Univerzitet u Nišu
Elektronski fakultet
Katedra za elektroniku

SISTEM ZA ZAŠTITU OD POŽARA
MASTER AUTOMAT

Seminarski rad iz predmeta Projektovanje elektronskih sistema

Mentor:
Prof. dr. Borisav Jovanović
Prof. dr. Miljana Milić

Student:
Aleksandar Mladenović17291

Sadržaj:

1. Opis zadatka

Projektovati sistem za sprećavanje požara u jednoj poslovnoj zgradi. U zgradi postoje ukupno 30 prostorija koje su smeštene na 3 sprata i u kojima se nikako ne sme dozvoliti pojava požara.

Sistem kontinuirano prati temperaturu i koncentraciju dima u prostorijama i na osnovu tih vrednosti sa sigurnošću može da utvrdi da li je došlo do pojave požara ili ne. Ukoliko se javio požar, u toj prostoriji se aktivira zvučni alarm, koji zaposlenima signalizira da u roku od 15 sekundi moraju da napuste prostoriju. Po isteku 15 sekundi, automatski se hermetički zatvaraju i zaključavaju sva vrata, tako da nije moguć ni ulaz u prostoriju, niti izlaz iz nje. Nakon toga u prostoriju se ubacuje specijalni gas, koji brzo vezuje kiseonik iz vazduha. Kada se sav kiseonik iz vazduha potroši (što se može detektovati pomoću senzora koncentracije kiseonika), dalje širenje plamena nije moguće i prestaje se sa ubacivanjem specijalnog gasa. Radi sigurnosti da vatra neće ponovo da se javi, prostorija ostaje bez kiseonika narednih 5 minuta. Nakon tog perioda, aktivira se sitem koji normalizuje atmosferu u prostoriji i tek kada je ona normalizovana, otključavaju se vrata i ponovo se omogućava ulazak.

U sistem postoji centralizovano mesto na kome se:

- prati stanje temperature, koncentracije dima i koncentracije kiseonika u svakoj od prostorija;
- daje grafička i zvučna indikacija alarma kada u nekoj od prostorija počne stvaranje požara;
- prati stanje ispravnosti senzora i aktuatora u prostorijama;
- može aktivirati lažni alarm u datoj prostoriji za potrebe vežbe u slučaju požara (tom prilikom se u prostoriji dešava sve kao i u slučaju pravog požara, osim što se ne ubacuje specijalni gas);
- vode dnevne, sedmične, mesečne I godišnje arhive događaja (alarmi I vežbe).

2. Funkcionalna i nefunkcionalna specifikacija sistema

a) Funkcionalne specifikacije sistema:

- Sistem se sastoji od jednog Master-a i 32 Slave uređaja, koji se instaliraju u svakoj pojedinačnoj sobi.
- Master automat je centralni uređaj koji se koristi 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 prikupljaju informacije sa senzora koji su instalirani u nekoj prostoriji.

    - **PIR** senzor proverava da li ima ljudi u prostoriji
    - **SMOKE** sensor proverava koncentraciju dima u prostoriji
    - **O2** senzor proverava kiseonik u prostoriji
    - **DRILL** aktivira se lažni alarm u slučaju vežbe

Slave automat takođe poseduje i izlaze (LED diode):

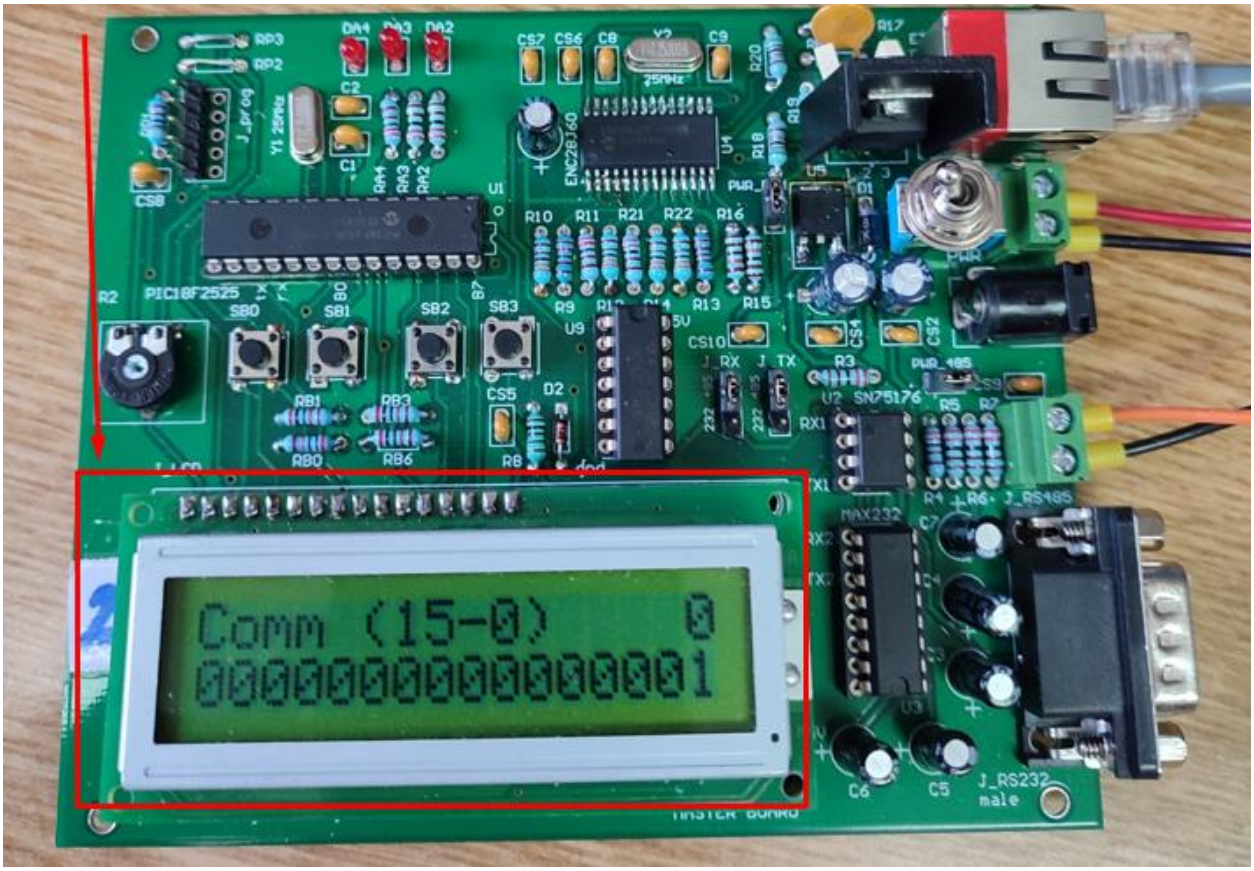
- **GAS** signalizira da je počelo ubacivanje specijalnog gasa za gašenje požara
- **DOOR_OPEN** pokazuje da li su vrata otvorena ili zatvorena
- **SOUND** simulira alarm u slučaju da se javio požar
- **VENT** pokazuje da li u prostoriji ima ventilacije

Slave automati prosleđuju prikupljene informacije sa senzora dalje Master 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 prostorijama, pri čemu se Slave-u pojedinačno dodaje komanda, a prozvani Slave odgovara na pristiglu komandu.

Pored komunikacije sistem treba da radi sledeće:

- Na Master-u se proverava i prati situacija u prostorijama zgrade.
- Ukoliko je došlo do požara oglašava se alarm.
- Ukoliko je temperatura iznad dozvoljene, aktivira se alarm.
- Ukoliko senzor detektuje povećanu koncentraciju dima, takođe se aktivira alarm.

Master automat poseduje LCD displej na kome su prikazana stanja svih ulaza (senzora) i izlaza (LED diode) za svih 32 Slave automata. Takođe na LCD displeju se prikazuje i da li postoji komunikacija između određenog Slave automata i Master automata.

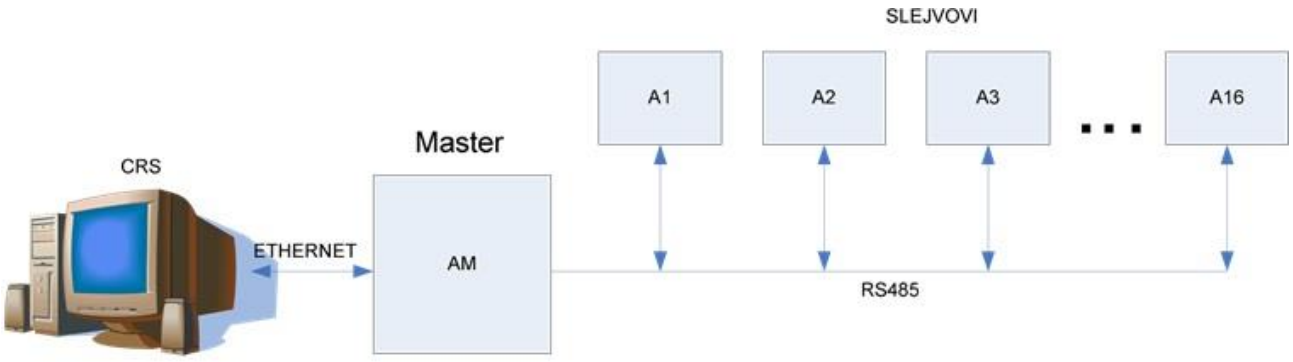


b) Nefunkcionalne specifikacije sistema:

- Sistem treba biti izolovan u pogledu spoljašnjih uslova kao što su visoka temperature i prisustvo povećane koncentracije dima, jer radi u zatvorenim uslovima gde može da se javi visoka temperature kao i povećana koncentracija dima..
- Server računar mora biti opremljen dodatnim diskom i opremom za backup podataka.
- Master i Slave uređaji napajaju se preko elektroenergetske mreže AC napona napajanja od 220V, i to su uređaji koji imaju malu potrošnju.

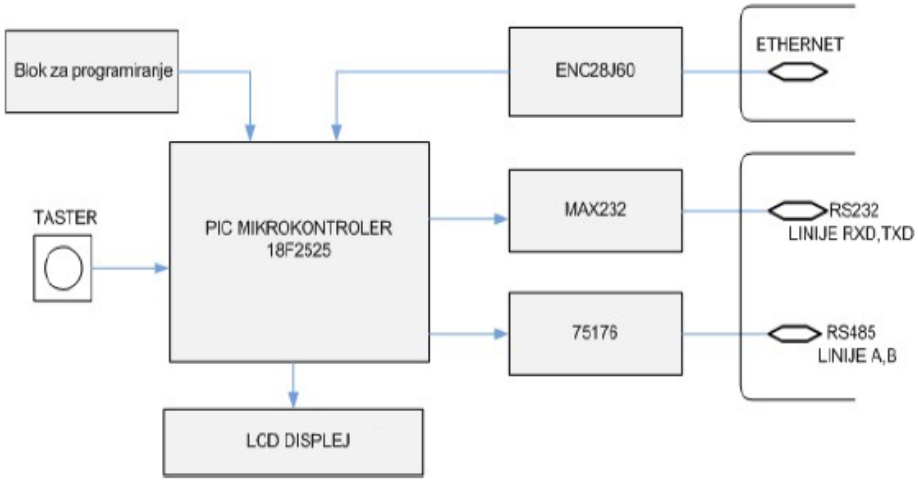
3. Arhitektura Sistema

Arhitektura celokupnog Sistema sastoji se iz Centralnog računarskog Sistema, Master automata i 32 Slave automata, koji preko RS485 komuniciraju sa Master automatom (slika 3.1). Centralni računarski sistem je povezan sa Master automatom preko Etherneta.

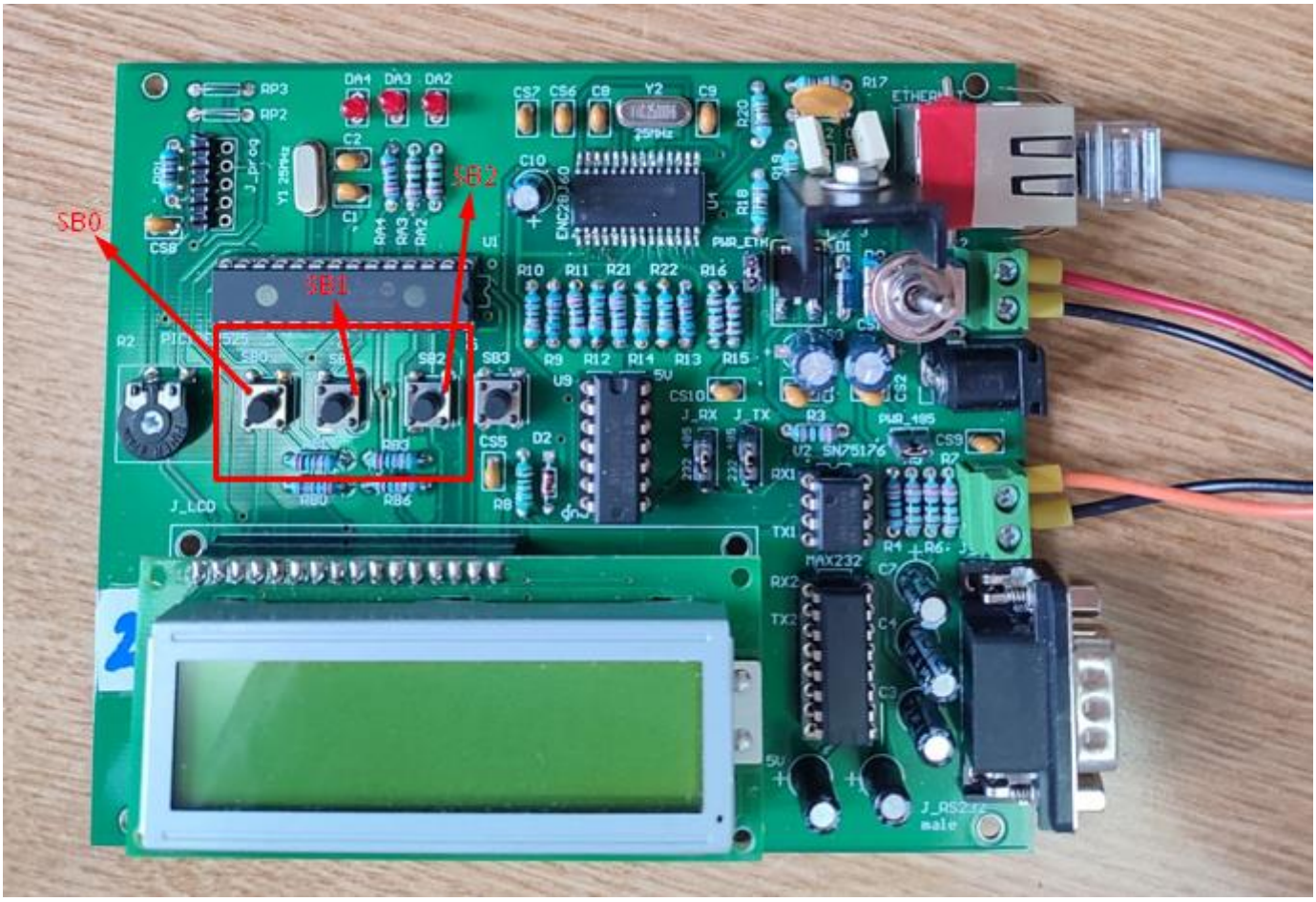


Slika 3.1. Arhitektura sistema

Osnovna blok šema Master automata prikazana je na slici 3.2. Rad automata zasnovan je na mikrokontroleru. Ugrađen je Microchip PIC18F2525. Pored mikrokontrolera Master obuhvata blokove za komunikaciju RS232/485(kola drajvera MAX232 i 75176), blok za Ethernet komunikaciju(kolo drajvera ENC28J60), blok za programiranje mikrokontrolera i LCD displej. Takođe, Master automat poseduje i dva tastera koji služe za menjanje stanja na LCD displeju i jedan taster za odabir Slave-a (0-15. ili 16-32. Slave).



Slika 3.2. Blok šema Master automata



SB0 taster – Down taster za menjanje stanja na LCD displeju.

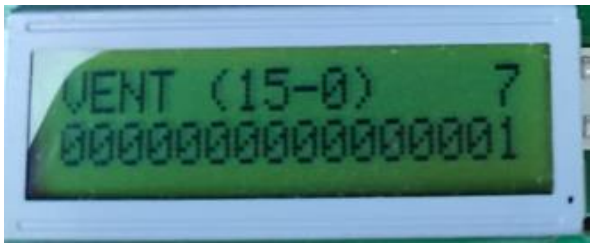
SB1 taster – UP taster za menjanje stanja na LCD displeju.

SB2 taster – Odabir izbora Slave-a (0-15 ili 16-32, slika 3.3.).

Slika 3.3. Prikaz pomoću SB2 tastera

Na LCD displeju Master automata prikazuju se statusi ulaznih i izlaznih promenljivih za svih 32 Slave-a, kao i da li postoji komunikacija izmedju Master automata i određenog Slave automata.

- **GAS** – da li se ubacuje specijalni gas (1-ubacuje se specijalni gas, 0- ne ubacuje se specijalni gas)
- **VENT** – da li postoji ventilacija u prostoriji (1-postoji ventilacija, 0- ne postoji ventilacija)
- **SMOKE** – da li ima dima u prostoriji(1- ima dima u prostoriji, 0- nema dima)
- **O2** – da li ima kiseonika u prostoriji (1- ima kiseonika u prostoriji, 0- nema kiseonika)
- **DOOROpen** – da li su vrata otvorena (1- vrata otvorena, 0- vrata zatvorena)
- **PIR** – da li ima ljudi u prostoriji (1- ima ljudi u prostoriji, 0- nema ljudi u prostoriji)
- **SOUND** – da li se oglasio alarm u prostoriji (1- alarm se oglasio, 0- alarm je ugašen)
- **DRILL** – da li je aktiviran lažni alarm za potrebe vežbe(1- aktivirana je vežba, 0- nije aktivirana vežba)



Slike iznad predstavljaju prikaz na LCD displeju

4. Hardverska realizacija Master automata

Električna šema sadrži mikrokontroler PIC18F2525, LCD displej dimenzija 16x2, drajvere za RS232 i RS485, blok za napajanje, blok za programiranje mikrokontrolera, LED diodu koja se koristi za signalizaciju postojanja komunikacije između Master-a i Slave-a, kao i dodatne prekidače i tastere.

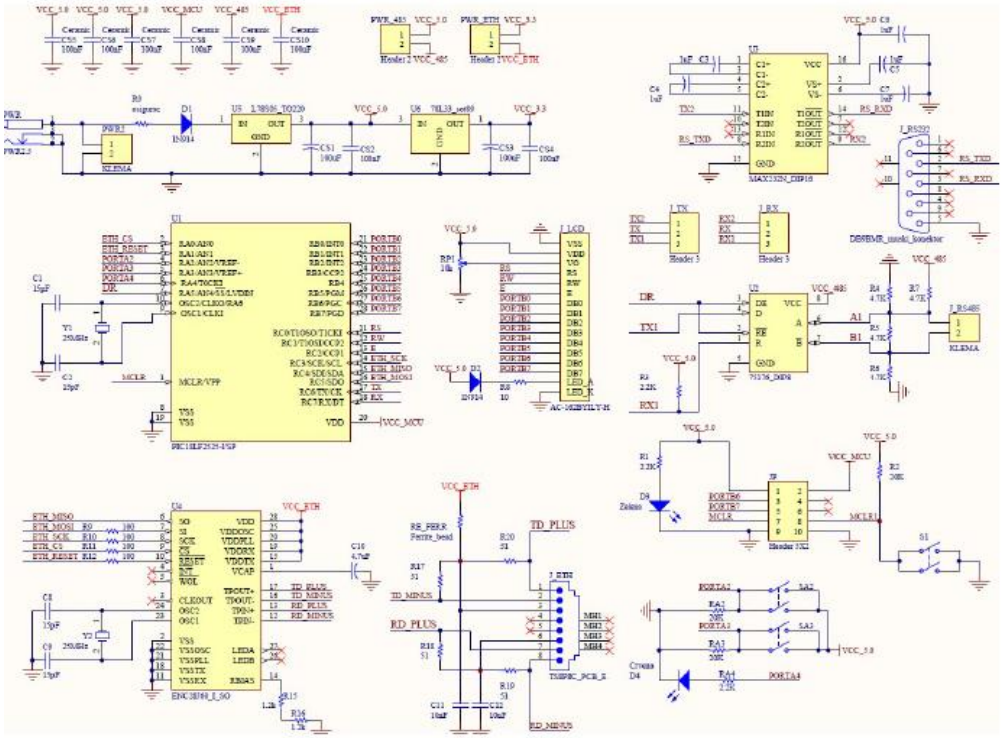
Centralna jedinica Master-a je Microchip-ov mikrokontroler PIC18F2525. Korišćeni mikrokontroler ima SPDIP kućište i radi na 25MHz i napona napajanja od 5V.

Osnovne karakteristike su sledece:

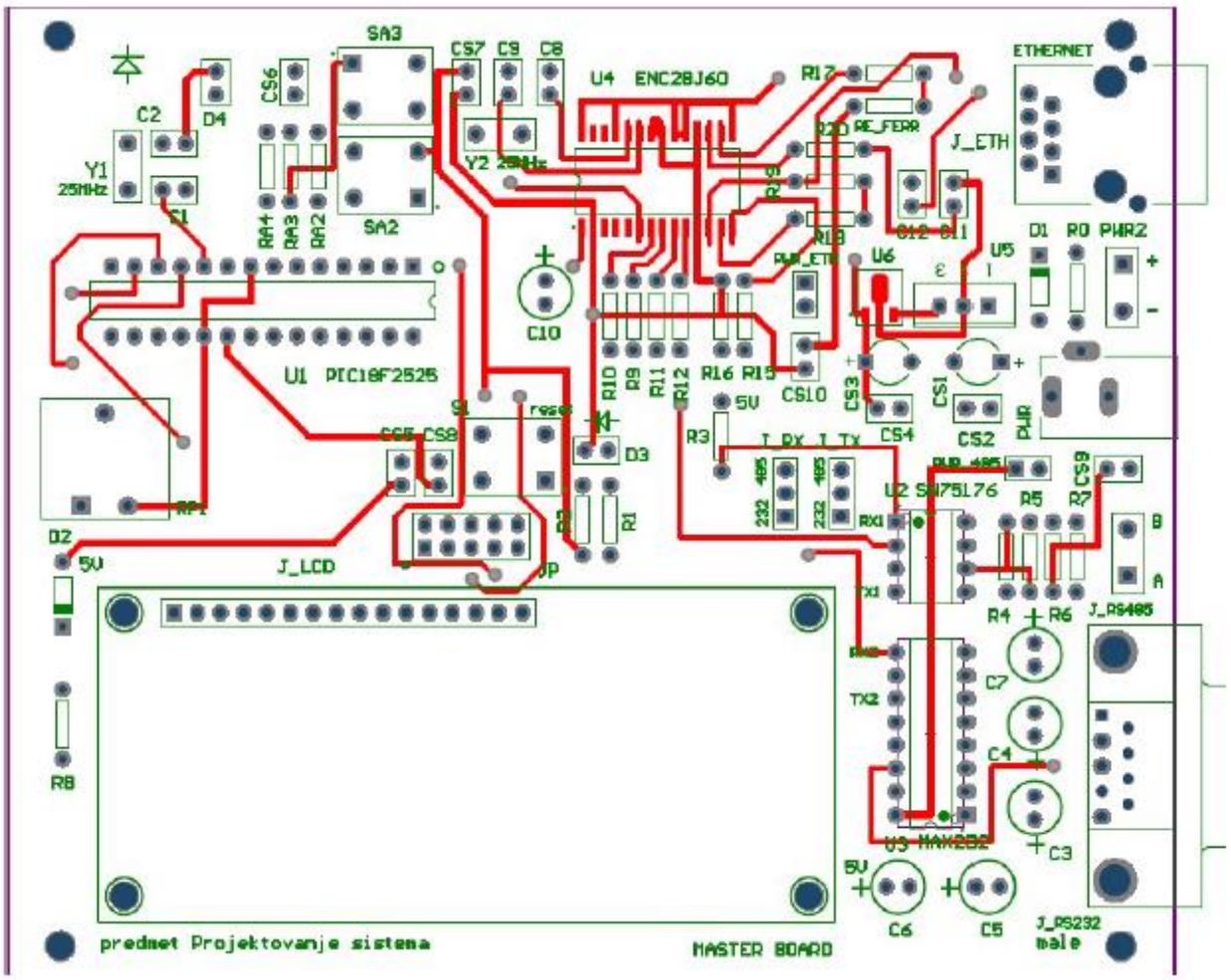
- mikrokontroler ima 48KB ROM memorije
- RAM memoriju od 3986B
- I2C i SPI komunikacioni interfejs
- 10-bitni A/D konvertor do 13 kanala

Postoje 3 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.

Električna šema i PCB dijagram prikazani su na slici 5.1. i 5.2.



Slika 5.1. Električna šema



Slika 5.2. PCB dijagram

5. Realizacija komunikacionog protokola

Projektovanje komunikacije između Master i Slave automata je jedan 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.

Kako bi Master dobio informacije od određenog Slave–a, neophodno je da svaki Slave ima svoj jedinstveni ID(identifikacioni) broj.

Master Slave–ovima salje komandu prozivke 0xA0, što znači da su prva tri bita "101". Preostala pet bita "XXXXX" predstavljaju ID(identifikacioni) broj Slave-ova.

ID broj Slave-a(4:0)

1	0	1	X	X	X	X	X

Komanda prozivke(7:5)

Kao odgovor Slave-a na poruku komandne prozivke, Slave vraća isti komandi bajt, statusni bajt svih ulaznih i izlaznih promenljivih, i statusni bajt za temperaturu. Ova poruka se sastoji od tri bajta.

ID broj Slave-a(4:0)

1	0	1	X	X	X	X	X

Komanda prozivke(7:5) Komandni bajt

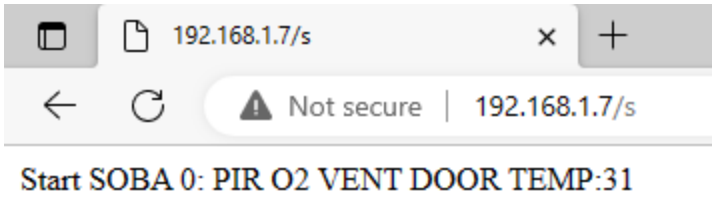
GAS	VENT	DOOROpen	SOUND	PIR	SMOKE	O2	DRILL

Statusni bajt svih ulaznih i izlaznih promenljivih

X10 (mesto gde se upisuju cifre desetice)	X1 (mesto gde se upisuju cifre jedinice)

Statusni bajt za temperature

Komunikacija između CRS i Master-a se ostvaruje putem Ethernet kabla. Za praćenje rada celog sistema koristi se WEB pretraživač, a IP adresa samog uređaja se definiše u kodu i njena vrednost je: 192.168.1.7. U komandnoj liniji WEB browsera unese se http://192.168.1.7/s .



Primer na WEB pretraživaču

6. Realizacija firmware-a Master automata

Kod Master automata, kao i Slave automata, realizovan je u programskom jeziku C. Za realizaciju i kompajliranje koda korišćen je program pod nazivom „microC PRO for PIC“. Za realizaciju projekta korišćen je mikrokontroler PIC18F2525 koji radi na 25MHz, koje je potrebno naznačiti u programu pre početka rada. Nakon toga, kreće se sa pisanjem koda. Kod se počinje definisanjem makroa, a zatim i definisanju promenljivih, funkcije prekida *interrupt()*, još par dodatnih pomoćnih funkcija kao što su *formirajNiz(), dodajuNiz(), putString()* i još mnoge druge, i na kraju glavne *main()* funkcije. Na slici 6.1 je prikazana tabela makroa i pinova koji se definišu na samom početku. Na slici 6.2 je prikazana tabela najvažnijih funkcija koje se koriste u programskom kodu.

Naziv makroa ili pina	Dodeljena vrednost	Opis
SPI_Ethernet_HALFDUPLEX	0	Taster za promenu prikaza na LCD.
SPI_Ethernet_FULLDUPLEX	1	LED koja se uključuje nakon prozivke.
CTRL	PORTA.F5	Definicija pina koji se koristi kao LED dioda koja se uključuje pri slanju poruke.

Slika 6.1. Tabela makroa i pinova

Naziv funkcije	Tip	Opis
init_variables()	void	Inicijalizuju se početne vrednosti promenljivih.
convert()	unisgned int	Izvlači cifre desetice i jedinice iz broja.
dodajuNiz(char *p_ch)	void	Dodaje elemente u niz.
formirajNiz()	void	Formira niz čije elemente čine pojedini odgovori svih 32 Slave-a.
LCDdisp()	void	Definiše se šta će prikazivati LCD displej.
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 pretraživača i izvršava se slanje podataka preko Etherneta.
interrupt()	void	Koristi se za obradu prekida(prekid serijske komunikacije i tajmera1).
main()	void	Glavna funkcija u kojoj se izvršavaju prethodno navedene funkcije.

Slika 6.2. Tabela najbitnijih funkcija

a) Funkcija *interrupt():*

Funkcija *interrupt()* je jedna od najvažnijih funkcija i koristi se za obradu prekida mikrokontrolera. Prekidi koji se obrađuju su prekid serijske komunikacije i prekid tajmera1. Funkcija *interrupt()* je tipa *void*, što znači da ona ne vraća nijedan parametar.

Prekid tajmera1:

Programski kod koji je korišćen za obradu prekida tajmera1 prikazan je na slici 6.3. Programski kod se izvršava jednom nakon intervala od 25ms, koliko traje perioda Tajmera1.

Prvo što se mora ispititi je uzrok prekida (da li je prekid nastao zbog tajmera1 ili zbog serijske komunikacije). Proveravaju se bit dozvole TMR1IE i flag bit TMR1IF. Ako je došlo do prekida resetuje se flag TMR1IF kako bi naredni prekid tajmera1 mogao biti detektovan. Zatim se ispituje da li je *br >= 0x04*, to znači da se na svakih 125ms proziva po jedna prostorija (Slave). Ukoliko je uslov ispunjen, *br* dobija vrednost 0 i podiže flag koji govori da je došlo vreme da se prozove prostorija (Slave). Ako uslov nije ispunjen, *br* se inkrementira za 1.

Nakon toga se ispituje vrednost brojača tt, koji se koristi za eliminaciju treperenja tastera PORTB.F0 i tastera PORTB.F1.

Na kraju programskog koda koji se bavi obradom prekida tajmera1, postavljaju se početne vrednosti TMR1L = 0xB5; i TMR1H = 0Xb3;

```
if ((PIE1.TMR1IE) && (PIR1.TMR1IF))
    {
        // prekid tajmera1 na svakih 25ms
        PIE1.TMR1IE = 1;
        PIR1.TMR1IF = 0;
        if (br >= 0x04)
        { // na svakih  125ms se proziva po jedna prostorija
            br = 0x00;
            Flag1 = 0x01; // podigo se flag koji govori da je doslo vreme da se
prozove slave(prostorija)
        }
        else
            br++;
        if (tt > 0) // treperenje tastera
            tt--;

        if ((PORTB.F0 == 1) && (tt == 0))
        {
            if (stanjeDisp == 8)
                stanjeDisp = 0;
            else
                stanjeDisp++;
            tt = 20;
        }
        if ((PORTB.F1 == 1) && (tt == 0))
        {
            if (stanjeDisp == 0)
                stanjeDisp = 8;
            else
                stanjeDisp--;
            tt = 20;
        }
        if ((PORTB.F2 == 1) && (tt == 0))
        {
            if (Offset == 0)
                Offset = 16;
            else
                Offset = 0;
            tt = 20;
        }

        TMR1L = 0xB5;
        TMR1H = 0xB3;
    }
```

Slika 6.3. Prekid tajmera1

Prekid serijske komunikacije:

Programski kod koji je korišćen za obradu prekida nastalog zbog serijske komunikacije dat je na slici 6.4. i slici 6.5, kao i slici 6.6.

U kodu se najpre ispituje uzrok prekida. Proveravaju se bitovi signala dozvole RCIE i flag bit RCIF. Ovime se provera da li je omogućen prekid nastao zbog serijske komunikacije i da li se desio prekid. Ukoliko jeste, pali se crvena LED dioda koja signalizira da je došlo do prekida i resetuje se flag RCIF da bi naredni prekid bio detektovan. Nakon toga, vrednost primljena preko UART-a smešta se u promenljivu pod nazivom *ch*. Nakon toga, prima se bajt koji se dekodira i odlučuje se da li još treba da se primaju bajtovi. Prvi bajt koji se prima je bajt prozivke koji Slave vraća Master-u nakon što je prozvan. Ispituje se da li se ch ID broj poklapa sa ROOM_ID brojem Slave-a. U komandi ch ID broj Slave automata se nalazi na bit pozicijama od 4 do 0. Ako je uslov ispunjen, komunikacija je dobra (deo koda je prikazan na slici 6.4.).

```
if ((PIE1.RCIE == 1) && (PIR1.RCIF == 1))
    { // prekid zbog serijske komunikacije
        PIR1.RCIF = 0;
        ch = RCREG; // prima se bajt preko UART-a
        if (OBB != 0x00)
        { // ocekuje se bar 1 bajt
            if (OBB == 0x03)
            { // prijem bajta komande koji se dekodira
                // i onda se odlucuje da li jos treba da se primaju bajtovi

                //  "000"&ID(4:0)
                if (((ch & 0x1F) == ROOM_ID) && ((ch & 0xE0) == 0xA0))
                {
                    //Comm[ROOM_ID] = 1;
                    OBB = 2;
                } // komunikacija je dobra
            }
```

Slika 6.4.

Nakon toga Master automat prima bajt koji sadrži stanja ulaznih i izlaznih promenljivih od određenog Slave automata. Ispituje se da li su bitovi na odgovarajućim pozicijama (prikazano je na slici 6.5.).

Potom se prima bajt koji sadrži vrednost temperature (prikazano na slici 6.6.).

```
else if (OBB == 0x02) // statusni bajt
            {
                //  GAS&VENT&DOOR&SOUND&PIR&SMOKE&O2&DRILL
                if ((ch & 0x01) == 0x01)
                    DRILL[ROOM_ID] = 1;
                else
                    DRILL[ROOM_ID] = 0;
                if ((ch & 0x02) == 0x02)
                    O2[ROOM_ID] = 1;
                else
                    O2[ROOM_ID] = 0;
                if ((ch & 0x04) == 0x04)
                    SMOKE[ROOM_ID] = 1;
                else
                    SMOKE[ROOM_ID] = 0;
                if ((ch & 0x08) == 0x08)
                    PIR[ROOM_ID] = 1;
                else
                    PIR[ROOM_ID] = 0;

                if ((ch & 0x10) == 0x10)
                    SOUND[ROOM_ID] = 1;
                else
                    SOUND[ROOM_ID] = 0;
                if ((ch & 0x20) == 0x20)
                    DOOR[ROOM_ID] = 1;
                else
                    DOOR[ROOM_ID] = 0;
                if ((ch & 0x40) == 0x40)
                    VENT[ROOM_ID] = 1;
                else
                    VENT[ROOM_ID] = 0;
                if ((ch & 0x80) == 0x80)
                    GAS[ROOM_ID] = 1;
                else
                    GAS[ROOM_ID] = 0;

                OBB--; // OBB=1;
            }
```

Slika 6.5.

```
else if (OBB == 0x01) // Statusni za temperaturu
            {
                Temperature[ROOM_ID] = ch;
                OBB = 0;
            }
```

Slika 6.6.

b) Funkcija *LCDdisp():*

Ova funkcija služi za ispisivanje stanja senzora na LCD displej. Tip funkcije je *void,* tako da ne vraća nikakvu povratnu vrednost. Kod je prikazan na slikama ispod.

```
void LCDdisp()
{
    int i = 0;
    if (stanjeDisp == 1)
    {
        if (Offset == 0)
            LCD_Out(1, 1, "DRILL (15-0)   1");
        else
            LCD_Out(1, 1, "DRILL (31-16)  1");
        for (i = 0; i < 16; i++)
            if (DRILL[i + Offset] == 1)
                LCD_Chr(2, 16 - i, '1');
            else
                LCD_Chr(2, 16 - i, '0');
    }
    else if (stanjeDisp == 2)
    {
        if (Offset == 0)
            LCD_Out(1, 1, "O2 (15-0)      2");
        else
            LCD_Out(1, 1, "O2 (31-16)     2");
        for (i = 0; i < 16; i++)
            if (O2[i + Offset] == 1)
                LCD_Chr(2, 16 - i, '1');
            else
                LCD_Chr(2, 16 - i, '0');
    }
    else if (stanjeDisp == 3)
    {
        if (Offset == 0)
            LCD_Out(1, 1, "SMOKE (15-0)   3");
        else
```

```
            LCD_Out(1, 1, "SMOKE (31-16)  3");
        for (i = 0; i < 16; i++)
            if (SMOKE[i + Offset] == 1)
                LCD_Chr(2, 16 - i, '1');
            else
                LCD_Chr(2, 16 - i, '0');
    }
    else if (stanjeDisp == 4)
    {
        if (Offset == 0)
            LCD_Out(1, 1, "PIR (15-0)     4");
        else
            LCD_Out(1, 1, "PIR (31-16)    4");
        for (i = 0; i < 16; i++)
            if (PIR[i + Offset] == 1)
                LCD_Chr(2, 16 - i, '1');
            else
                LCD_Chr(2, 16 - i, '0');
    }
    else if (stanjeDisp == 5)
    {
        if (Offset == 0)
            LCD_Out(1, 1, "SOUND (15-0)   5");
        else
            LCD_Out(1, 1, "SOUND (31-16)  5");
        for (i = 0; i < 16; i++)
            if (SOUND[i + Offset] == 1)
                LCD_Chr(2, 16 - i, '1');
            else
                LCD_Chr(2, 16 - i, '0');
    }
    else if (stanjeDisp == 6)
    {
        if (Offset == 0)
            LCD_Out(1, 1, "DOORopen (15-0)6");
        else
            LCD_Out(1, 1, "DOORopen(31-16)6");
        for (i = 0; i < 16; i++)
            if (DOOR[i + Offset] == 1)
                LCD_Chr(2, 16 - i, '1');
            else
                LCD_Chr(2, 16 - i, '0');
    }
    else if (stanjeDisp == 7)
    {
```

```
        if (Offset == 0)
            LCD_Out(1, 1, "VENT (15-0)    7");
        else
            LCD_Out(1, 1, "VENT (31-16)   7");
        for (i = 0; i < 16; i++)
            if (VENT[i + Offset] == 1)
                LCD_Chr(2, 16 - i, '1');
            else
                LCD_Chr(2, 16 - i, '0');
    }
    else if (stanjeDisp == 8)
    {
        if (Offset == 0)
            LCD_Out(1, 1, "GAS (15-0)     8");
        else
            LCD_Out(1, 1, "GAS (31-16)    8");
        for (i = 0; i < 16; i++)
            if (GAS[i + Offset] == 1)
                LCD_Chr(2, 16 - i, '1');
            else
                LCD_Chr(2, 16 - i, '0');
    }
    else if (stanjeDisp == 0)
    {
        if (Offset == 0)
            LCD_Out(1, 1, "Comm (15-0)    0");
        else
            LCD_Out(1, 1, "Comm (31-16)   0");
        for (i = 0; i < 16; i++)
            if (Comm[i + Offset] == 1)
                LCD_Chr(2, 16 - i, '1');
            else
                LCD_Chr(2, 16 - i, '0');
    }
}
```

c) Funkcije *convert(unsigned char number), dodajuNiz(char *p_ch), formirajNiz():*

Funkcija *convert(unsigned char number)* izvlači cifre desetice i cifre jedinice, koji se kasnije koriste za temperaturu. Programski kod ove funkcije je prikazan na slici ispod:

```
void convert(unsigned char number)
{
    X1 = number;
    X10 = 0;
    while (X1 > 9)
    {
        X1 = X1 - 10;
        X10++;
    }
}
```

Funkcija *dodajuNiz(char *p_ch)* predstavlja funkciju koja se korsti radi dodavanja odgovora svakog senzora Slave-a u jedan veliki niz karaktera, koji se zatim šalje Masteru, programski kod je prikazan na slici ispod:

```
void dodajuNiz(char *p_ch)
{
    while ((*p_ch) != 0x00)
    {
        niz[br_ch] = *p_ch;
        br_ch++;
        p_ch++;
    }
}
```

Funkcija *formirajNiz()* je zadužena da formira niz koji će biti sačinjen od odgovora svakog Slave-a, naravno, samo ukoliko postoji komunikacija sa nekim od njih. Programski kod je prikazan na slici ispod:

```
void formirajNiz()
{

    int i = 0;
    char txt[4]; // ID Sobe
    br_ch = 0;   // pozicioniranje na pocetak niza

    for (i = 0; i < 32; i++)
    {
        if (Comm[i] == 1)
        { // verovatno je Comm[i]==1 ako se u prostoriji javio pozar
            dodajuNiz(room);
            ByteToStr(i, txt); // temperatura
            dodajuNiz(txt);
            dodajuNiz(": ");
            if (PIR[i] == 1)
                dodajuNiz("PIR ");
            if (SMOKE[i] == 1)
                dodajuNiz("SMOKE ");
            if (O2[i] == 1)
                dodajuNiz("O2 ");
            if (DRILL[i] == 1)
                dodajuNiz("DRILL ");
            if (GAS[i] == 1)
                dodajuNiz("GAS ");
            if (VENT[i] == 1)
                dodajuNiz("VENT ");
            if (DOOR[i] == 1)
                dodajuNiz("DOOR ");
            if (SOUND[i] == 1)
                dodajuNiz("SOUND ");
            dodajuNiz("TEMP:");
            convert(Temperature[i]);
            niz[br_ch] = X10 + 0x30;
            br_ch++;
            niz[br_ch] = X1 + 0x30;
            br_ch++;
            dodajuNiz("\n\n");
        }
    }
    niz[br_ch] = 0x00;
}
```

d) Funkcija *init_variables():*

Funkcija *init_variables()* služi za inicijalizaciju, odnosno postavljanje početnih vrednosti definisanih promenljivih. Tipa je *void*, što znači da nema povratnu vrednost. Kod ove funkcije dat je na sledećoj slici:

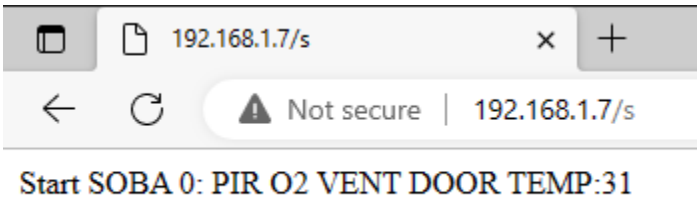
```
void init_variables()
{
    i = 0;
    br = 0x00;
    br_ch = 0x00;
    Flag1 = 0x00;
    ROOM_ID = 0x00;
    OBB = 0x00;// OBB=Ocekivani broj bajtova
    for (i = 0; i < 150; i++)
        niz[i] = 0x00;
    stanjeDisp = 0;

    for (i = 0; i < 32; i++)
    {
        PIR[i] = 0x00;
        SMOKE[i] = 0x00;
        O2[i] = 0x00;
        DRILL[i] = 0x00;
        GAS[i] = 0x00;
        VENT[i] = 0x00;
        DOOR[i] = 0x00;
        SOUND[i] = 0x00;
        Comm[i] = 0x00;
        Temperature[i] = 0x00;
    }
}
```

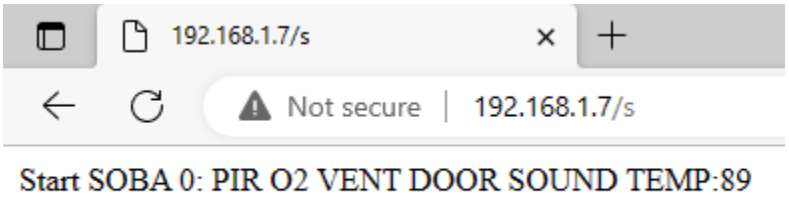
e) Funkcija *SPI_Ethernet_UserTCP(unsigned char *remoteHost, unsigned int remotePort, unsigned int localPort, unsigned int reqLength, char *canClose):*

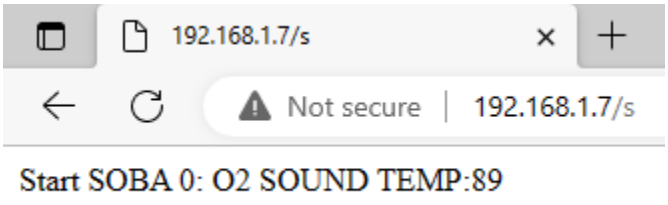
Ova funkcija služi da nakon otvaranja WEB pretraživača i upisa odgovarajuće IP adrese i ID broja Slave-a sa komandom s, ispiše stanja tog Slave-a (192.168.1.7/s). Programski kod ove funkcije prikazan je na sledećoj slici:

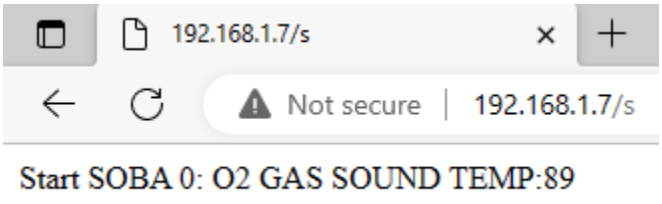
```
unsigned int SPI_Ethernet_UserTCP(unsigned char *remoteHost, unsigned int
remotePort, unsigned int localPort, unsigned int reqLength, char *canClose)
{
    int len = 0; // duzina odgovora
    int i = 0;
    if (localPort != 80)
        return (0); // obradjuje se samo web zahtev na portu 80

    for (i = 0; i < 10; i++)  // B.J. OVDE JE BILO 5!!!
    { // primaju se komande preko Etherneta
        getRequest[i] = SPI_Ethernet_getByte();
    }
    getRequest[i] = 0;

    if (memcmp(getRequest, httpMethod, 5))
        return (0);

    if (getRequest[5] != 's')
    {
        return 0;
    }
    if (len == 0)
    {
        PORTA.F3=1;
        // kopira niz u kome su smesteni odgovori prozvanih slejvova u
        // pravi niz koji se salje preko etherneta
        formirajNiz();
        len = putConstString(httpHeader);
        len += putConstString(httpMimeTypeHTML);
        len += putString("Start\n\n");
        len += putString(niz);
        PORTA.F3=0;
    }
    // vraca se u biblioteku sa brojem bajtova za prenos
    return (len); // return to the library with the number of bytes to transmit
}
```

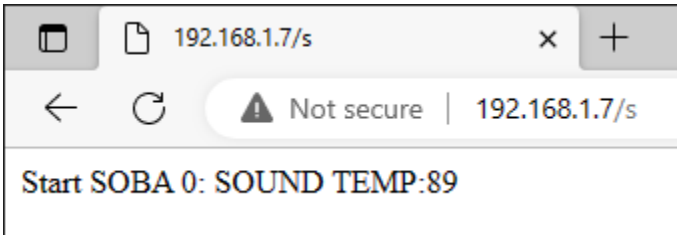
Na sledećim slikama biče prikazana stanja Slave-ova očitana preko web browsera:

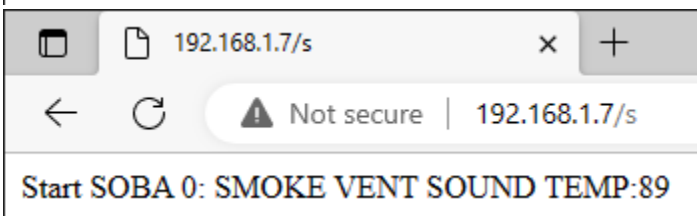












f) Funkcija *main()*:

Funkcija *main()* je glavna funkcija programa. U okviru nje pozivamo prethodno navedene i definisane funkcije *init_variables(), LCDdisp() i init*(). Takođe, vrši se prozivka svih 32 prostorije, koje je ostvareno pomoću brojača koji ide do poslednjeg Slave-a. Kada se stigne do njega, resetuje se, odnosno vrati na nulu, a do tada se inkrementira ID prostorije. Na svakih 125ms se ulazi u petlju i Flag1 se vraća na nulu. Displej se osvežava nakon svakog četvrtog Slave-a. Na sledećoj slici će biti prikazan programski kod glavne funkcije *main()*.

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

    while (1)
    {
        SPI_Ethernet_doPacket();
        if (Flag1 == 1)
        {
            Flag1 = 0;
            if (OBB>0) Comm[ROOM_ID]=0;
            else Comm[ROOM_ID]=1;

            PORTA.F2=0;
            if (ROOM_ID >= 31) {
                ROOM_ID = 0;
                PORTA.F2=1;
            }
            else
                ROOM_ID++;

            ByteX = 0b10100000 + ROOM_ID;
            Ctrl = 1;
            transmit(ByteX);
            Ctrl = 0;
            OBB = 3; // svake sekunde ozvezavanje displeja
            if ((ROOM_ID & 0b00000011) == 0x00)
                LCDdisp();
        }
    }
}
```