može, imati vrednost od 0 do 99
7. Ko- promenljiva koja govori da li je ostvarena komunikacija između Master i Slave automata, ukoliko jeste promenljiva ima vrednost 1, u suprotnom slučaju ima vrednost 0

S	L		P	r	O	s	S	p	S	i	B	r		K	O
X_{10}	X_1		X_{10}	X_1	X_{10}	X_1	X_{10}	X_1	X_{10}	X_1	X_{10}	X_1		X_{10}	X_1

Tabela 1: Šematski prikaz LCD displeja

4. REALIZACIJA KOMUNIKACIONOG PROTOKOLA

Svi Slave automati su preko RS485 veze povezani sa Master automatom, dok je sam Master automat preko Etherneta povezan sa centralnim računarskim sistemom.

4.1. ETHERNET KOMUNIKACIJA

Na samom PC-u, u Web Browser-u (npr. Google Chrome), u komandnoj liniji navodimo IP adresu Master automata. Nakon IP adrese unosimo neku poruku, formata:

IP/pXXYZZ,

gde je XX redni broj Slave automata, kojih ukupno ima 16, sa oznakama od 0-15, zatim Y predstavlja redni broj programa (režima rada) može iznositi od 0-4 i na posletku ZZ predstavlja zadatu osvetljenost (Os), čiji je opseg vrednosti 0-99.

Ethernet komunikacija može biti UDP ili TCP. Obe moraju biti implementirane u kodu Master automata, i pored toga što se UDP ne koristi, ona mora biti navedena kako ne bi došlo do greške.

4.2. KOMUNIKACIJA IZMEĐU MASTER I SLAVE AUTOMATA

Projektovanje komunikacije između Master i Slave automata je jedan je od najbitnijih zadataka pri projektovanju sistema. Uređaji poseduju drajvere za RS232 i RS485 komunikaciju. Za RS232 koristi se integrisano kolo MAX232, dok se za RS485 koristi kolo SN65176. Kako su Master i Slave realizovani upotrebom mikrokontrolera, koriste se i UART primopredajnici koji su integrisani u njima. Komunikacija između uređaja je paketnog tipa – na nivou bajtova. Nadalje će biti objašnjen princip komunikacije i dat detaljan prikaz poruka koje se razmenjuju.

Master automat svakom Slave automatu prosleđije isti paket koji sadrži 3 bajta (24 bita). Prva 4 bita 1010 predstavljaju komandne bitove, naredna 4 predstavljaju broj prostorije, odnosno Slave automata, (ID) može imati vrednost od 0 do 15 (0000-1111), zatim sledi bajt koji predstavlja program, odnosno režim rada (mode0 - mode4), on ima opseg od 0 do 4 (00000000-00000100) i poslednji, treći, bajt predstavlja zadatu osvetljenost (Os) koja, kao što je već rečeno, može imati vrednosti od 0 do 99 (00000000-01100011).

Komandni bajt	Režim rada (program)	Zadata osvetljenost
1010 (ID 4b (0000-1111))	(00000000-00000100)	(00000000-01100011)

Tabela 2: Prikaz paketa koji Master automat prosleđuje Slave automatu

Master automat od svakog Slave automata očekuje odgovor u vidu paketa, koji sadrži 5B (40b). Prva 4 bita 1010 predstavljaju kontrolne bitove, naredna 4 predstavljaju broj prostorije, odnosno Slave automata, (ID) može imati vrednost od 0 do 15 (0000-1111), sledeći bajt predstavlja stanje o brisolejima (Br), koje ima opseg od 0 do 99 (00000000-01100011), zatim sledi bajt koji predstavlja jačinu sijalice (Si), koja takođe može imati vrednost od 0 do 99 (00000000-01100011), četvrti bajt predstavlja zadatu osvetljenost (Os) koja, kao što je već rečeno, može imati vrednosti od 0 do 99 (00000000-01100011) i poslednji, peti, bajt predstavlja spoljno osvetljenje (Sp), koje takođe može imati vrednost od 0 do 99 (00000000-01100011).

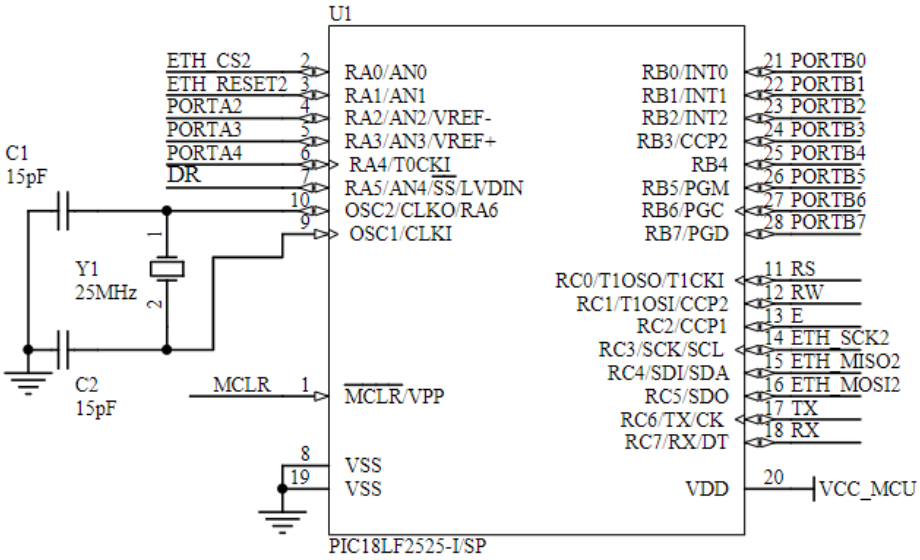
Kontrolni bajt	Br	Si	Os	Sp
1010 (ID 4b (0000-1111))	(00000000-01100011)	(00000000-01100011)	(00000000-01100011)	(00000000-01100011).

Tabela 3: Prikaz paketa koji Slave automati prosleđuju Master automatu

5. HARDVERSKA REALIZACIJA AUTOMATA

Električna šema sadrži mikrokontroler PIC18F2525, 16x2 LCD displej, drajvere za RS232 i RS485, blok za napajanje, blok za programiranje mikrokontrolera, taster koji se koristi za promenu prikaza LCD displeja, LED diodu koja se koristi za signalizaciju postojanja komunikacije između Slave i Master automata.

Srce sistema je mikrokontroler PIC18F2525 koji ima SPDIP kućište i radi na 25MHz i naponima napajanja od 5V. Osnovne karakteristike su sledeće: mikrokontroler ima 48KB ROM memorije, 3986B RAM memorije, I2C i SPI komunikacioni interfejs, 10-bitni A/D konvertor sa do 13 kanala. Postoje tri porta sa digitalnim ulazima i izlazima koji su obeleženi redom A, B i C. Na Master ploči se nalazi i bitna komponenta ENC28J60. Ova komponenta predstavlja Ethernet drajver u SOIC–28 pakovanju, radi na 25MHz i napajanju od 3,3V, međutim, uređaj poseduje još jedan blok koji omogućava toleranciju do 5V za neke ulaze i izlaze. Šematski prikaz hardverske realizacije mikrokontrolera PIC18LF2525 prikazan je na slici 4.



Slika 4: Šematski prikaz hardverske realizacije Master automata

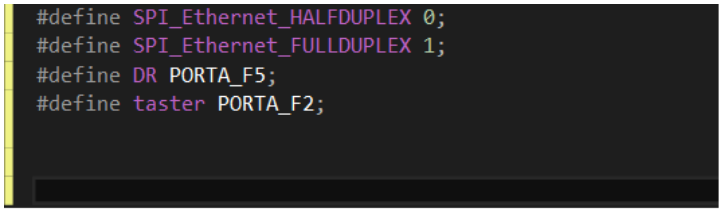
U tabeli 4 navedeni su i opisani korišćeni ulazni i izlazni pinovi mikrokontrolera PIC18LF2525.

Port	Smer	Opis
RB0	Ulaz	UP taster – bira se adresa Slave-a, inkrementiranjem
RB1	Ulaz	Down taster – bira se adresa Slave-a, dekrementiranjem
RC0	Izlaz	RS- Data/Instruction select – kontrolni signal LCD displeja
RC2	Izlaz	E (Enable) – kontrolni signal LCD displeja
RA5	Izlaz	DR – Write/Read – kontrola RS485 drajvera SN65176; ako je DR='1' kolo je u Write modu, u suprotnom je u Read modu
RB4-RB7	Izlaz	Magistrala podataka za prenos informacija između mikrokontrolera i LCD displeja

Tabela 4: Ulazni i izlazni pinovi mikrokontrolera PIC18LF2525

6. REALIZACIJA FIRMVERA

Kod Master automata realizovan je u C programskom jeziku. Pritom je korišćen kompajler MikroC PRO firme Mikroelektronika. Upotrebljeni PIC mikrokontroler je P18F2525 koji radi na frekvenciji od 25MHz. Definicija pinova data je na slici 5. Svim promenljivama potrebno je dodeliti vrednost "0" pri inicijalizaciji. To je opisano u funkciji *init_variables()* koja se poziva u glavnom *main()* programu i to na samom početku.



Slika5 : Prikaz definicije pinova

U tabelama 5 i 6 nabrojane su i opisane najznačajnije promenljive, kao i funkcije korišćene u kodu.

Naziv promenljive	Tip	Opis
cntTas	unsigned char	Kada ima vrednost: 0 radi, pritiskom na njega; 20 –period blockade. Služi za eliminaciju treptaja
cntDisp	unsigned char	Koristi se za promeu sadržaja na displeju
Flag1	Bit	Govori da je došlo vreme da se prozove soba
OBB	unsigned char	Očekivani broj bajtoa, koji prima Master automat
Ch	unsigned char	Pomoćna promenljiva u kojoj se smešta primljeni bajt
Brojac	unsigned char	Brojač koji se koristi u funkciji *Interrupt()*
PROGRAM	unsigned char	Promenljiva u kojoj se smešta informacija o režimu rada (0-4)
ROOM_ID	unsigned char	Promenljiva u kojoj se smešta redni broj prostorije (0-15)
BRISOLEJ	unsigned char	Promenljiva u kojoj se smešta vrednost stanja brisoleja (0-99)
SIJALICE	unsigned char	Promenljiva u kojoj se smešta vrednost jačine sijalice (0-99)
OSVETLJENOST_SOBE	unsigned char	Promenljiva u kojoj se smešta vrednost zadate osvetljenosti (0-99)
SPOLJNA_OSVETLJENOST	unsigned char	Promenljiva u kojoj se smešta vrednost spoljne osvetljenosti (0-99)

Tabela 5: Najznačajnije promenljive u programskom kodu

Naziv funkcije	Tip	Opis
init_variables()	Void	Inicijalizacija početnih vrednosti
init()	Void	Podešava rad pinova mikrokontrolera, konfiguriše periferije kontrolera, AD konvertor, prekide, kola tajmera; inicijalizuje UART
DodajUNiz(char *p_ch)	Void	Dodaje elemente u niz
FormirajNiz()	Void	Formira se niz čije elemente čine pojedinačni odgovori svih šesnaest Slave automata
UpdateLCD()	Void	Definiše prikaz sadržaja na LCD displeju
interrupt()	Void	Obrada prekida mikrokontrolera
main()	Void	Glavna funkcija programskog koda u kojoj se izvršavaju druge navedene funkcije
transmit()	void	Obavlja slanje bajtova preko kola UART-a. Kao ulazni parameter funkcija uzima 8-bitni podatak
SPI_Ethernet_UserTCP(unsigned char *remoteHost, unsigned int remotePort, unsigned int localPort, unsigned int reqLength, char *canClose)	unsigned int	Obrađuju se zahtevi koji pristižu iz Web Browser-a i izvršava se slanje podataka preko Etherneta

Tabela 6: Funkcije korišćene u kodu

7. DETALJAN OPIS NAJVAŽNIJIH FUNKCIJA SA SEGMENTIMA KODA

7.1. FUNKCIJA init()

Što se tiče same funkcije *init(),* u njoj se vrši niz podešavanja vezanih za mikrokontroler, pa tako imamo podešavanja rada pinova mikrokontrolera, konfigurišu se periferije kontrolera, prekidi, kola tajmera, inicijalizuje se *UART* (Universal Asynchronous Receiver-Transmitter) komunikacioni modul itd. Zbog svega toga funkcija *init()* mora biti izvršena na samom početku rada mikrokontrolera, pre svih drugih funkcija. Kod ove funkcije prikazan je na slikama 6 i 7.

```
void init()
{

    PIR1 = 0b00000000; // flegovi prijema preko UART-a
    PIE1 = 0b00100001; // dozvola prekida za UART, RCIE, TMR1IE
    //PIE1.TMR1IE = 1;
    //PIE1.RC1IE=1;

    T1CON = 0b10110000; // konfiguracija za tajmer1
    T1CON.TMR1ON = 1;
    // 16-bit operation
    // preskaler 1:8
    // 25MH T0=40ns
    // 40ns*4*8=1.28us
    // 25ms=25000us=1.28*19531=  B5B3
    TMR1L = 0xB5;
    TMR1H = 0xB3;

    INTCON = 0b01000000; // periferijski interapt
    INTCON.GIE = 1;      // globalna dozvola prekida

    // svi pinovi na portu A su izlazi
    TRISA = 0x00;
    TRISB = 0x0F; // prekidaci na portovima od RB0 do RB3 su digitalni ulazi
    TRISC = 0xD0; // 0b11010000

    PORTA = 0x00;
    PORTB = 0x00;
    PORTC = 0x00;
     //ukljucujemo A/D konverziju
    // ADCON1 = 0x4E;
```

Slika 6 : Prikaz f-je *init()* u "*Microsoft Visual Studio"* razvojnom okruženju (1. deo)

```
    PORTA = 0x00;
    PORTB = 0x00;
    PORTC = 0x00;
     //ukljucujemo A/D konverziju
    // ADCON1 = 0x4E;

    ADCON0 = 0x00; // iskljucujemo A/D konverziju
    ADCON1 = 0x0F; // svi digitalni

    //PORTA.F0=0;

    UART1_Init(19200);
    // konfigurisemo brzinu od 19200 i dodatne bitove za serijski prenos
    TXSTA.TXEN = 1;
    RCSTA.SPEN = 1;
    RCSTA.CREN = 1;

    Lcd_Init();
    Lcd_Cmd(_LCD_CURSOR_OFF);

    UpdateLCD();

    SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV64, _SPI_DATA_SAMPLE_MIDDLE, _SPI_CLK_IDLE_LOW, _SPI_LOW_2_HIGH);
    SPI_Ethernet_Init(myMacAddr, myIpAddr, SPI_Ethernet_FULLDUPLEX); // inicijalizujemo Ethernet port
}
```

Slika 7: Prikaz f-je *init()* u "*Microsoft Visual Studio"* razvojnom okruženju (2. deo)

U nastavku sledi opis koda koji je prikazan na slikama iznad.

Na početku funkcije postavljaju se vrednosti *PIR* (Peripheral Interrupt Request) i *PIE* (Peripheral Interrupt Enable). *PIR1* sadrži isključivo flegove za periferijske portove na čipu koji mogu da generišu prekid. Flegovi prekida u registru *PIR1* se uvek setuju/resetuju kada je uslov za setovanje/resetovanje flegova ispunjen. Do prekida ne dolazi ukoliko odgovarajući bit u registru *PIE1* nije setovan na jedinicu (ukoliko mu nije odobren prekid). *PIE1* sadrži kontrolne bitove koji omogućavaju ili sprečavaju svaki od periferijskih prekida.

PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
0	0	1	0	0	0	0	1

Tabela 7: Prikaz strukture registra PIE1

Bitovi *PIE1.RCIE* i *PIE1.TMR1IE* imaju vrednost 1. Što, redom, podrazumeva da će mikrokontroler dozvoliti prekid koji postavlja *UART* (naime svaki put kada se primi neki karakter preko *UART* komunikacije, generisaće se prekid) i da će mikrokontroler dozvoliti prekid tajmera *Timer1*, što podrazumeva da će doći do prekida tajmera1.

Registri *T1CON, TMR1L i TMR1H* služe za podešavanje tajmera *Timer1*. Bitovi *T1CON* registra kontrolišu rad *Timer1* modula.

RD16	T1RUN	T1CKPS1	T1CPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
1	0	1	1	0	0	0	0

Tabela 8: Prikaz strukture registra T1CON

Bitovi *T1CON.RD16, T1CON.T1CKPS1, T1CON.T1CKPS0* imaju vrednost 1. Što, redom, podrazumeva da je u programu korišćen šesnaesto-bitni tajmer, naredna dva navedena bita služe za postavljanje vrednosti preskalera koji u slučaju da oba bita budu setovana ima vrednost 1:8. Izlaz preskaler bloka generiše taktni signal koji inkrementira šesnaesto-bitni brojač. Zatim se i bit *T1CON.TMR1ON* takođe postavlja na vrednost 1, čime omogućavamo rad tajmera.

Tajmer/brojač sastoji se iz dva 8-bitna registra *TMR1H* i *TMR1L*. Ovim registrima dodeljene su startne vrednosti, na način koji je opisan u nastavku. Poznato je da frekvencija oscilatora (f_{osc}) za mikrokontroler *PIC18LF2525* iznosi 25MHz, a samim tim njegova perioda $T_{osc}=40ns$, kao i vrednost preskalera koja iznosi 1:8. Takođe je poznato da na ulaz za takt bloka preskalera dolazi taktni signal (interni taktni signal) čija je frekvencija f_{int} četiri puta manja od osnovne frekvencije oscilatora f_{osc}=25MHz. Na osnovu navedenog može se izračunati perioda internog takta: $T_{int}=40ns*8*4=1,28\mu s$.

Kako je cilj generisati prekid tajmera na svakih 25ms, na osnovu dobijenog internog takta može se izračunati koliko puta će brojač odbrojiti do prekoračenja tako što ćemo podeliti željenih 25ms sa peridom internog takta, ovom računicom dobija se broj 19531, koji u heksadecimalnom zapisu iznosi 4C4B, kada ovu vrednost oduzmemo od 10000, heksadecimalno, dobijamo vrednost B3B5, koja predstavlja startnu vresdnost 16-bitnog brojača. Iz prethodno navedenog sledi da će se registru *TMR1H* dodeliti vrednost 0xB3, a registru *TMR1L* vrednost 0xB5.

Registar *INCON* sadrži različite flag, enable i priority bitove.

GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TIMR0IF	INT0IF	RBIF
0	1	0	0	0	0	0	0

Tabela 9: Prikaz strukture registra *INTCON*

Bit *INTCON.PEIE* postavljen je na 1, čime se dozvoljavaju svi periferijski prekidi, zatim i bit *INTCON.GIE* postavljamo na 1, čime se dozvoljavaju sve globalne prekide.

TRIS registrom se određuje smer *PORT* pinova, dok *PORT* registar omogućava čitanje informacija sa *I/O* pinova. Svaki bit setovan u *TRIS* registru će definisati odgovarajući *PORT* pin kao ulaz, a resetovan kao izlaz. U našem slučaju svi pinovi na portu A su izlazi, dok su prekidači na portovima od *RB0* do *RB3* digitalni ulazi.

Modul A/D konverzije omogućava konverziju nekog analognog ulaza u 10-bitni digitalni broj. Registrima *ADCON0* i *ADCON1* podešavamo ovaj modul, ovi registri, redom, imaju vrednosti 0x00 i 0x0F, čime se isključuje A/D konverzija i svi ulazi postaju digitalni.

UART1_Init(19200) je funkcija iz biblioteke mikroC kompajlera koja služi za podešavanje *UART* komunikacije. Njome incijalizujemo BAUD-rate. U kodu primećujemo da je postavljena na vrednost *19200 [UART1_Init(19200)]* čime je podešen prenos podataka na 19200 bitova po sekundi.

Za uspostavljanje komunikacije sa displejem pored podešenih pinova neophodno je pozvati funkcije *LCD_Init i LCD_Cmd*. Za inicijalizaciju LCD modula koristimo funkciju *LCD_Init* dok za slanje komandi LCD modulu koristimo funkciju *LCD_Cmd*.

SPI1_Init_Advanced i SPI_Ethernet_Init funkcije služe za uspostavljanje SPI komunikacije. Funkcija *SPI1_Init_Advanced* inicijalizuje konekciju sa Ethernet kontrolerom dok funkcija *SPI_Ethernet_Init* incijalizuje sam Ethernet kontroler.

7.2. FUNKCIJA interrupt()

Funkcija *interrupt()* se koristi za obradu prekida mikrokontrolera. Ona je tipa *void*, što znači da ne vraća nijedan parameter. U funkciji su obrađeni prekidi tajmera 1 i serijskog prijema preko UART-a. Do prekida tajmera dolazi na svakih 25ms.

Na slikama 8, 9 i 10 prikazan je kod funkcije *interrupt()* u "*Microsoft Visual Studio"* razvojnom okruženju.

```
void interrupt()
{

    unsigned char ch = 0;

    if ((PIE1.TMR1IE == 1) && (PIR1.TMR1IF == 1))
    {
        // prekid tajmera 1 -  na svakih 25ms
        PIE1.TMR1IE = 1;
        PIR1.TMR1IF = 0;

        if (brojac >= 4)
        { // na svakih 125ms proziva se po jedna soba od 16
            brojac = 0;
            Flag1 = 1; // podize se flag koji koristimo ga u Main funkciji
        }
        else
            brojac++;

        if (cntTas > 0)
            cntTas--;

        if ((PORTB.F0 == 1) && (cntTas == 0)) // taster UP
        {
            if (cntDisp == 15)
                cntDisp = 0; // za promenu sadrzaja displeja na masteru
            else
                cntDisp++;
            cntTas = 20;
        }

        if ((PORTB.F1 == 1) && (cntTas == 0)) // taster DOWN
        {
            if (cntDisp == 0)
                cntDisp = 15;
```

Slika 8 : Prikaz f-je *interrupt()* u "*Microsoft Visual Studio"* razvojnom okruženju (1. Deo)

```
            else
                cntDisp--;
            cntTas = 20;
        }

        TMR1L = 0xB5;
        TMR1H = 0xB3;
    }

    if ((PIE1.RCIE == 1) && (PIR1.RCIF == 1))
    { // prekid zbog serijske komunikacije

        PIR1.RCIF = 0;
        ch = RCREG; //prima se bajt preko UART-a

        if (OBB == 5)
        {
            // prima kom. bajt
            if (((ch & 0x0F) == ROOM_ID) && ((ch & 0xF0) == 0xA0))
                OBB = 4;
        }
        else if (OBB == 4)
        {
            if (ch > 99)
                ch = 99;

            BRISOLEJ[ROOM_ID] = ch; // prima brisolej
            OBB--;
```

Slika 9: Prikaz f-je *interrupt()* u "*Microsoft Visual Studio"* razvojnom okruženju (2. Deo)

```
                OBB = 4;
        }
        else if (OBB == 4)
        {
            if (ch > 99)
                ch = 99;

            BRISOLEJ[ROOM_ID] = ch; // prima brisolej
            OBB--;
        }
        else if (OBB == 3)
        {
            if (ch > 99)
                ch = 99;

            SIJALICE[ROOM_ID] = ch; // prima sijalice
            OBB--;
        }
        else if (OBB == 2)
        {
            OSVETLJENOST_SOBE[ROOM_ID] = ch; // osvetljenost PRIMAMO OD ETHERNETA
            OBB--;
        }
        else if (OBB == 1)
        {
            if (ch > 99)
                ch = 99;

            SPOLJNA_OSVETLJENOST[ROOM_ID] = ch; // prima napolje
            OBB = 0;
        }
    }
}
```

Slika 10: Prikaz f-je *interrupt()* u "*Microsoft Visual Studio"* razvojnom okruženju (3. Deo)

Prvo se ispituje uzrok prekida, do prekida može doći ili zbog tajmera 1 ili zbog serijske komunikacije, tako što se ispituju bitovi dozvole *PIE1.TMR1IE* i *PIR1.TMR1IF*. Ako i *PIE1.TMR1IE* i *PIR1.TMR1IF* imaju vrednost 1, došlo je do prekida tajmera. Tada se resetuje fleg *PIR1.TMR1IF*, kako bi mogao biti detektovan naredni prekid tajmera 1.

Zatim ispitujemo da li je *brojac>=0x04*, što znači da se na svakih 125ms proziva jedan Slave automat. Ako je uslov ispunjen, *brojac* postavljamo na nulu i podiže se flag (*Flag1=1*) koji govori da je došlo vreme da se prozove Slave. Ukoliko uslov nije ispunjen *brojac* se inkrementira za 1.

Prekid tajmera 1 smo još iskoristili za taster *PORTB.F0* kako bi došlo do promene sadržaja prikazanog na displeju. Za svaku sobu na istom displeju se prikazuju određeni podaci. Zato je potreban jedan taster preko koga se bira broj sobe (*PORTB.F0*). Ako je *PORTB.F0* na 1 dolazi do promene promenljive *cntDisp* koja određuje čije će se informacije prikazivati. Zatim se *cntDisp* inkrementira (ako je došlo do 15 kreće se od 0). Promenljiva *cntTas* se koristi za eliminaciju treperenja tastera. Kada je *cntTas* na 0 tada je dozvoljen pritisak, a kada se pritisne dobija vrednost 20, i to je period blokade (250ms je zabranjen pritisak tastera , tj on ne reaguje). Na kraju funkcije za prekid tajmera ponovo se dodeljuju startne vrednosti za registre *TMR1L* i *TMR1H*, na isti način kao što je to određeno u funkciji *init()*. Ukoliko bi se ovo izostavilo tajmer ne bi radio ispravno.

Zatim sledi ispitivanje serijskog prijema. Master automat prima pet bajtova: komandni bajt (1010ID), stanje o brisolejima (Br), jačinu sijalice (Si), zadato osvetljenje (Os) i spoljno osvetljenje (Sp).
Naime ispituje se da li flegovi *PIE1.RCIE* i *PIR1.RCIF* imaju vrednost jedan, ukoliko su ovi uslovi ispunjeni znači da je došlo do prekida zbog serijske komunikacije zato resetujemo fleg *PIR1.RCIF* kako bismo oslobodili bafer za naredni bajt. Bajt primljen preko UART-a u specijalnom funkcijskom registru *RCREG* dodeljujemo pomoćnoj promenljivoj *ch*. Zatim ispitujemo promenljivu *OBB* (Očekivani Broj Bajtova). Ako je *OBB =5* primamo komandni bajt, *OBB=4* primamo stanje o brisolejima, *OBB=3* jačinu sijalice, *OBB=2* zadato osvetljenje ili ako je *OBB=1* primamo spoljno osvetljenje. Kada je 5 znamo da primamo komadni bajt, moramo videti šta je u nižim bit pozicijama, da li je prozvan ID broj nekog Slave automata i šta je u višim bit pozicijama, da li je kombinacija 1010. Maskiranjem sa 0x0F izdvajaju se poslednja 4 bita i ispituje da li ta 4 bita odgovaraju ID broju prozvane sobe, ako je to ispunjeno i ako je ispunjeno da su prva 4 bita kombinacija 1010 (0xF0 je maska za prva 4 bita) onda smo primili komandni bajt i prelazimo na naredni. Kada naredni put primamo novi bajt, funkcija *interrupt()* kreće ispočetka, ponovo proveravamo *if* naredbu i kada dođemo do *else if (OBB = = 4)* znamo da primamo drugi bajt po redu, tj. stanje o brisolejima. U ovoj, ali i u svakoj narednoj *else if* petlji prvo se ispituje ispravnost dospelog bajta pomoću *if* petlji, čiji uslovi proveravaju da li se dospela informacija nalazi u dozvoljenom opsegu (npr. *if(ch<99)*). Zatim je potrebno dobijenu informaciju sačuvati, tako što ćemo je smestiti u odgovarajući niz na osnovu vrednosti *OBB-a*. U konkretnom slučaju postoji četiri niza: *BRISOLEJ[], SIJALICE[], OSVETLJENOST_SOBE[]* i *SPOLJNA_OSVETLJENOST[];* Svaki od njih sadrži šesnaest članova koji odgovaraju po jednom Slave automatu (indeks člana niza podudara se sa brojem Slave automata), kada je pristigli bajt sačuvan, sledi dekrementiranje promenljive *OBB*, nakon čega se postupak ponavlja za njenu sledeću vrednost. Na ovaj način se popunjavaju gore navedeni nizovi.

7.3. FUNKCIJA UpdateLCD()

Nakon funkcija koje smo obradili do sada, kod nas vodi do funkcije koja služi za ispisivanje stanja senzora na LCD displeju, a ona se naziva *UpdateLCD(),* tipa *void.* Kao što je već ranije rečeno, na LCD displeju Master automata prikazuju se vrednosti svih ulaznih i izlaznih promenljivih za 16 Slave automata, kao i da li je ostvarena komunikacija između Master automata i odgovarajućeg Slave automata, što ukupno predstavlja sedam različitih informacija. Ove informacije i njihove vrednosti se redom ispisuju na displeju i to u dva reda od po šesnaest karaktera. U prvom redu se ispisuju nazivi promenljivih, a u drugom redu njihove vrednosti. U nastavku je opisan način na koji funkcija *UpdateLCD()* ispisuje stanja senzora na LCD displeju.

Primećujemo da se u kodu ove funkcije, koji je prikazan na slici 11, javljaju funkcije *ConvertValue(), LCD_Out()* i *LCD_Chr().* Argument funkcije *ConvertValue()* je promenljiva u kojoj je smeštena vrednost same informacije. Njen zadatak je da ovu vrednst raščlani na cifre, kako bi bilo moguće prikazivanje iste na displeju (npr. ukoliko promenljiva ima vrednost 53, navedena funkcija će njenu cifru desetica smestiti u globalnu promenljivu *X10*, a cifru jedinica u globalnu promenljivu *X1*, što znači da će ove dve globalne promenljve redom dobiti vrednosti 5 i 3). Funkcija *LCD_Out()* ispisuje string, naziv promenljive, u prvom redu displeja. Ona ima tri argumenta, prva dva argument su tipa *int* i oni, redom, predstavljau redni broj reda i kolone, pozicije na displeju od koje će krenuti ispisivanje stringa koji je ujedno treći argument ove funkcije. Funkcija *LCD_Chr()* ispisuje vrednosti informacija cifru po cifru. Kako LCD displej radi u skaldu sa ASCI kodom, potrebno je ove vrednosti konvertovati u isti (*X10+0x30*). Kao i prethodna i ova funkcija poseduje tri argumenta koji, redom, predstavljaju redni broj reda, kolone i jednu od cifara vrednosti informacije uvećanu za 0x30.

```
void UpdateLCD()
{

    //koja je soba pozvana na LCD
    LCD_Out(1, 1, "Sl"); // redni broj sobe/slejva
    ConvertValue(cntDisp);
    Lcd_Chr(2, 1, X10 + 0x30);
    Lcd_Chr(2, 2, X1 + 0x30);

    Lcd_Out(1, 4, "Pr"); // program rada
    ConvertValue(PROGRAM[cntDisp]);
    Lcd_Chr(2, 4, X10 + 0x30);
    Lcd_Chr(2, 5, X1 + 0x30);

    Lcd_Out(1, 6, "Os"); // osvetljenost sobe (zadata)
    ConvertValue(OSVETLJENOST_SOBE[cntDisp]);
    Lcd_Chr(2, 6, X10 + 0x30);
    Lcd_Chr(2, 7, X1 + 0x30);

    Lcd_Out(1, 8, "Sp"); // spoljna osvetljenost (izmerena)
    ConvertValue(SPOLJNA_OSVETLJENOST[cntDisp]);
    Lcd_Chr(2, 8, X10 + 0x30);
    Lcd_Chr(2, 9, X1 + 0x30);

    Lcd_Out(1, 10, "Si"); // sijalice (izlaz)
    ConvertValue(SIJALICE[cntDisp]);
    Lcd_Chr(2, 10, X10 + 0x30);
    Lcd_Chr(2, 11, X1 + 0x30);

    LCD_Out(1, 12, "Br"); // brisoleji (izlaz)
    ConvertValue(BRISOLEJ[cntDisp]);
    Lcd_Chr(2, 12, X10 + 0x30);
    Lcd_Chr(2, 13, X1 + 0x30);

    LCD_Out(1, 15, "Ko"); // komunikacija
    Lcd_Chr(2, 15, KOMUNIKACIJA[cntDisp] + 0x30);
}
```

Slika 11: Prikaz f-je *UpdateLCD()* u "*Microsoft Visual Studio"* razvojnom okruženju

7.4. FUNKCIJA main()

Glavna funkcija svakog programa pri projektovanju rada mikrokontrolera je funkcija *main(),* u okviru koje pozivamo prethodno navedene i definisane funkcije *init_variables(), UpdateLCD(),* i *init().* Funkcija *main()* prikazana je na slikama 12 i 13.

```
void main()
{
    unsigned char ByteX = 0x00;
    init();
    init_variables();
    UpdateLCD();

    while (1)
    {

        SPI_Ethernet_doPacket();
        if (Flag1 == 1)
        {
            //na svakih 125ms
                Flag1 = 0;

            if (OBB > 0) /// u ovom trenutku OBB treba da bude 0 ako imamo komunikaciju sa prethodno prozvanim slejvom
                KOMUNIKACIJA[ROOM_ID] = 0;
            else
                KOMUNIKACIJA[ROOM_ID] = 1;

            ROOM_ID++; //biramo sledeci slejv

            if (ROOM_ID == 16)
            {
                //kada se prozovu svih 16 prostorija onda se brojac prostorija vrati na 0
                    ROOM_ID = 0;
                UpdateLCD();
            }
```

Slika 12: Prikaz f-je *main()* u "*Microsoft Visual Studio"* razvojnom okruženju (1. deo)

```
            DR = 1; // saljemo 3 bajta
            ByteX = 0xA0 + ROOM_ID;
            transmit(ByteX);
            ByteX = (PROGRAM[ROOM_ID]); // do 4
            transmit(ByteX);
            ByteX = OSVETLJENOST[ROOM_ID]; // do 99
            transmit(ByteX);
            DR = 0; // sada primamo bajtove
            OBB = 5; // ocekujemo da primimo 5 bajtova od prozvanog slejva

        } //od Flag1
    }
     //od while
} // od main
```

Slika 13 : Prikaz f-je *main()* u "*Microsoft Visual Studio"* razvojnom okruženju (2.deo)

U beskonačnoj *while* petlji najpre je pozvana funkcija *SPI_Ethernet_doPacket()* koja proverava komunikaciju. Pozvana funkcija služi za ostvarivanje Ethernet komnikacije (slanje/prijem podataka). Zatim se proveravaju flegovi, *Flag1* je vezan za tajmer, on se podiže svaki put kada tajmer odbroji (*Flag1=0x01*), što je na svakih 125ms, zatim se resetuje (*Flag1=0*). Nakon čega se bira sledeći Slave automat, kada brojač prozove svih šesnaest Slave automata (prostorija) vraća se na nulu. Nakon 2s kada su prozvane sve prostorije ispisuje se sadržaj na displeju. U promenljivoj *ROOM_ID* se nalazi redni broj Slave automata, kog treba prozvati. Promenljiva *DR* dobija vrednost 1, što znači da RS485 kontroler prelazi u stanje slanja, na taj način šaljemo Slave automatu 3 bajta koji, redom, sadrže sledeće informacije: broj prostorije, broj programa i zadatu osvetljenost. Kada su podaci poslati, kontroler se vraća u stanje prijema (*DR=0*), kako se očekuje prijem pet bajtova promenljivu OBB postavljamo na očekivanu vrednost (*OBB=5*). Ovi bajtovi, redom, nose informacije o rednom broju Slave automata, režimu rada, vrednostima zadatog osvetljenja, spoljnjeg osvetljenja, jačine sijalice i stanja brisoleja.

7.5. FUNKCIJA FormirajNiz()

Funkcja *FormirajNiz()* formira niz karaktera, string, za svaki od Slave automata sa informacijama o stanjima svih senzora, koji ćemo poslati preko Etherneta centralnm računarskom sistemu. Kod ove funkcije prikazan je na slikama 14 i 15.

Po ulasku u for petlju najpre ispitujemo ispravnost komunikacije između Master i Slave automata, ukoliko je ispravna počinjemo sa popunjavanjem stringa vrednostima svih ulaznih i izlaznih promenljivih tekućeg Slave automata. Ove informacije je najpre potrebno konvertovati u string funkcijom *ByteToStr()*, a zatim ih dodati u niz funkcijom *dodajNiz()*. Po izlasku iz for petlje na kraj stringa postavljamo *NULL* karakter i inkrementiramo broj karaktera u stringu za jedan. Kroz for petlju prolazimo onoliko puta koliko imamo Slave automata, u našem slučaju šesnaest. Na ovaj način se formira niz sa sledećim informacijama:

broj sobe, zadato osvetljenje, spoljno osvetljenje, stanje o brisolejima, jačina sijalice i režim rada sistema; za svih šesnaest slave automata, pod uslovom da je veza između Master i Slave automata ostvarena.

```
void FormirajNiz()
{
    int i;
    char text[4];

    br_ch = 0; // na pocetku niz treba da ima 0 karaktera
    // posle ga ponimo pozivajuci dodajUNiz()

    for (i = 0; i < 16; i++)
    {
        if (KOMUNIKACIJA[i] == 1) //imamo komunikaciju sa i-tim slejvom
        {
            dodajUNiz("Soba ID:");
            ByteToStr(i, text);
            dodajUNiz(text);

            dodajUNiz(" Osv.:");
            ByteToStr(OSVETLJENOST_SOBE[i], text);
            dodajUNiz(text);

            dodajUNiz(" Nap.:");
            ByteToStr(SPOLJNA_OSVETLJENOST[i], text);
            dodajUNiz(text);
```

Slika 14: Prikaz f-je *FormirajNiz()* u "*Microsoft Visual Studio"* razvojnom okruženju (1. deo)

```
            dodajUNiz(" Nap.:");
            ByteToStr(SPOLJNA_OSVETLJENOST[i], text);
            dodajUNiz(text);

            dodajUNiz(" Bris.:");
            ByteToStr(BRISOLEJ[i], text);
            dodajUNiz(text);

            dodajUNiz(" Sij.:");
            ByteToStr(SIJALICE[i], text);
            dodajUNiz(text);

            dodajUNiz(" Prog.:");
            ByteToStr(PROGRAM[i], text);
            dodajUNiz(text);

            dodajUNiz("\n\n");
        }
    }
    niz[br_ch] = 0x00; // string se zavrsava NULL karakterom
    br_ch++;
}
```

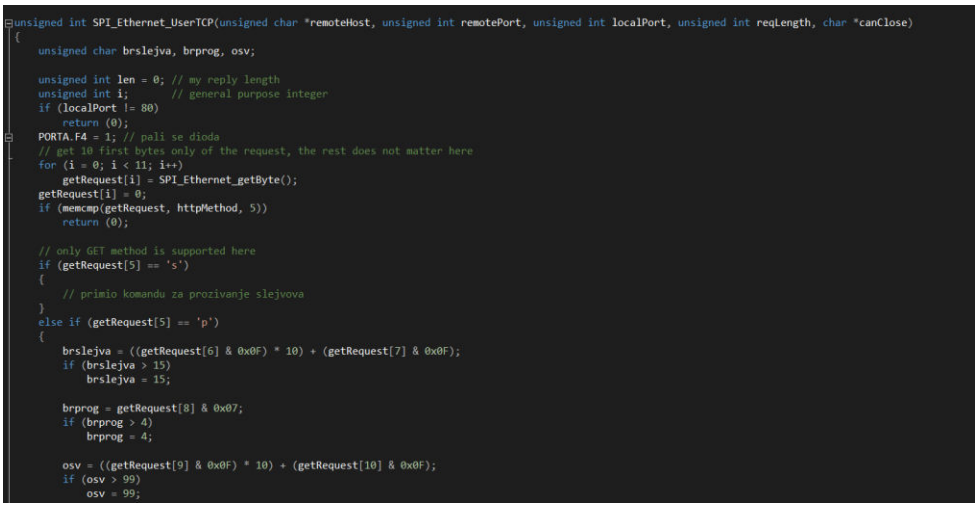
Slika 15: Prikaz f-je *FormirajNiz()* u "*Microsoft Visual Studio"* razvojnom okruženju (2. deo)

7.6. FUNKCIJA SPI_Ethernet_UserTCP()

Funkcija *SPI_Ethernet_UserTCP()* služi da nakon navođenja IP adrese Master automata kao i odgovarajuće poruke, na način koji je opisan u podnaslovu 4.1 Ethernet komunikacija, u komandnoj liniji Web Browser-a obradi zahteve koji stižu iz Web Browser-a i izvrši slanje preko Ethernet-a. Kod ove funkcije prikazan je na slikama 16 i 17.

For petljom, koja se javlja na početku koda ove funkcije, vrši se prihvatanje bajtova koji stižu preko Ethernet-a, za TCP. Poziva se funkcija *SPI_Ethernet_getByte(),* čime se prvih jedanaest bajtova ubacuju u niz *getRequest[]*. Po izlasku iz petlje poslednji član ovog niza (*getRequest[11]*) dobija vrednost 0. Prvih pet članova niza *getRequest[]* je definisano za *httpMethod* string. Neposredno pre *for* petlje izvršeno je paljenje diode postavljanjem porta *PORTA.F4* na 1.

U konzoli Web Browser-a kucamo karaktere koji se upisuju u niz *getRequest[]*, počev od pozicije 5. Na poziciji 5 naveden je tip komande, na pozicijama 6 i 7 nalazi se ID Slave automata, na poziciji 8 broj programa i na pozicijima 9 i 10 zadata osvetljenost. Ispitujemo vrednost karaktera sa indeksom 5. Ukoliko on odgovra karakteru 's', vrši se ispisivanje svih informacija, koje je Master automat prosledio preko Etherneta CRS-u, u prozoru Web Browser-a. A ukoliko on odgovara karakteru 'p' izvršiće se definisanje programa, obradiće se zahtevi koji pristižu iz Web Browser-a i vratieć se informacije o stanjima traženog Slave automata. CRS-u se preko Etherneta prosleđuju informacije u vidu stringa koji se formira pozivom funkcije *FormirajNiz()*, pored čega šaljemo još i neke konstante fajlove, neke, unapred definisane stringove, neki header, html fajl. Usput brojimo karaktere, a oni se broje u promenljivoj *len*. Ona sadrži broj karaktera koji smo poslali preko Ethernet-a i uvek kada pozovemo neku fukncijju za slanje stringa, mi uvećavamo *len*. U ovom delu koda iskorišćene su i funkcije *putConstString* i *putString* koje nisu deo biblioteke *mikroC* kompajlera. Razlika u njihovom kodu je ta da li se šalje konstantni string ili onaj koji je promenljiv.



```
unsigned int SPI_Ethernet_UserTCP(unsigned char *remoteHost, unsigned int remotePort, unsigned int localPort, unsigned int reqLength, char *canClose)
{
    unsigned char brslejva, brprog, osv;

    unsigned int len = 0; // my reply length
    unsigned int i;       // general purpose integer
    if (localPort != 80)
        return (0);
    PORTA.F4 = 1; // pali se dioda
    // get 10 first bytes only of the request, the rest does not matter here
    for (i = 0; i < 11; i++)
        getRequest[i] = SPI_Ethernet_getByte();
    getRequest[i] = 0;
    if (memcmp(getRequest, httpMethod, 5))
        return (0);

    // only GET method is supported here
    if (getRequest[5] == 's')
    {
        // primio komandu za prozivanje slejvova
    }
    else if (getRequest[5] == 'p')
    {
        brslejva = ((getRequest[6] & 0x0F) * 10) + (getRequest[7] & 0x0F);
        if (brslejva > 15)
            brslejva = 15;

        brprog = getRequest[8] & 0x07;
        if (brprog > 4)
            brprog = 4;

        osv = ((getRequest[9] & 0x0F) * 10) + (getRequest[10] & 0x0F);
        if (osv > 99)
            osv = 99;
```

Slika 16: Prikaz f-je *SPI_Ethernet_UserTCP()* u "*Microsoft Visual Studio"* razvojnom okruženju (1. deo)

```
        brprog = getRequest[8] & 0x07;
        if (brprog > 4)
            brprog = 4;

        osv = ((getRequest[9] & 0x0F) * 10) + (getRequest[10] & 0x0F);
        if (osv > 99)
            osv = 99;

        PROGRAM[brslejva] = brprog;
        OSVETLJENOST_SOBE[brslejva] = osv;
    }
    else
        return 0;

    if (len == 0)
    {
        //kopira niz u kome su smesteni odgovori prozvanih slejvova u
        //pravi niz koji se salje preko etherneta
        FormirajNiz();
        len = putConstString(httpHeader); // HTTP header
        len += putConstString(httpMimeTypeHTML);
        len += putString(niz);
    }
    return (len);
}
```

Slika 17: Prikaz f-je *SPI_Ethernet_UserTCP()* u "*Microsoft Visual Studio"* razvojnom okruženju (2. deo)

8. POSTUPAK VERIFIKACIJE I VREDNOVANJA PROJEKTA

Na slici 18 možemo videti uspešno prikazivanje vrednosti sa Slave automata broj 00 na LCD displeju Master automata.

Slika 18: Prikaz Mastera i Slave automata sa ispisanim vrednostima na LCD displejima

Sada ćemo proveriti ispravnost svih pet režima rada:

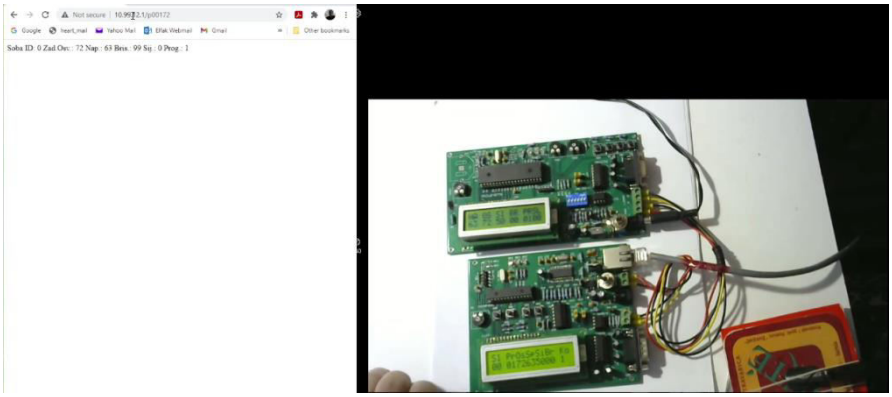
1. Program 0 :

Da bismo odabrali program 0 u komandnu linju Browser-a unosimo *10.99.12.1/p00072*. *10.99.12.1* predstavlja IP adresu Master automata, prve dve nule su ID broj Slave automata (00), sledeća vrednost je redni broj programa (0), a (72) je vrednost zadate osvetljenosti. Rezultat je prikazan na slici 19.

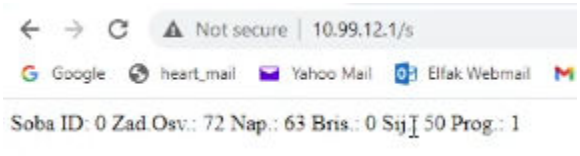
Slika 19: Prikaz provere ispravnosti prvog režima rada (Program0)

2. Program 1:

Za program 1 unosimo komandu /p00172. Rezultati se vide na pločama koje su zajedno sa prozorom Web Browser-a prikazane na slici 20. Nakon korišćenja komande /s rezultat će se ispisati i u prozoru Browser-a, slika 21.



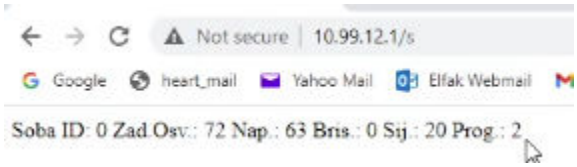
Slika 20: Prikaz provere ispravnosti drugog režima rada (Program1) na Master i Slave automatima



Slika 21: Prikaz provere ispravnosti drugog režima rada (Program1) u prozoru Web Browser-a

3. Program 2:

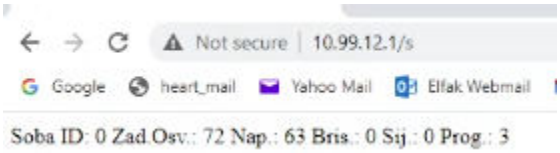
Komanda za program 2 je /p00272. Stanja Slave automata se mogu videti na slici 22.



Slika 22: Prikaz provere ispravnosti trećeg režima rada (Program2)

4. Program 3:

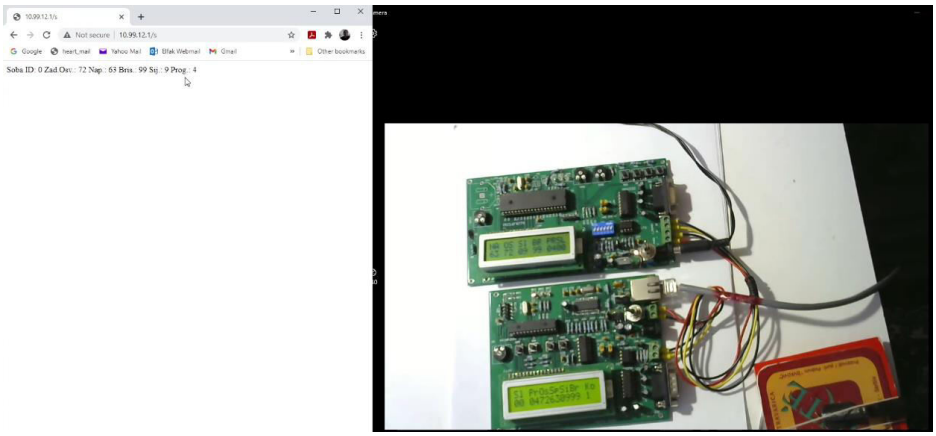
Komanda za program 3 je /p00372. Stanja Slave automata se mogu videti na slici 23:



Slika 23: Prikaz provere ispravnosti četvrtog režima rada (Program3)

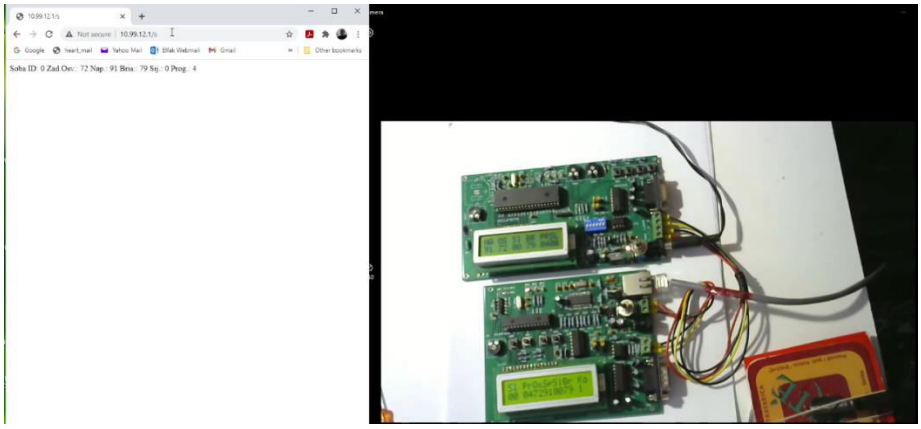
5. Program 4:

Komanda za program 4 je /p00472. Stanja Slave automata se mogu videti na slici 24.



Slika 24: Prikaz provere ispravnosti petog režima rada (Program4)

Slučaj kada je osvetljenje napolju veće od zadatog osvetljenja možemo videti na slici 25.



Slika 25: Prikaz provere ispravnosti petog režima rada (Program4), kada je spoljno osvetljenje veće od zadatog

9. DETALJNO UPUTSTVO ZA UPOTREBU

U objektu se nalazi 16 prostorija, u svakoj prostoriji se nalazi po jedan Slave automat, svaki povezan sa Master automatom koji je povezana sa PC računarom. Na samom PC-u, u Web Browser-u (npr. Google Chrome), u komandnoj liniji potrebno je navesti IP adrese Master automata, redni broj Slave automata, redni broj programa, kao i vrednost zadate osvetljenosti. Ovi podaci se sa Master automata prosleđuju na Slave automat koji izvršava neophodne promene osvetljenja.

Pomoću tastera na Master automatu prozivaju se Slave automati. LCD displej Master automata prikazuje trenutni ID, broj Slave automata, program, zadatu osvetljenost, spoljnu osvetljenost, vrednost jačine sijalica, brisoleja, kao i da li postoji komunikacije izmedju Master i Slave automata (ako je 1 – postoji veza, ako je 0 – nije uspostavljena veza).

Osvetljenje napolju se zadaje preko potenciometra Slave automata koji simulira senzor. Zadato osvetljenje se zadaje u komandoj liniji Web Browser-a. U komandnu liniju Web Browser-a prvo unosimo IP adresu (10.99.12.1) zatim /pXXYZZ. Gde XX predstavlja redni broj ploče, može imati vrednosti od 00-15. Y je program (0-4). A ZZ je zadata osvetljenost, koja može imati vrednost od 00 do 99. Nakon što smo uneli poruku u komandu liniju, pritiskom tastera Enter na našoj tastaturi ova poruka se prosleđuje Master automatu. Ukoliko želimo da se u prozoru našeg Web Browser-a ispišu vrednosti sa trenutno odabranog Slave automata, u komandnu liniju Web Browser-a unosimo 10.99.12.1/s.